Episerver CMS Development Fundamentals

May 2018

Product version: Update 214
Course version: 18.05

Episerver

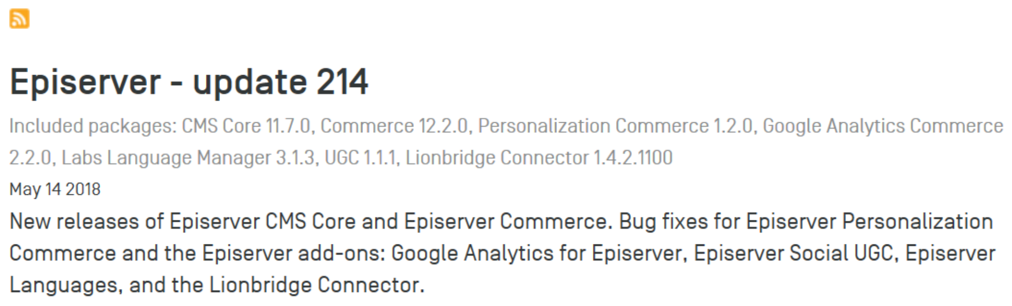
Course title: *Episerver CMS – Development Fundamentals* Course code: 170-3020

Course version: 18.05, 17th May 2018 Product Update 214, 14th May 2018

Episerver CMS Visual Studio Extension version: 11.3.0.359

Episerver CMS packages: EPiServer.CMS.Core 11.7.0, EPiServer.CMS.UI 11.4.4

http://world.episerver.com/releases/

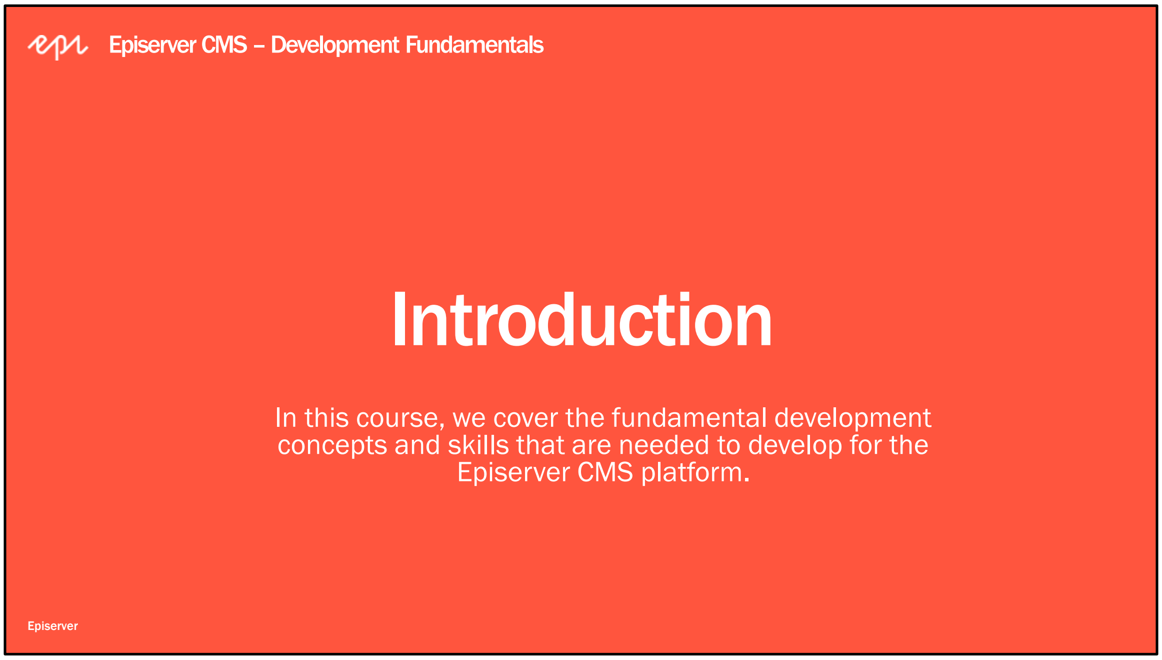


Episerver - update 214

Included packages: CMS Core 11.7.0, Commerce 12.2.0, Personalization Commerce 1.2.0, Google Analytics Commerce 2.2.0, Labs Language Manager 3.1.3, UGC 1.1.1, Lionbridge Connector 1.4.2.1100

May 14 2018

New releases of Episerver CMS Core and Episerver Commerce. Bug fixes for Episerver Personalization Commerce and the Episerver add-ons: Google Analytics for Episerver, Episerver Social UGC, Episerver Languages, and the Lionbridge Connector.



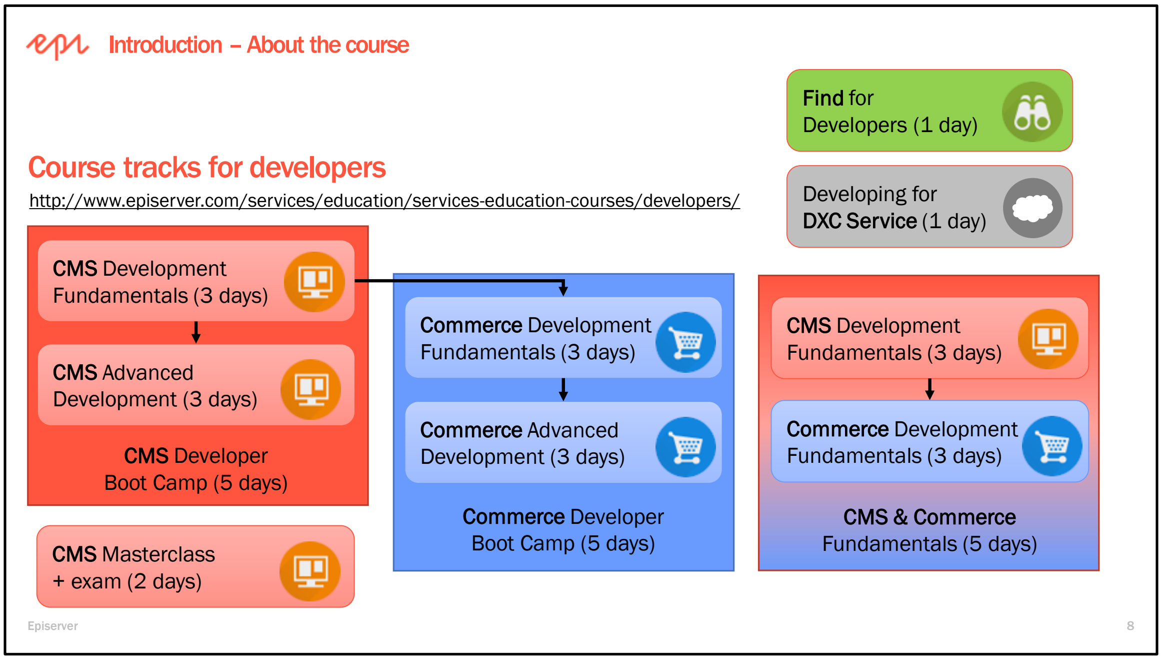
Episerver CMS – Development Fundamentals

Introduction

In this course, we cover the fundamental development concepts and skills that are needed to develop for the Episerver CMS platform.

Episerver

Prerequisites are experience with Microsoft Visual Studio 2015 or later, ASP.NET MVC, and web front end technologies.



Introduction – About the course

Course tracks for developers

http://www.episerver.com/services/education/services-education-courses/developers/

CMS Development Fundamentals (3 days)
CMS Advanced Development (3 days)
CMS Developer Boot Camp (5 days)

CMS Masterclass + exam (2 days)

Commerce Development Fundamentals (3 days)
Commerce Advanced Development (3 days)
Commerce Developer Boot Camp (5 days)

Find for Developers (1 day)

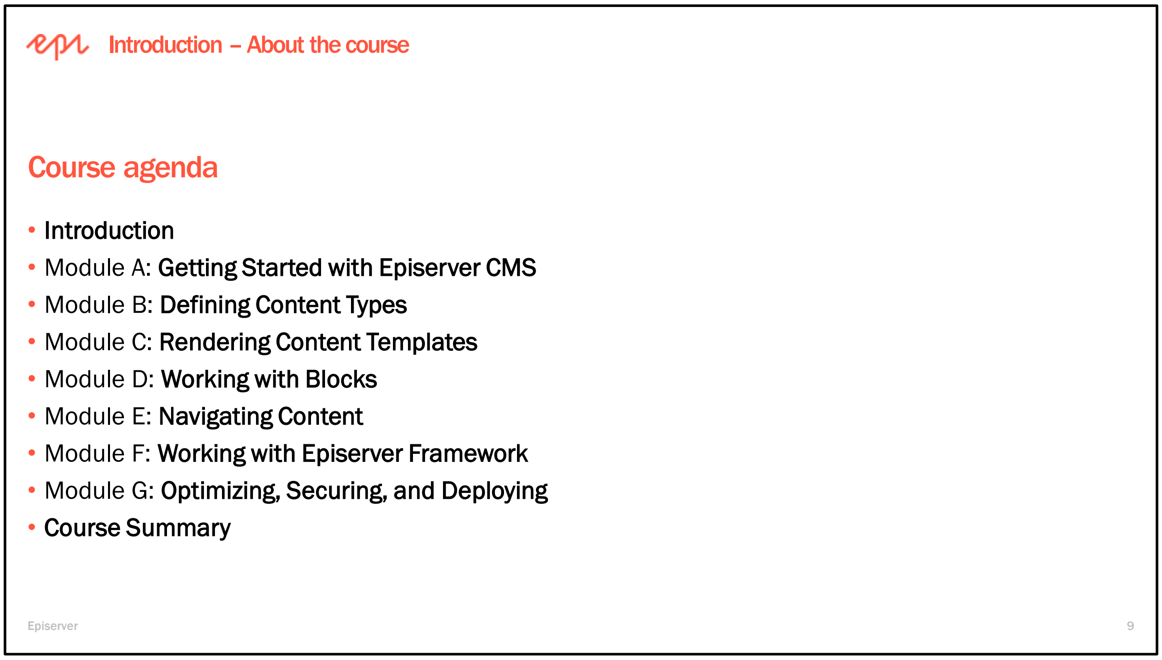
Developing for DXC Service (1 day)

CMS Development Fundamentals (3 days)
Commerce Development Fundamentals (3 days)
CMS & Commerce Fundamentals (5 days)

Episerver

8

Episerver CMS – Development Fundamentals
This step-by-step guide-style training course focuses on developing core functionality for Episerver CMS solutions, including defining custom content types and templates, handling media, reusing shared content, implementing navigation and indexed search, and optimizing, securing, and deploying websites, both on-premise and in the cloud.
Episerver CMS – Advanced Development
This cookbook-style training course is a deep dive into development with Episerver CMS with focus on reviewing the fundamentals and then customizing and extending the Episerver platform following recommended good practice. You will learn how to use APIs for taking control of content approvals, user notifications, and key performance indicators, you will integrate data with partial routers and a combination of scheduled jobs and system-level content events, and you will integrate Episerver Find and Episerver Social microservices to build advanced features into your websites.
Episerver CMS – Developer Boot Camp
Get up to speed with Episerver CMS development – fast! Join our Developer Boot Camp and get set to both take on real world projects with Episerver CMS and prepare to test your competence with the Episerver Certified Developer for Episerver CMS 11 exam. This course includes all the content from our *Development Fundamentals* and *Advanced Development* training courses in an accelerated format.
Episerver Commerce – Development Fundamentals
Learn how to work with the different parts of Episerver Commerce and after the course you should be able to build your first e-commerce solution from scratch.
Episerver Commerce – Advanced Development
Get the knowledge and understanding of how to work with our Commerce platform and API. The course highlights many abilities of the platform for being able to create a feature rich and automated website built with Episerver Commerce.
Developing for DXC Service
Learn how to successfully develop Episerver solutions for the cloud, taking into account development, deployment, and security considerations, so that you avoid the common traps and have a smooth productive experience with Episerver's DXC Service.
Episerver Find for Developers
Learn all about the magic behind Episerver Find. With this course you will get the skills necessary to build a powerful search function, including automatic landing pages and dynamic navigation.

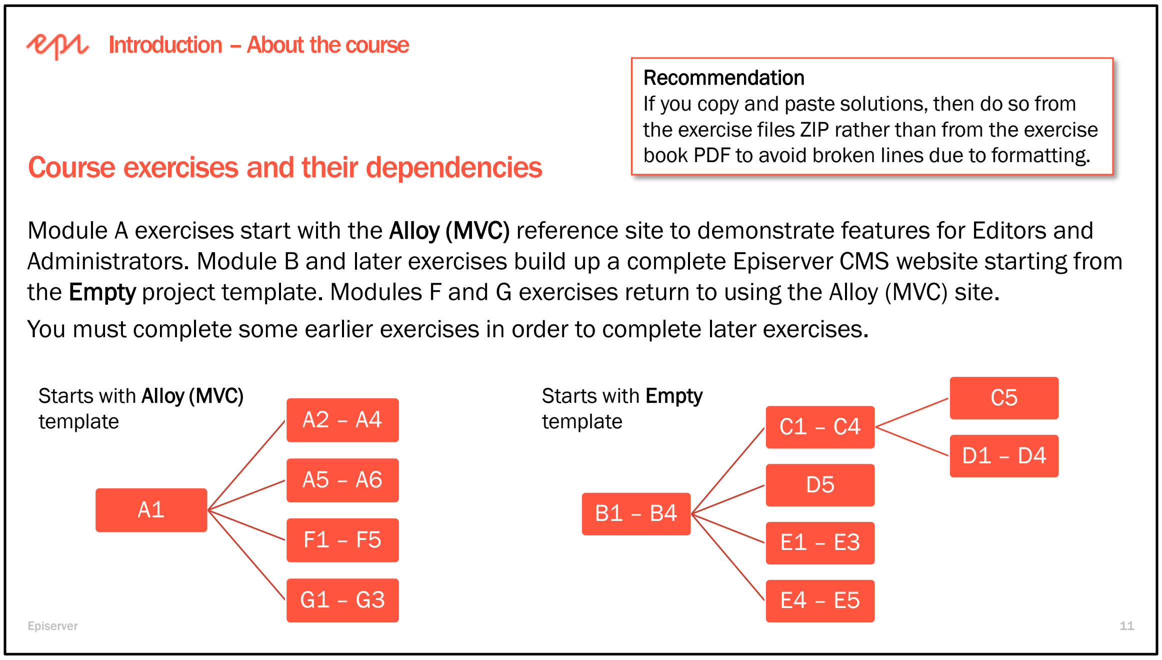

Introduction – About the course

Course agenda

- Introduction
- Module A: Getting Started with Episerver CMS
- Module B: Defining Content Types
- Module C: Rendering Content Templates
- Module D: Working with Blocks
- Module E: Navigating Content
- Module F: Working with Episerver Framework
- Module G: Optimizing, Securing, and Deploying
- Course Summary

Episerver 9

Module A: Getting Started with Episerver CMS
In this module, you will have a walkthrough of the Episerver user interface for Content Editors, Marketers, and Administrators, learn how to install the product and how to setup your development environment, and finally create a minimal Episerver website.
Module B: Defining Content Types
In this module, you will learn how to define content types with properties, and how to render them with content templates. You will learn about the important attributes that control how a content type and its properties are registered with Episerver CMS.
Module C: Rendering Content Templates
In this module, you will learn about content areas, display channels, display options, and tags for selecting between multiple templates for a content type.
Module D: Working with Blocks
In this module, you will learn about the two uses of blocks: as an item of shared content and as a property type.
Module E: Navigating Content
In this module, you will learn how to create content listings and menus using IContentLoader and common filters, and you will learn how to work with the built-in search for Episerver CMS.
Module F: Working with Episerver Framework
In this module, you will learn about the Episerver architecture and framework, know the various important classes and abstractions.
Module G: Optimizing, Securing, and Deploying
In this module, you will learn about deployment options and tools, and how to secure and optimize an Episerver website.



Introduction – About the course

Course exercises and their dependencies

Recommendation
If you copy and paste solutions, then do so from the exercise files ZIP rather than from the exercise book PDF to avoid broken lines due to formatting.

Module A exercises start with the **Alloy (MVC)** reference site to demonstrate features for Editors and Administrators. Module B and later exercises build up a complete Episerver CMS website starting from the **Empty** project template. Modules F and G exercises return to using the Alloy (MVC) site.

You must complete some earlier exercises in order to complete later exercises.

Starts with **Alloy (MVC)** template

A1 → A2 – A4, A5 – A6, F1 – F5, G1 – G3

Starts with **Empty** template

B1 – B4 → C1 – C4, D5, E1 – E3, E4 – E5

C1 – C4 → C5, D1 – D4

Episerver

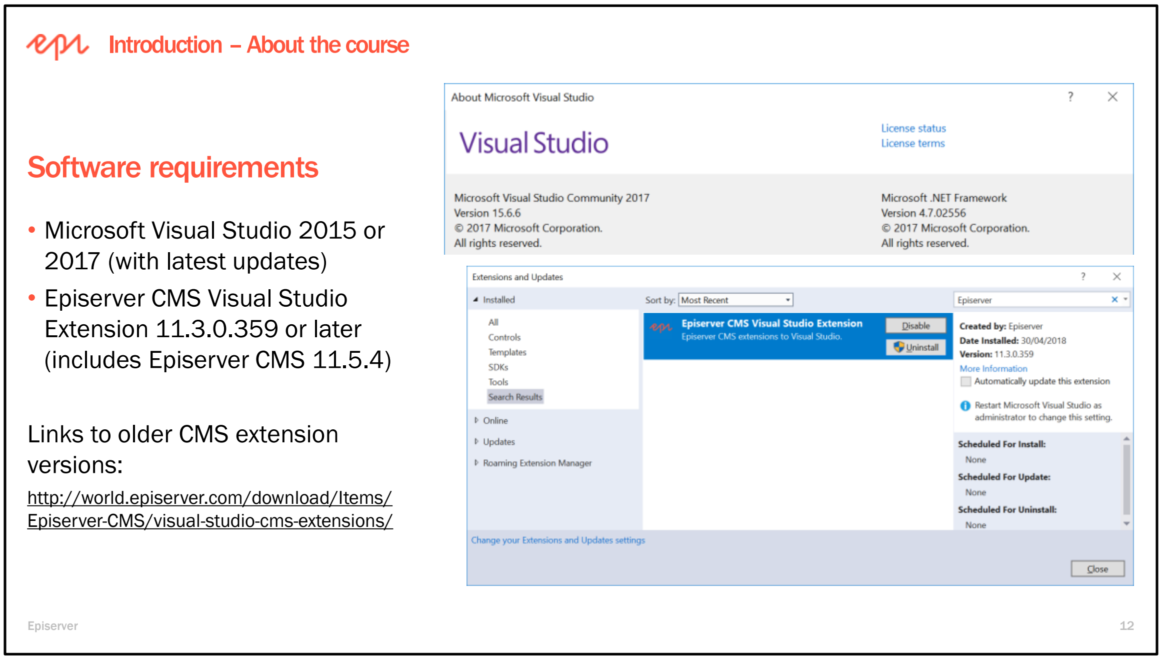
11

Modules A, F, and G use the **Alloy (MVC)** project template.

Alloy (MVC) is a website for a fictional company named Alloy that shows many Episerver CMS features and is implemented following Episerver good practices. But it is not designed as an example of a massively scalable website.

The Alloy (MVC) project template makes use of built-in functionality like categories, personalization, and blocks to illustrate some possibilities when implementing dynamic websites with Episerver CMS. Use Alloy (MVC) to inspire and guide you to success with your own custom websites.

Modules B to E use the **Empty** project template. This provides a minimum set up for an Episerver website, but does not include any content type or templates, so visitors will see a 404 Missing resource error, but CMS Editors and CMS Admins can manually enter `/EPiServer/CMS/` to log in to the Episerver CMS user interface.



Introduction – About the course

Software requirements

- Microsoft Visual Studio 2015 or 2017 (with latest updates)
- Episerver CMS Visual Studio Extension 11.3.0.359 or later (includes Episerver CMS 11.5.4)

Links to older CMS extension versions:
http://world.episerver.com/download/Items/Episerver-CMS/visual-studio-cms-extensions/

About Microsoft Visual Studio

Visual Studio

License status
License terms

Microsoft Visual Studio Community 2017
Version 15.6.6
© 2017 Microsoft Corporation.
All rights reserved.

Microsoft .NET Framework
Version 4.7.02556
© 2017 Microsoft Corporation.
All rights reserved.

Extensions and Updates

Installed: All, Controls, Templates, SDKs, Tools, Search Results; Online; Updates; Roaming Extension Manager

Sort by: Most Recent
Episerver

Episerver CMS Visual Studio Extension
Episerver CMS extensions to Visual Studio.

Disable
Uninstall

Created by: Episerver
Date Installed: 30/04/2018
Version: 11.3.0.359
More Information
Automatically update this extension
Restart Microsoft Visual Studio as administrator to change this setting.

Scheduled For Install:
None
Scheduled For Update:
None
Scheduled For Uninstall:
None

Change your Extensions and Updates settings

Close

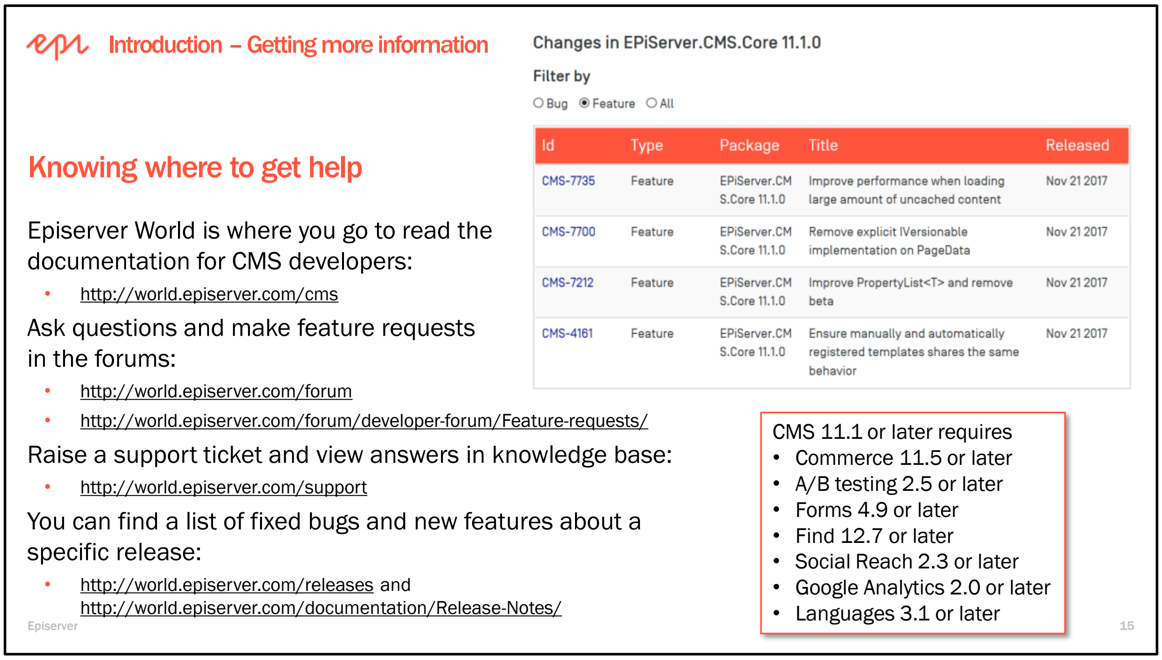
Episerver 12

If you are using an Episerver virtual machine, it will have:

- Microsoft Windows 10 Enterprise with IIS
- Microsoft Visual Studio 2017
- Microsoft SQL Server Management Studio and SQL Server LocalDb
- Microsoft Azure SDK
- Microsoft Edge, Internet Explorer 11, Firefox, Chrome

If you would like to use your own PC, use the following guide, available as a PDF:

- *Setting Up Episerver Sites for Training*



Introduction – Getting more information

Knowing where to get help

Episerver World is where you go to read the documentation for CMS developers:

- http://world.episerver.com/cms

Ask questions and make feature requests in the forums:

- http://world.episerver.com/forum
- http://world.episerver.com/forum/developer-forum/Feature-requests/

Raise a support ticket and view answers in knowledge base:

- http://world.episerver.com/support

You can find a list of fixed bugs and new features about a specific release:

- http://world.episerver.com/releases and http://world.episerver.com/documentation/Release-Notes/

Changes in EPiServer.CMS.Core 11.1.0

Filter by

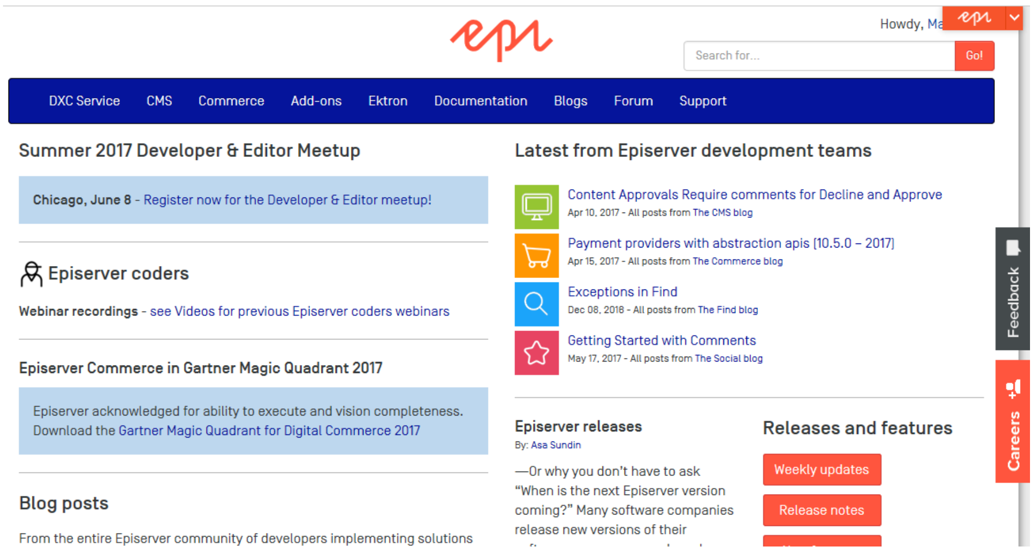
Bug [✓ Feature] All

Id	Type	Package	Title	Released
CMS-7735	Feature	EPiServer.CMS.Core 11.1.0	Improve performance when loading large amount of uncached content	Nov 21 2017
CMS-7700	Feature	EPiServer.CMS.Core 11.1.0	Remove explicit IVersionable implementation on PageData	Nov 21 2017
CMS-7212	Feature	EPiServer.CMS.Core 11.1.0	Improve PropertyList<T> and remove beta	Nov 21 2017
CMS-4161	Feature	EPiServer.CMS.Core 11.1.0	Ensure manually and automatically registered templates shares the same behavior	Nov 21 2017

CMS 11.1 or later requires

- Commerce 11.5 or later
- A/B testing 2.5 or later
- Forms 4.9 or later
- Find 12.7 or later
- Social Reach 2.3 or later
- Google Analytics 2.0 or later
- Languages 3.1 or later

Episerver 15



Summer 2017 Developer & Editor Meetup

Chicago, June 8 - Register now for the Developer & Editor meetup!

Episerver coders

Webinar recordings - see Videos for previous Episerver coders webinars

Episerver Commerce in Gartner Magic Quadrant 2017

Episerver acknowledged for ability to execute and vision completeness. Download the Gartner Magic Quadrant for Digital Commerce 2017

Blog posts

From the entire Episerver community of developers implementing solutions

Latest from Episerver development teams

Content Approvals Require comments for Decline and Approve
Apr 10, 2017 - All posts from The CMS blog

Payment providers with abstraction apis [10.5.0 – 2017]
Apr 15, 2017 - All posts from The Commerce blog

Exceptions in Find
Dec 08, 2016 - All posts from The Find blog

Getting Started with Comments
May 17, 2017 - All posts from The Social blog

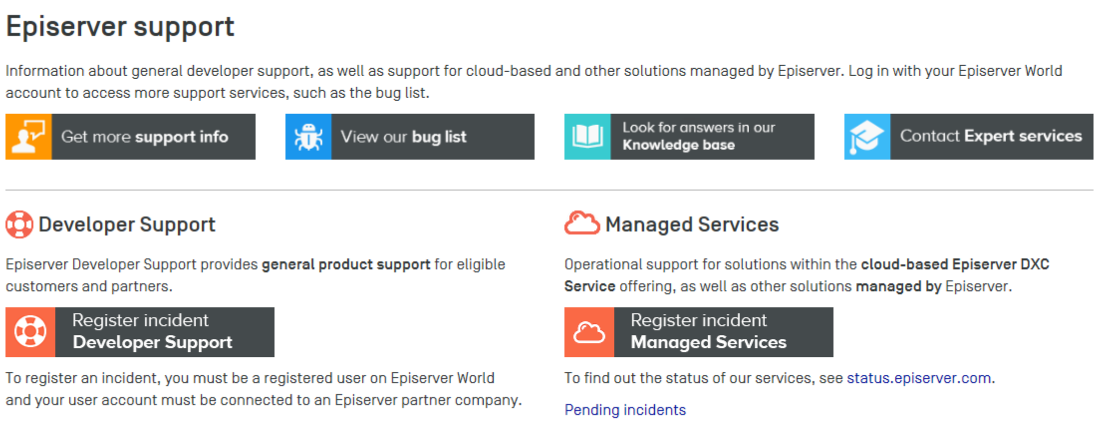
Episerver releases
By: Asa Sundin

—Or why you don't have to ask "When is the next Episerver version coming?" Many software companies release new versions of their

Releases and features

Weekly updates

Release notes



Episerver support

Information about general developer support, as well as support for cloud-based and other solutions managed by Episerver. Log in with your Episerver World account to access more support services, such as the bug list.

Get more support info
View our bug list
Look for answers in our Knowledge base
Contact Expert services

Developer Support

Episerver Developer Support provides **general product support** for eligible customers and partners.

Register incident Developer Support

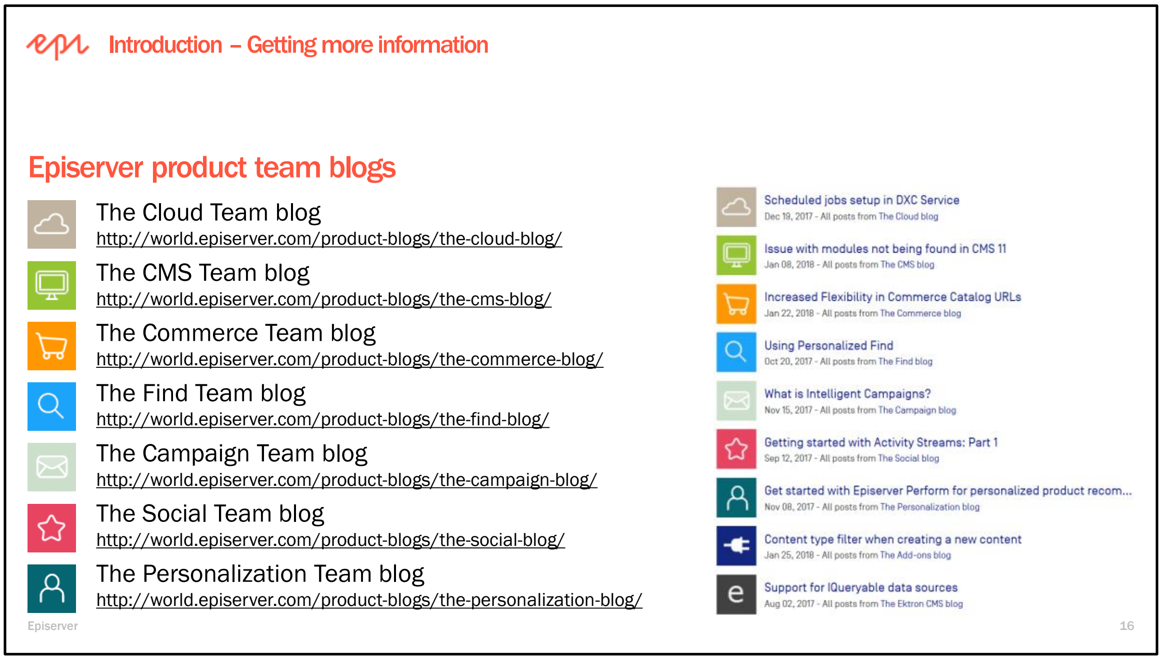
To register an incident, you must be a registered user on Episerver World and your user account must be connected to an Episerver partner company.

Managed Services

Operational support for solutions within the **cloud-based Episerver DXC Service** offering, as well as other solutions **managed by** Episerver.

To find out the status of our services, see status.episerver.com.

Pending incidents



Introduction – Getting more information

Episerver product team blogs

The Cloud Team blog
http://world.episerver.com/product-blogs/the-cloud-blog/

The CMS Team blog
http://world.episerver.com/product-blogs/the-cms-blog/

The Commerce Team blog
http://world.episerver.com/product-blogs/the-commerce-blog/

The Find Team blog
http://world.episerver.com/product-blogs/the-find-blog/

The Campaign Team blog
http://world.episerver.com/product-blogs/the-campaign-blog/

The Social Team blog
http://world.episerver.com/product-blogs/the-social-blog/

The Personalization Team blog
http://world.episerver.com/product-blogs/the-personalization-blog/

Scheduled jobs setup in DXC Service
Dec 19, 2017 - All posts from The Cloud blog

Issue with modules not being found in CMS 11
Jan 08, 2018 - All posts from The CMS blog

Increased Flexibility in Commerce Catalog URLs
Jan 22, 2018 - All posts from The Commerce blog

Using Personalized Find
Oct 20, 2017 - All posts from The Find blog

What is Intelligent Campaigns?
Nov 15, 2017 - All posts from The Campaign blog

Getting started with Activity Streams: Part 1
Sep 12, 2017 - All posts from The Social blog

Get started with Episerver Perform for personalized product recom...
Nov 08, 2017 - All posts from The Personalization blog

Content type filter when creating a new content
Jan 25, 2018 - All posts from The Add-ons blog

Support for IQueryable data sources
Aug 02, 2017 - All posts from The Ektron CMS blog

Episerver 16

Issue with modules not being found in CMS 11

Posted: Jan 08, 2018

Hello everyone! I want to quickly address the issues around add-on modules not being found after upgrading to CMS 11 before misinformation starts to spread. Marija mentioned some of these issues in her previous blog post http://mariajemaria.net/u... 4

By: Ben McKernan
Views: 774
Rating: ★★★★★

Pin the top menu in the Episerver UI

Posted: Dec 19, 2017

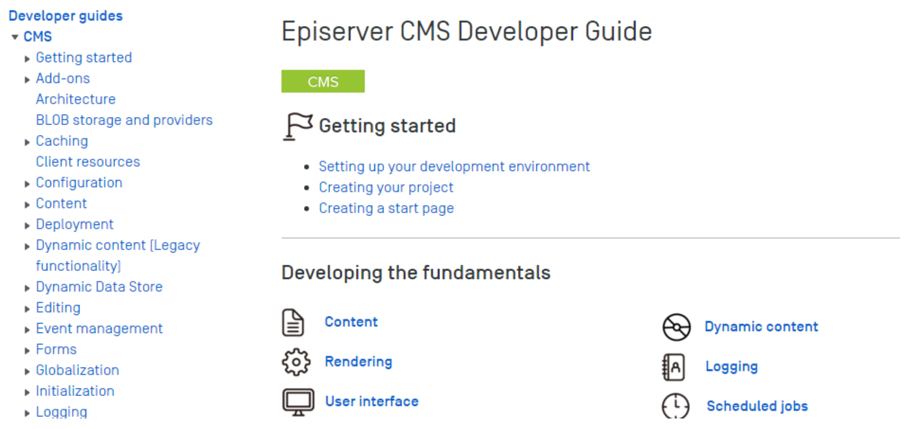
The Episerver UI now has the ability to pin the menu at the top of the screen as a core feature. Its possible to do this by clicking the icon once the menu is opened as shown below: Your browser does not support HTML5 video. This feature is... 5

By: David Knipe
Views: 875
Rating: ★★★★★

Introduction – Getting more information

CMS documentation

http://world.episerver.com/documentation/developer-guides/CMS/



Developer guides
- CMS
  - Getting started
  - Add-ons
  - Architecture
  - BLOB storage and providers
  - Caching
  - Client resources
  - Configuration
  - Content
  - Deployment
  - Dynamic content (Legacy functionality)
  - Dynamic Data Store
  - Editing
  - Event management
  - Forms
  - Globalization
  - Initialization
  - Logging

Episerver CMS Developer Guide

CMS

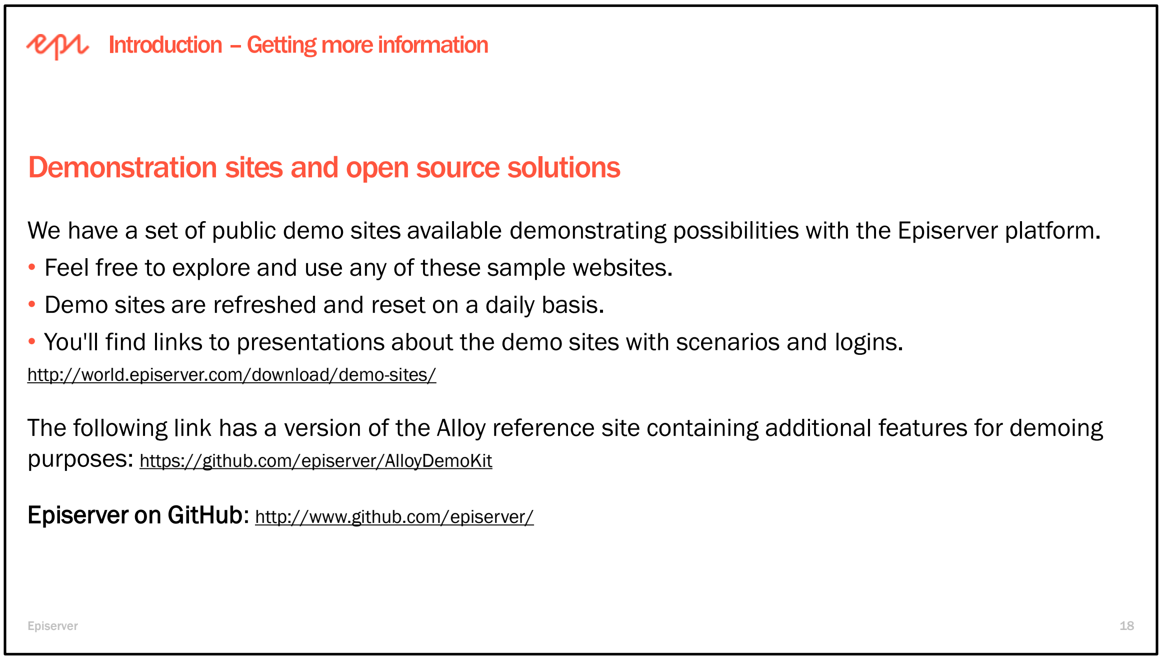
Getting started

- Setting up your development environment
- Creating your project
- Creating a start page

Developing the fundamentals

Content	Dynamic content
Rendering	Logging
User interface	Scheduled jobs

Episerver 17



Introduction – Getting more information

Demonstration sites and open source solutions

We have a set of public demo sites available demonstrating possibilities with the Episerver platform.

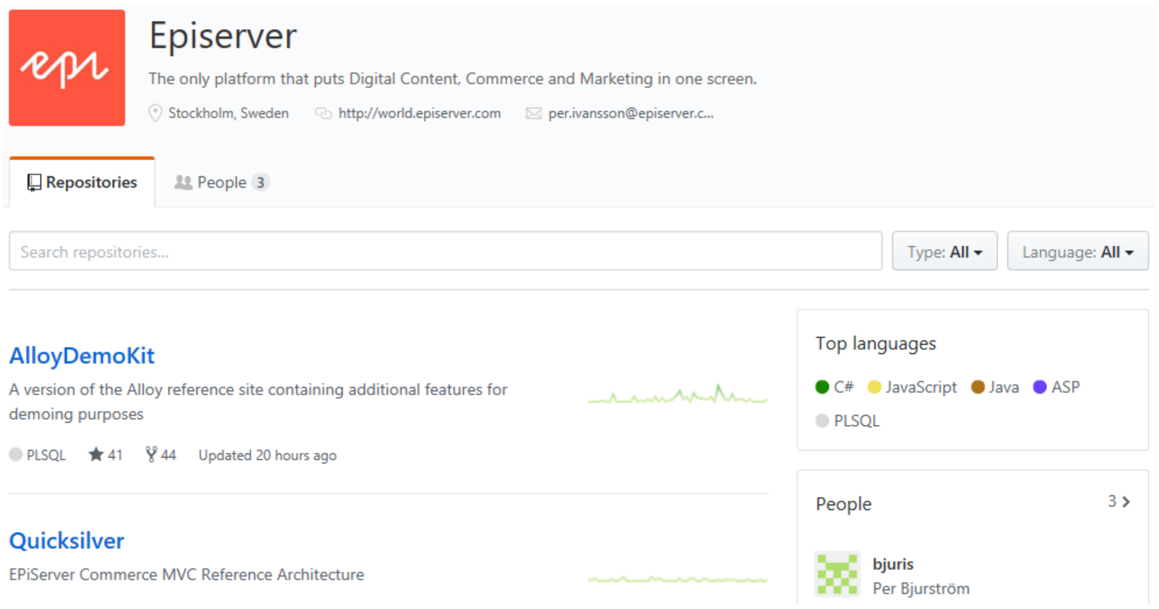
- Feel free to explore and use any of these sample websites.
- Demo sites are refreshed and reset on a daily basis.
- You'll find links to presentations about the demo sites with scenarios and logins.

http://world.episerver.com/download/demo-sites/

The following link has a version of the Alloy reference site containing additional features for demoing purposes: https://github.com/episerver/AlloyDemoKit

Episerver on GitHub: http://www.github.com/episerver/

Episerver 18



Episerver

The only platform that puts Digital Content, Commerce and Marketing in one screen.

Stockholm, Sweden http://world.episerver.com per.ivansson@episerver.c...

AlloyDemoKit

A version of the Alloy reference site containing additional features for demoing purposes

PLSQL ★ 41 44 Updated 20 hours ago

Quicksilver

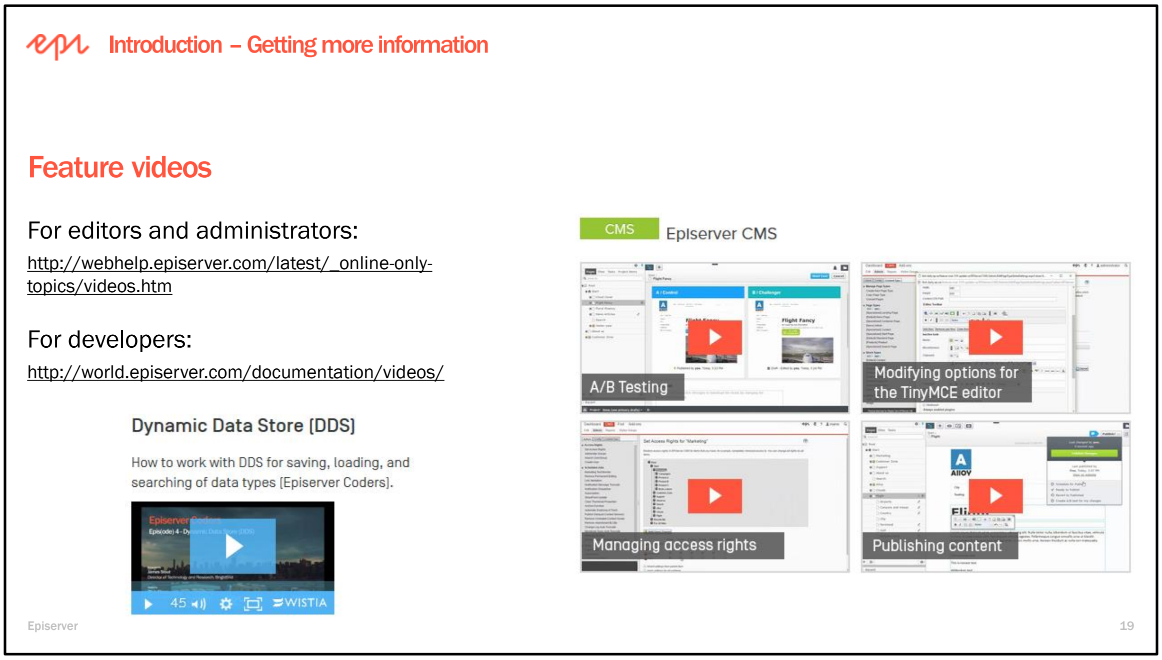
EPiServer Commerce MVC Reference Architecture

Top languages

● C# ● JavaScript ● Java ● ASP ● PLSQL

People 3

bjuris
Per Bjurström



Introduction – Getting more information

Feature videos

For editors and administrators:
http://webhelp.episerver.com/latest/_online-only-topics/videos.htm

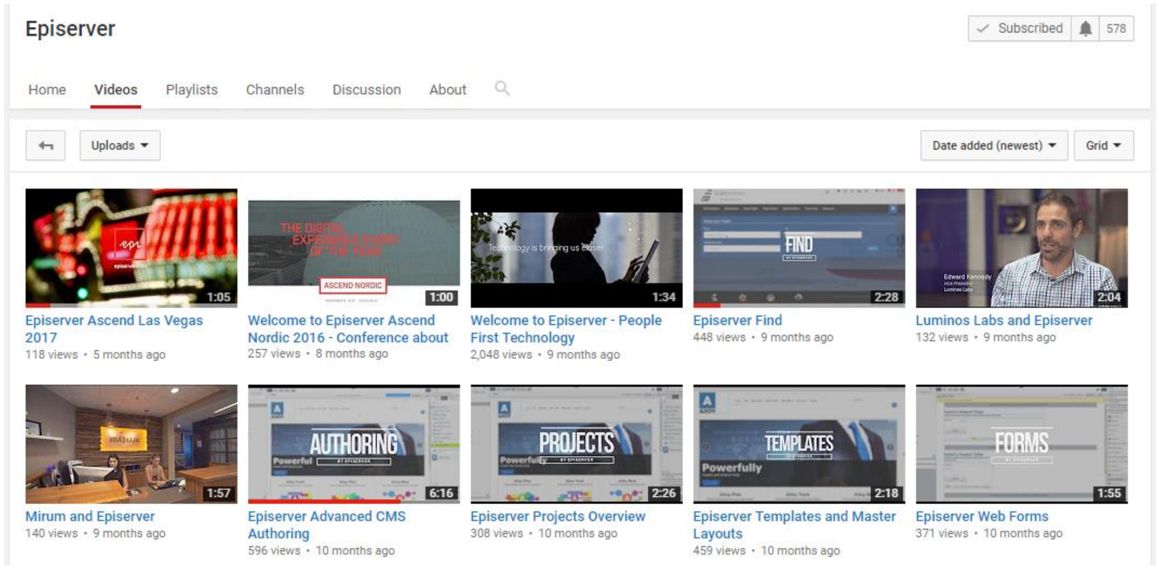
For developers:
http://world.episerver.com/documentation/videos/

Dynamic Data Store [DDS]

How to work with DDS for saving, loading, and searching of data types [Episerver Coders].

Episerver 19

Episerver on YouTube
https://www.youtube.com/user/EpiserverAB



Episerver Ascend Las Vegas 2017 — 1:05 — 118 views • 5 months ago

Welcome to Episerver Ascend Nordic 2016 - Conference about — 1:00 — 257 views • 8 months ago

Welcome to Episerver - People First Technology — 1:34 — 2,048 views • 9 months ago

Episerver Find — 2:28 — 448 views • 9 months ago

Luminos Labs and Episerver — 2:04 — 132 views • 9 months ago

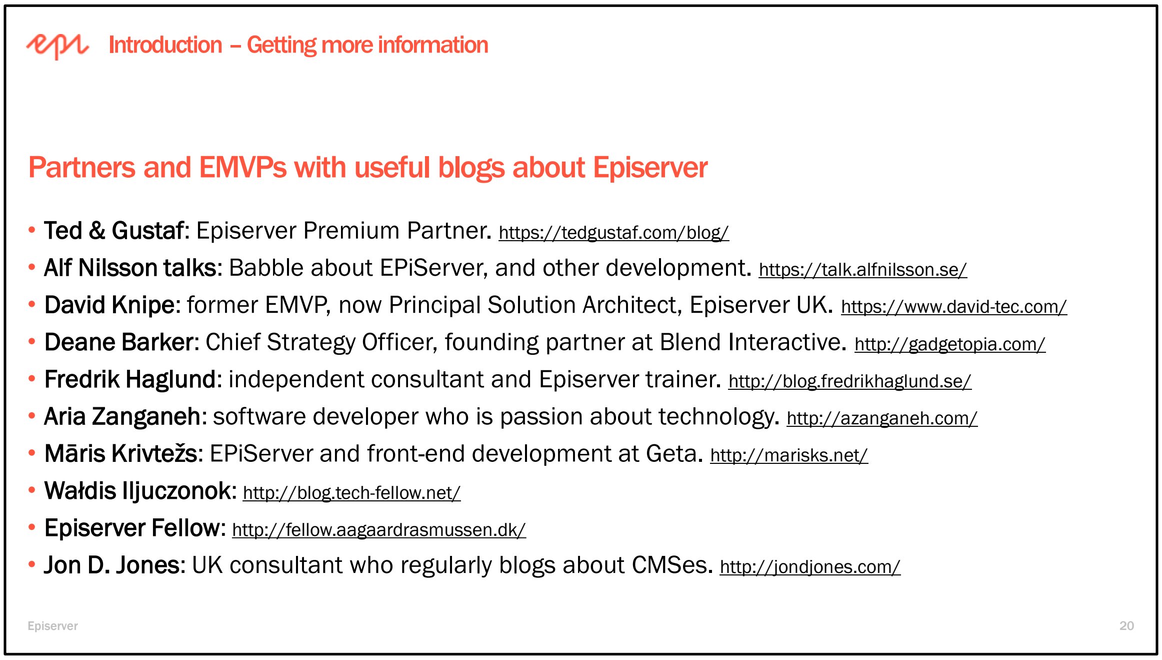
Mirum and Episerver — 1:57 — 140 views • 9 months ago

Episerver Advanced CMS Authoring — 6:16 — 596 views • 10 months ago

Episerver Projects Overview — 2:26 — 308 views • 10 months ago

Episerver Templates and Master Layouts — 2:18 — 459 views • 10 months ago

Episerver Web Forms — 1:55 — 371 views • 10 months ago



Introduction – Getting more information

Partners and EMVPs with useful blogs about Episerver

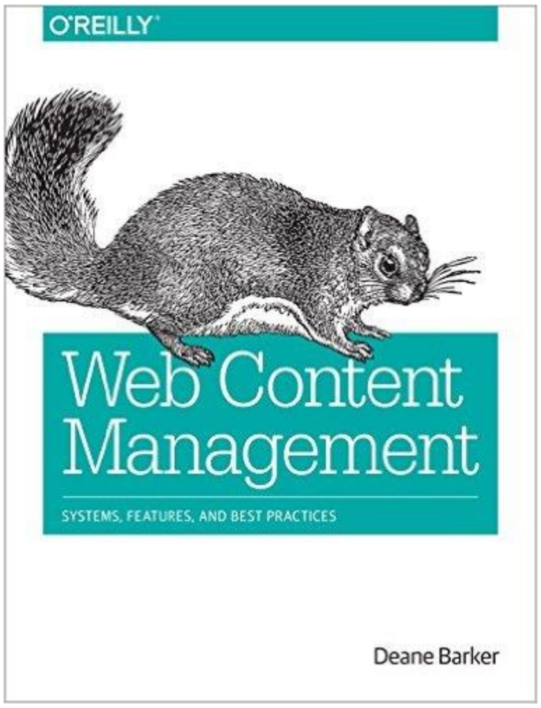
- **Ted & Gustaf**: Episerver Premium Partner. https://tedgustaf.com/blog/
- **Alf Nilsson talks**: Babble about EPiServer, and other development. https://talk.alfnilsson.se/
- **David Knipe**: former EMVP, now Principal Solution Architect, Episerver UK. https://www.david-tec.com/
- **Deane Barker**: Chief Strategy Officer, founding partner at Blend Interactive. http://gadgetopia.com/
- **Fredrik Haglund**: independent consultant and Episerver trainer. http://blog.fredrikhaglund.se/
- **Aria Zanganeh**: software developer who is passion about technology. http://azanganeh.com/
- **Māris Krivtežs**: EPiServer and front-end development at Geta. http://marisks.net/
- **Wałdis Iljuczonok**: http://blog.tech-fellow.net/
- **Episerver Fellow**: http://fellow.aagaardrasmussen.dk/
- **Jon D. Jones**: UK consultant who regularly blogs about CMSes. http://jondjones.com/

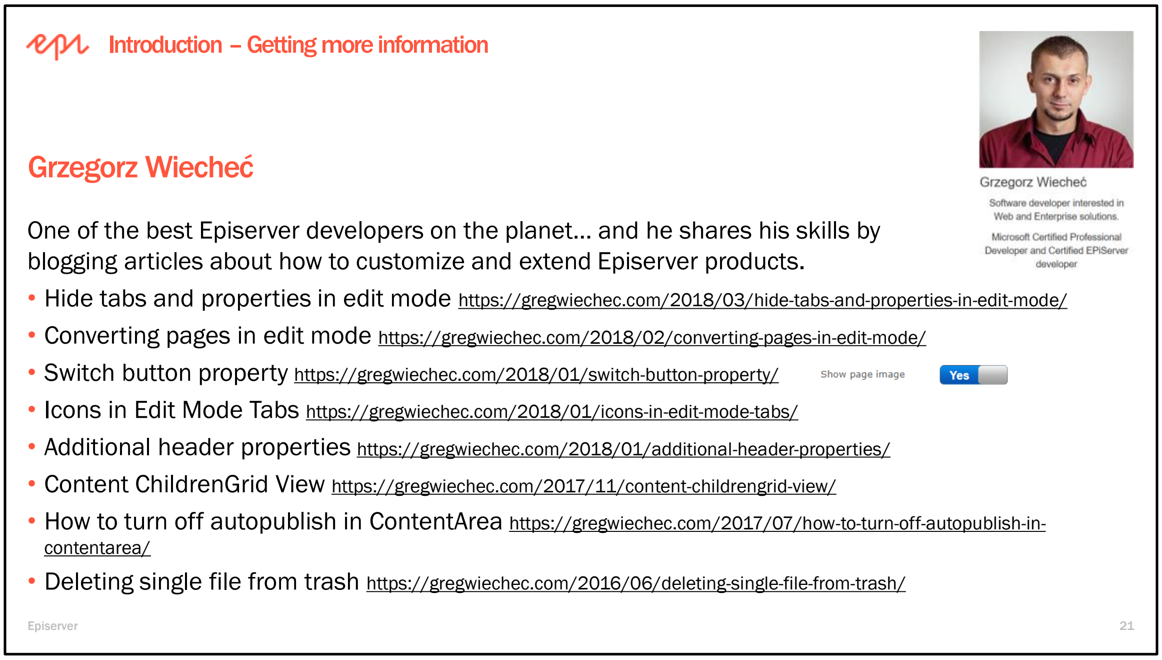
Episerver 20

If you do not have a background with Web Content Management Systems (WCM/CMS), then we recommend this book. It is not specific to Episerver.

https://www.amazon.co.uk/Web-Content-Management-Features-Practices/dp/1491908122/

Deane Barker is an Episerver Most Valued Professional (EMVP): http://world.episerver.com/emvp/





Introduction – Getting more information

Grzegorz Wiecheć

Software developer interested in Web and Enterprise solutions.

Microsoft Certified Professional Developer and Certified EPiServer developer

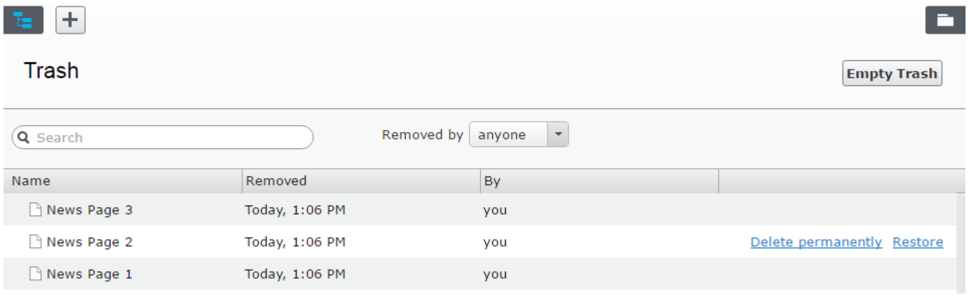
Grzegorz Wiecheć

One of the best Episerver developers on the planet… and he shares his skills by blogging articles about how to customize and extend Episerver products.

- Hide tabs and properties in edit mode https://gregwiechec.com/2018/03/hide-tabs-and-properties-in-edit-mode/
- Converting pages in edit mode https://gregwiechec.com/2018/02/converting-pages-in-edit-mode/
- Switch button property https://gregwiechec.com/2018/01/switch-button-property/ Show page image Yes
- Icons in Edit Mode Tabs https://gregwiechec.com/2018/01/icons-in-edit-mode-tabs/
- Additional header properties https://gregwiechec.com/2018/01/additional-header-properties/
- Content ChildrenGrid View https://gregwiechec.com/2017/11/content-childrengrid-view/
- How to turn off autopublish in ContentArea https://gregwiechec.com/2017/07/how-to-turn-off-autopublish-in-contentarea/
- Deleting single file from trash https://gregwiechec.com/2016/06/deleting-single-file-from-trash/

Episerver 21

Deleting single file from trash



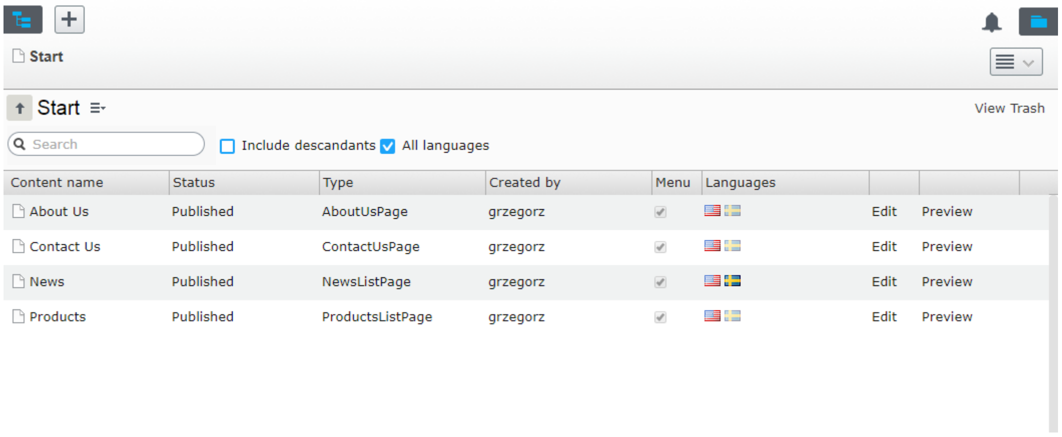
Trash

Empty Trash

Search Removed by anyone

Name	Removed	By	
News Page 3	Today, 1:06 PM	you	
News Page 2	Today, 1:06 PM	you	Delete permanently Restore
News Page 1	Today, 1:06 PM	you	

Content ChildrenGrid View



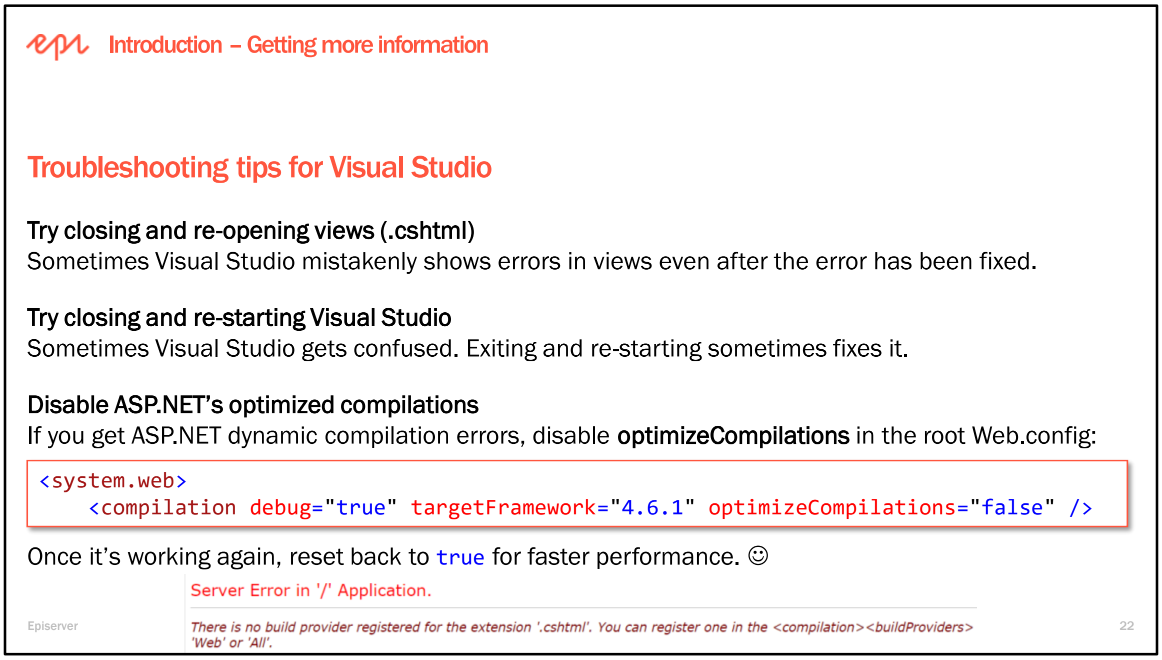
Start

Start

View Trash

Search ☐ Include descandants ☑ All languages

Content name	Status	Type	Created by	Menu	Languages		
About Us	Published	AboutUsPage	grzegorz	✓		Edit	Preview
Contact Us	Published	ContactUsPage	grzegorz	✓		Edit	Preview
News	Published	NewsListPage	grzegorz	✓		Edit	Preview
Products	Published	ProductsListPage	grzegorz	✓		Edit	Preview



Introduction – Getting more information

Troubleshooting tips for Visual Studio

Try closing and re-opening views (.cshtml)
Sometimes Visual Studio mistakenly shows errors in views even after the error has been fixed.

Try closing and re-starting Visual Studio
Sometimes Visual Studio gets confused. Exiting and re-starting sometimes fixes it.

Disable ASP.NET's optimized compilations
If you get ASP.NET dynamic compilation errors, disable **optimizeCompilations** in the root Web.config:

```
<system.web>
    <compilation debug="true" targetFramework="4.6.1" optimizeCompilations="false" />
```

Once it's working again, reset back to `true` for faster performance. ☺

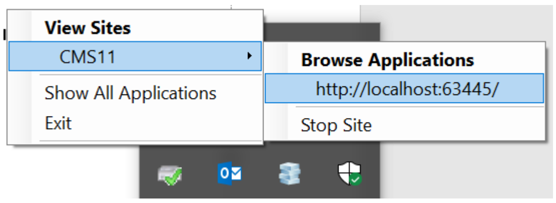
Server Error in '/' Application.

There is no build provider registered for the extension '.cshtml'. You can register one in the <compilation><buildProviders> 'Web' or 'All'.

Episerver 22

Disable Visual Studio's Browser Link
If the browser seems to hang while drawing the Episerver UI, it may be Visual Studio's Browser Link feature interfering with our Dojo library: disable Browser Link to prevent the JavaScript error.

Reset IIS or IIS Express
Use the `iisreset` command line to stop and restart IIS or use the Taskbar tray icon for IIS Express, as shown in the following screenshot:

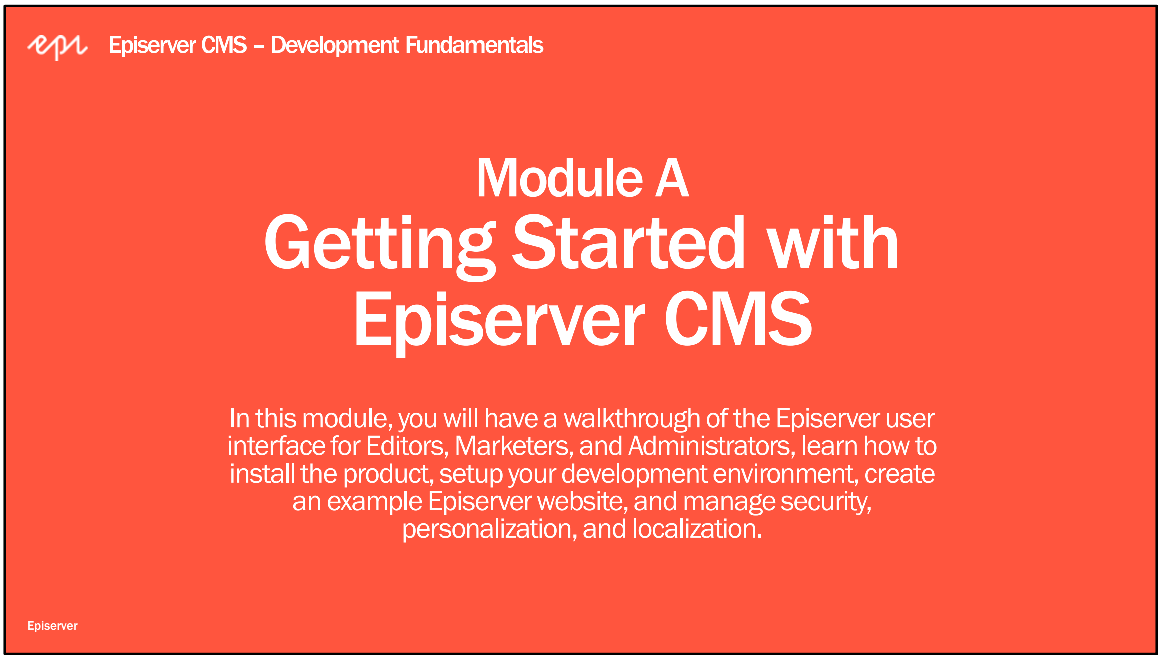


Empty ASP.NET Temporary Files folder
By default, the dynamic compilation of views stores the assemblies (*.dll) in a temporary folder. Stop the site, shut down Visual Studio, and clear the folder sometimes fixes issues.

Disable Web Sockets on Windows 7 (to hide error messages about no real-time communication)

```
<add key="Epi.WebSockets.Enabled" value="false" />
```

http://world.episerver.com/documentation/developer-guides/CMS/user-interface/websocket-support/

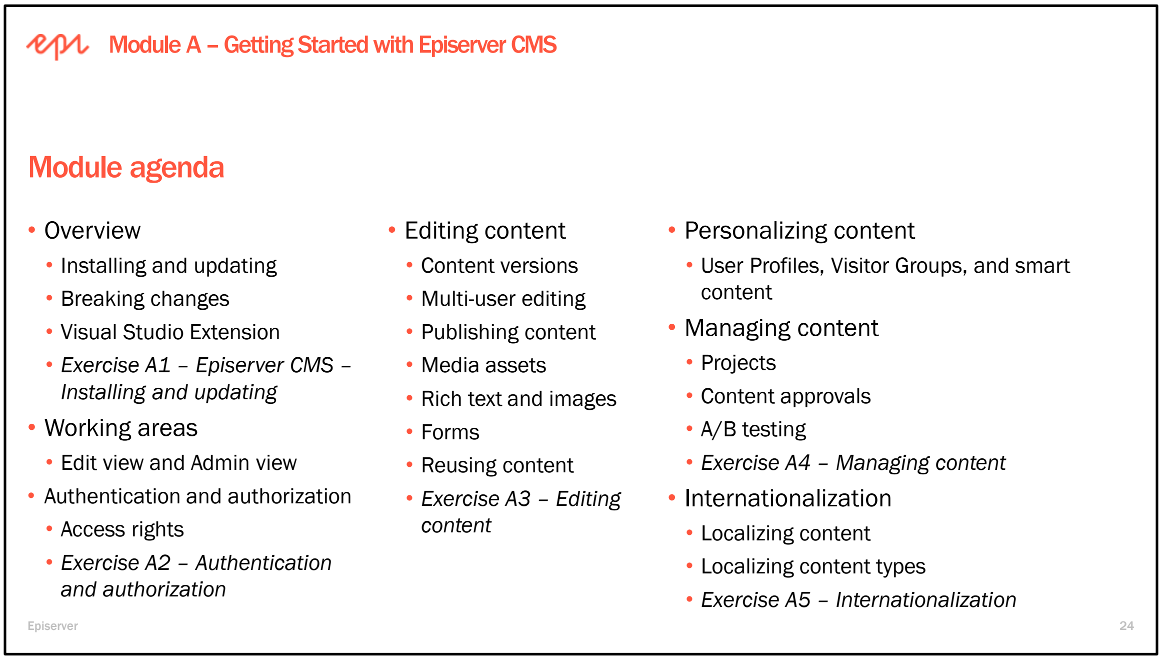


Episerver CMS – Development Fundamentals

Module A
Getting Started with Episerver CMS

In this module, you will have a walkthrough of the Episerver user interface for Editors, Marketers, and Administrators, learn how to install the product, setup your development environment, create an example Episerver website, and manage security, personalization, and localization.

Episerver

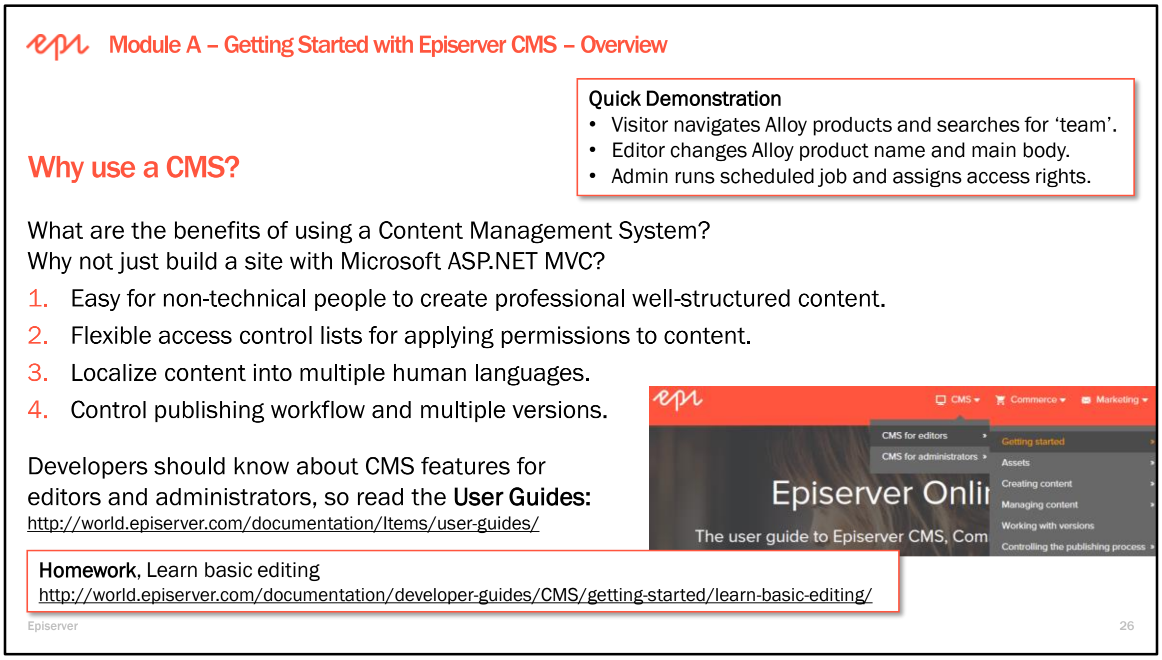


Module A – Getting Started with Episerver CMS

Module agenda

- Overview
  - Installing and updating
  - Breaking changes
  - Visual Studio Extension
  - *Exercise A1 – Episerver CMS – Installing and updating*
- Working areas
  - Edit view and Admin view
- Authentication and authorization
  - Access rights
  - *Exercise A2 – Authentication and authorization*
- Editing content
  - Content versions
  - Multi-user editing
  - Publishing content
  - Media assets
  - Rich text and images
  - Forms
  - Reusing content
  - *Exercise A3 – Editing content*
- Personalizing content
  - User Profiles, Visitor Groups, and smart content
- Managing content
  - Projects
  - Content approvals
  - A/B testing
  - *Exercise A4 – Managing content*
- Internationalization
  - Localizing content
  - Localizing content types
  - *Exercise A5 – Internationalization*

Episerver 24



Module A – Getting Started with Episerver CMS – Overview

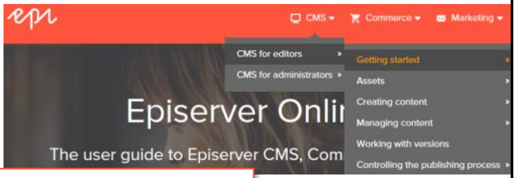
Quick Demonstration
- Visitor navigates Alloy products and searches for 'team'.
- Editor changes Alloy product name and main body.
- Admin runs scheduled job and assigns access rights.

Why use a CMS?

What are the benefits of using a Content Management System?
Why not just build a site with Microsoft ASP.NET MVC?

1. Easy for non-technical people to create professional well-structured content.
2. Flexible access control lists for applying permissions to content.
3. Localize content into multiple human languages.
4. Control publishing workflow and multiple versions.

Developers should know about CMS features for editors and administrators, so read the **User Guides:**
http://world.episerver.com/documentation/Items/user-guides/

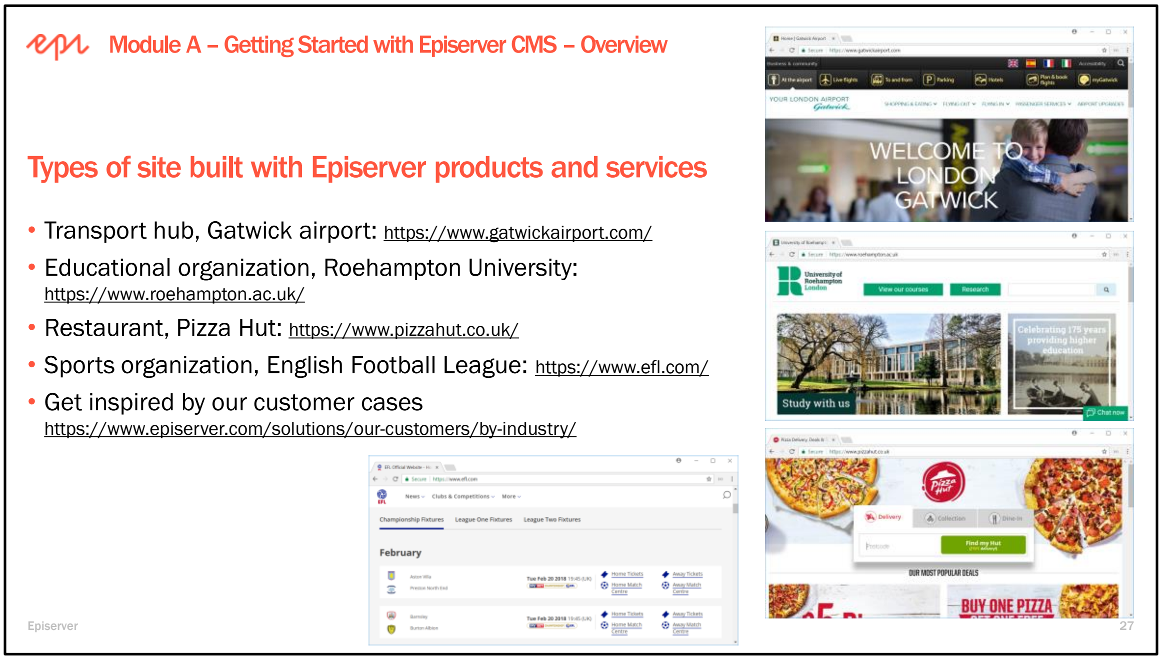


Homework, Learn basic editing
http://world.episerver.com/documentation/developer-guides/CMS/getting-started/learn-basic-editing/

Episerver 26

Links to **Developer Guide** and **User Guide** topics for Module A:

- Installing and updating: http://world.episerver.com/documentation/developer-guides/CMS/getting-started/
- Visual Studio Extension: http://world.episerver.com/documentation/Items/Installation-Instructions/installing-episerver/
- Working areas: http://webhelp.episerver.com/latest/getting-started/user-interface.htm
- Authentication and authorization: Access rights: http://webhelp.episerver.com/latest/cms-admin/access-rights.htm
- Editing content: Media assets: http://webhelp.episerver.com/latest/platform/media.htm
- Editing content: Rich text and images: http://webhelp.episerver.com/latest/cms-edit/editing-content.htm
- Editing content: Forms: http://webhelp.episerver.com/latest/cms-edit/working-with-web-forms.htm
- Editing content: Personalizing content: http://webhelp.episerver.com/latest/cms-edit/personalizing-content.htm
- Editing content: Projects: http://webhelp.episerver.com/latest/cms-edit/projects.htm
- Editing content: Content approvals: http://webhelp.episerver.com/latest/cms-edit/content-approvals.htm
- Editing content: A/B testing: http://webhelp.episerver.com/latest/cms-edit/ab-testing.htm
- Internationalization: http://webhelp.episerver.com/latest/platform/working-with-multiple-languages.htm



Module A – Getting Started with Episerver CMS – Overview

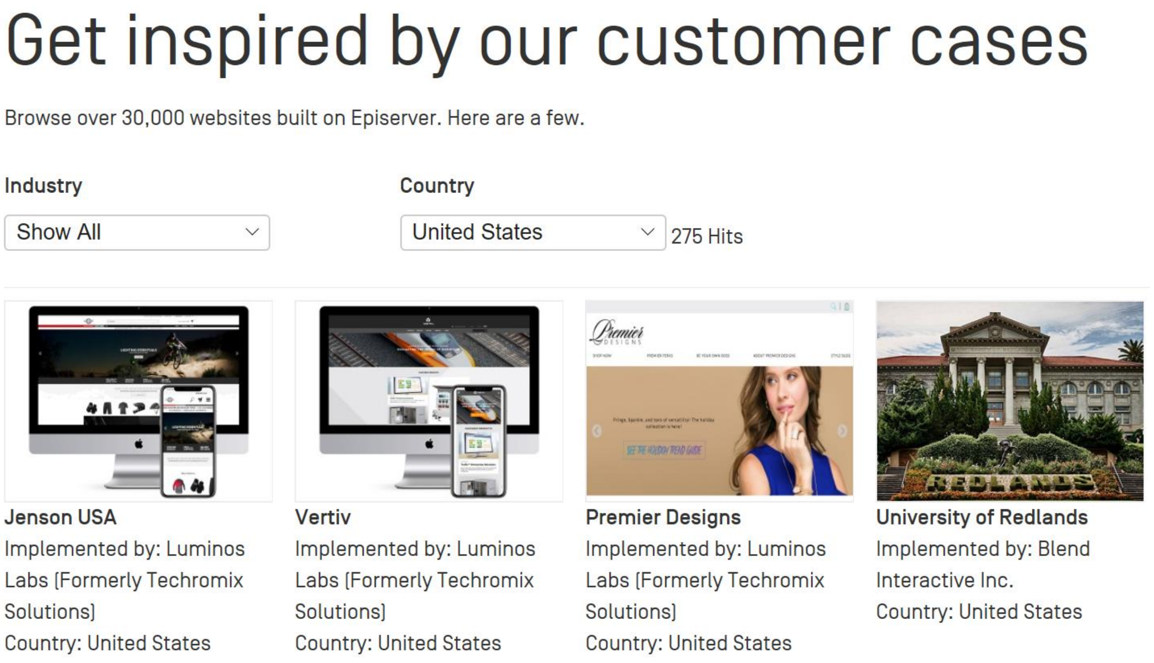
Types of site built with Episerver products and services

- Transport hub, Gatwick airport: https://www.gatwickairport.com/
- Educational organization, Roehampton University: https://www.roehampton.ac.uk/
- Restaurant, Pizza Hut: https://www.pizzahut.co.uk/
- Sports organization, English Football League: https://www.efl.com/
- Get inspired by our customer cases https://www.episerver.com/solutions/our-customers/by-industry/

Episerver

27

https://www.episerver.com/solutions/our-customers/by-industry/



Get inspired by our customer cases

Browse over 30,000 websites built on Episerver. Here are a few.

Industry: Show All

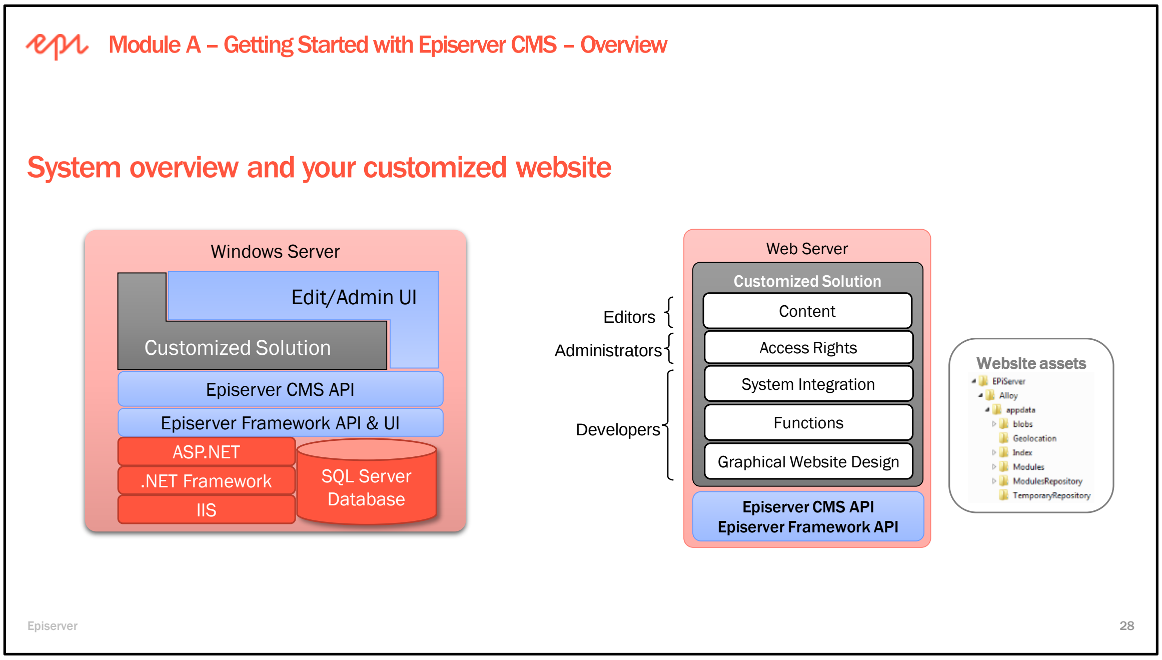
Country: United States 275 Hits

Jenson USA
Implemented by: Luminos Labs (Formerly Techromix Solutions)
Country: United States

Vertiv
Implemented by: Luminos Labs (Formerly Techromix Solutions)
Country: United States

Premier Designs
Implemented by: Luminos Labs (Formerly Techromix Solutions)
Country: United States

University of Redlands
Implemented by: Blend Interactive Inc.
Country: United States



Module A – Getting Started with Episerver CMS – Overview

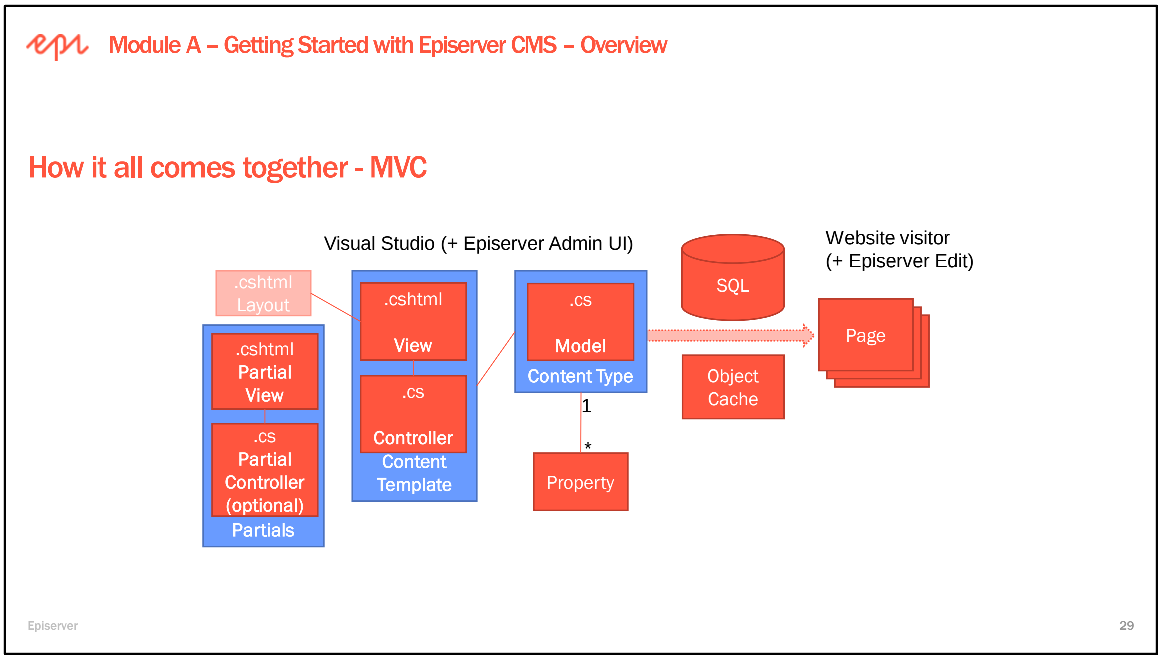
System overview and your customized website

Episerver

28

Episerver Framework and Episerver CMS APIs are what supports version management, preview, workflow, access rights, etc. These are functions that enable you to work in Edit mode as an editor. It is the platform that is available in different versions and developed continuously.
On top of the platform is the customized solution, which makes the website different from other websites. Certified partners create the customized solution in cooperation with the customer. In slightly simplified terms, the customized part can be divided into the following parts:

1. The access rights for editors and visitors to different pages.
2. The content on the website, which is stored in a database. Any images and documents are stored outside the actual Episerver CMS on a suitable data source.
3. The company's branding is saved in a format template (CSS). This contains the predefined fonts, colors, etc. that are to be used on the website.
4. A number of different functions that visitors can use on the website, for example, participating in a discussion forum, sending an e-mail with a link to a page or printing a page. Each function is normally linked to an individual page template.
5. If applicable, integration with other systems. For instance an e-commerce or a community module, with a wide variety of integration methods available.



Module A – Getting Started with Episerver CMS – Overview

How it all comes together - MVC

Episerver 29

Terminology

A **content type** defines a set of properties.

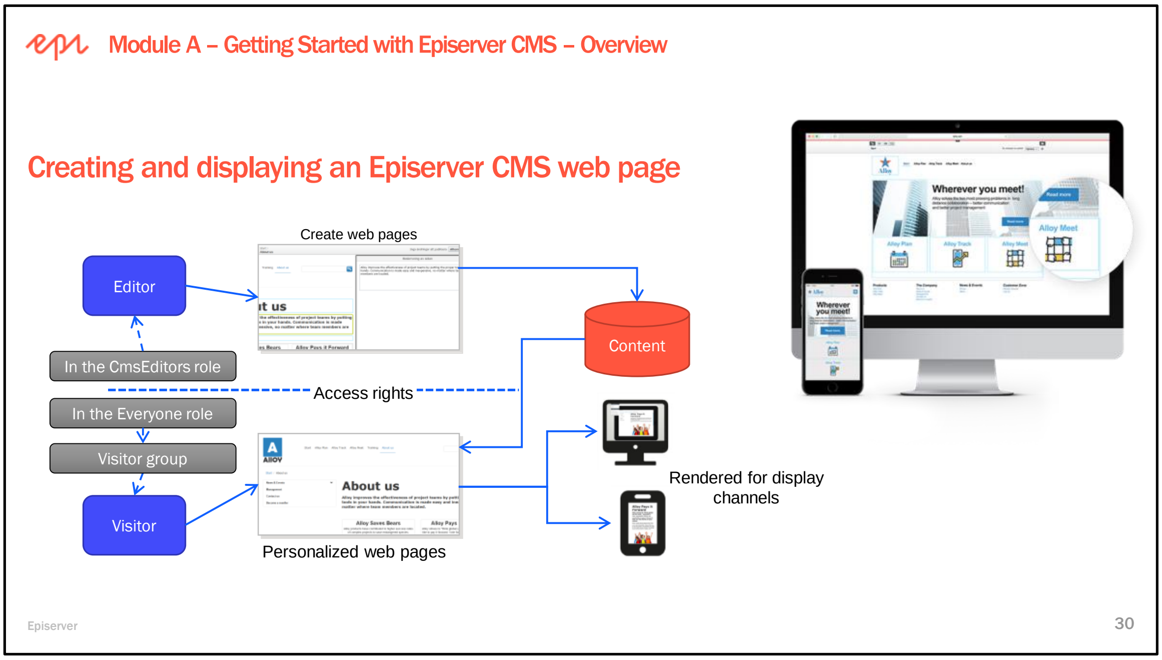
- Through these properties the content type defines the way in which content can be entered into a page, block, media asets, or other type of content. A content type can be associated with multiple content templates, which is useful when publishing content in multiple scenarios. Content types can be created from code or from Admin view.

A **content item** is an instance of the .NET class that defined the content type.

- Content items are used by editors in Edit view to set the properties and fill them with values.
- When a content item is requested by a visitor, the most suitable content template that is associated with the content type is used to render the content.
- Content templates consist of markup, calls to HtmlHelper extension methods, and dynamic programming logic. Content templates in Episerver CMS can be created using either ASP.NET Web Forms or ASP.NET MVC, but we recommend new websites use ASP.NET MVC.

Strongly Typed Models

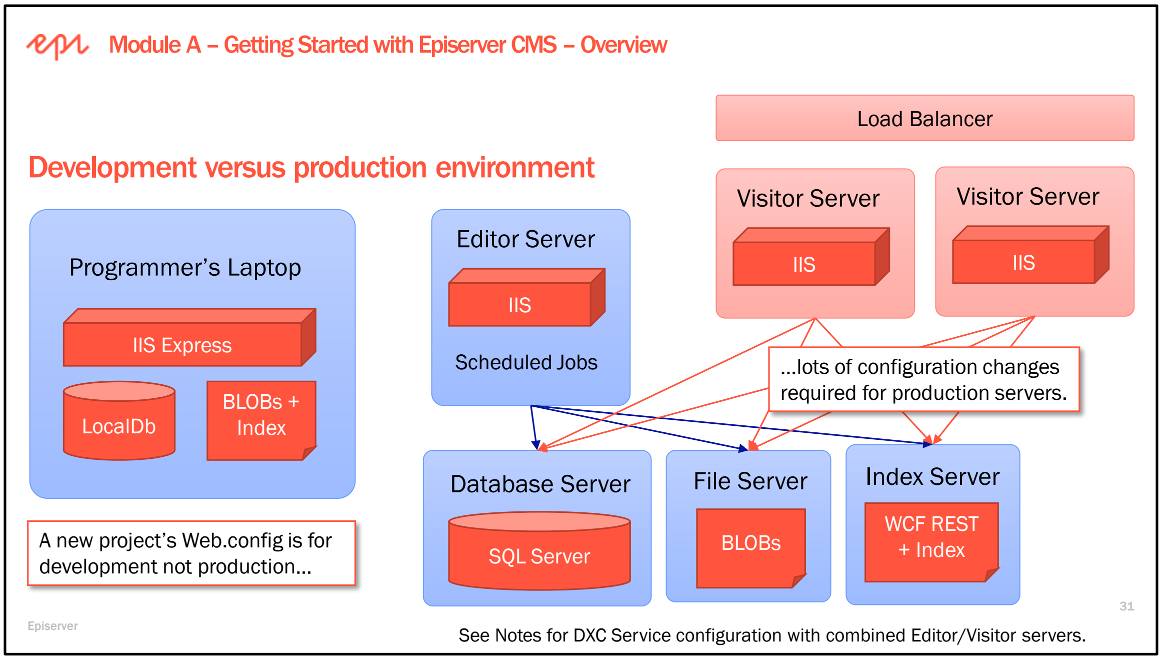
The content system in Episerver CMS supports strongly typed models. This means that when a content item is requested, the instance will be created as the model type that is associated with the content type in question. The APIs contain generic classes and methods to return typed objects and there is also the possibility of defining content types through annotations in code. The detection of code-defined content types is handled via class and property attributes. During site initialization all assemblies in the **bin** folder are scanned and all class types that implement IContent are passed to the synchronization engine. The annotation information is constructed by merging the annotated settings with the settings stored in the database using the administrative interface. Any automatic properties on your content type class will reflect the values of the backing PropertyData collection without the need of writing any code.



Module A – Getting Started with Episerver CMS – Overview

Creating and displaying an Episerver CMS web page

1. When an editor logs on to the website, the system will control what the editor can do and where on the website.
2. Editors will create content in page types, working with content blocks and page layout. Content such as text and links are stored in the database.
3. When a visitor enters the web page, access rights are checked as well as membership in any defined visitor groups and language settings. Depending on these, content starts to load.
4. The graphical design for the website is retrieved together with any images, videos or documents linking to the page.
5. The final web page is assembled and displayed using the appropriate page template, depending on the display device selected by the visitor accessing the page.


Module A – Getting Started with Episerver CMS – Overview

Development versus production environment

A new project's Web.config is for development not production...

See Notes for DXC Service configuration with combined Editor/Visitor servers.

On-Premise decoupled production deployments

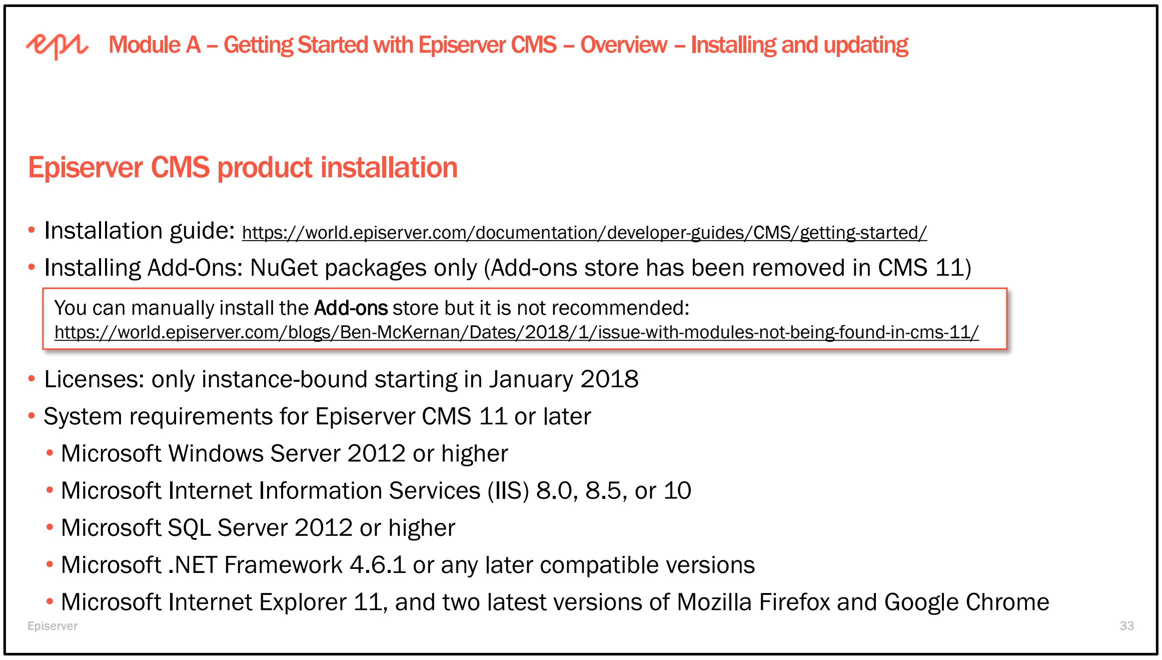
When deploying to production on-premise, you will typically have less flexibility in the number and size of servers and in the depth of security available. Therefore we recommend that you configure a separate server (or servers) for Editors with the Episerver UI enabled, and configure separate load balanced servers for Visitors with the Episerver UI and some features disabled. This provides more control, security, and improved performance and scalability.

http://world.episerver.com/documentation/developer-guides/CMS/security/decoupled-setup/

Cloud production deployments

When deploying to production in the cloud, you will typically have more flexibility in the number and size of servers supporting automatic scaling up and down, and have in depth security with multiple layers: Microsoft Azure infrastructure with permanent Red Teams performing penetration testing, a dedicated Web Application Firewall with rules applied within seconds of newly discovered vulnerabilities, and so on. Therefore all servers for Editors and Visitors can have identical configuration and enabled features.

http://world.episerver.com/documentation/developer-guides/CMS/security/Securing-edit-and-admin-user-interfaces/



Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Episerver CMS product installation

- Installation guide: https://world.episerver.com/documentation/developer-guides/CMS/getting-started/
- Installing Add-Ons: NuGet packages only (Add-ons store has been removed in CMS 11)

  You can manually install the **Add-ons** store but it is not recommended:
  https://world.episerver.com/blogs/Ben-McKernan/Dates/2018/1/issue-with-modules-not-being-found-in-cms-11/
- Licenses: only instance-bound starting in January 2018
- System requirements for Episerver CMS 11 or later
  - Microsoft Windows Server 2012 or higher
  - Microsoft Internet Information Services (IIS) 8.0, 8.5, or 10
  - Microsoft SQL Server 2012 or higher
  - Microsoft .NET Framework 4.6.1 or any later compatible versions
  - Microsoft Internet Explorer 11, and two latest versions of Mozilla Firefox and Google Chrome

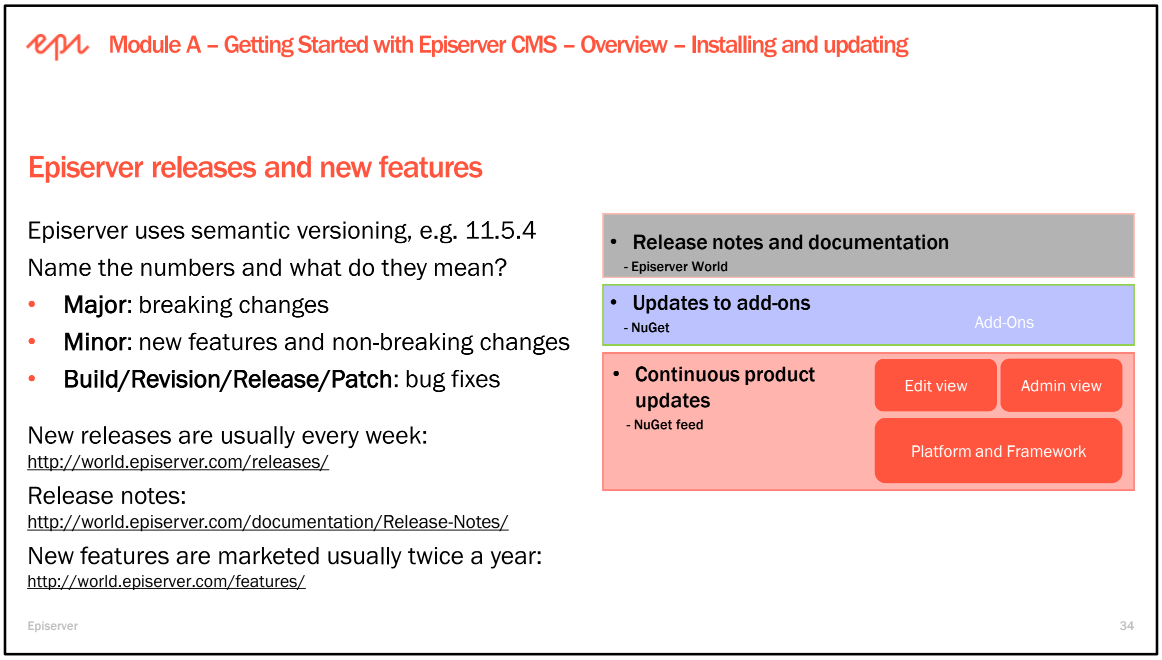
Episerver 33

Installing Episerver (first time installation)
http://world.episerver.com/documentation/Items/Installation-Instructions/installing-episerver/

Available Add-Ons
http://world.episerver.com/add-ons-page/

Licenses
A license is not required for Episerver when using Visual Studio Express & IIS Express. A demo license can be requested good for 45 days if needed.
https://license.episerver.com/

System Requirements for Episerver CMS – Development Environment
Note that these requirements were correct at the time of writing and could have changed since.
The full and up-to-date System Requirements, including Production Environment and Client requirements for Editing, are available on Episerver World: http://world.episerver.com/documentation/Items/System-Requirements/System-Requirements---Episerver/
Note that the requirements are different for a production server and a development and demonstration environment. You will need the Development and demonstration environment setup for this course or when you develop solutions on your local computer.



Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Episerver releases and new features

Episerver uses semantic versioning, e.g. 11.5.4
Name the numbers and what do they mean?

- **Major**: breaking changes
- **Minor**: new features and non-breaking changes
- **Build/Revision/Release/Patch**: bug fixes

New releases are usually every week:
http://world.episerver.com/releases/

Release notes:
http://world.episerver.com/documentation/Release-Notes/

New features are marketed usually twice a year:
http://world.episerver.com/features/

- **Release notes and documentation**
  - Episerver World
- **Updates to add-ons**
  - NuGet
- **Continuous product updates**
  - NuGet feed

Episerver 34

Product updates are published weekly on the NuGet feed. By doing this we can get new functionality and fixes out as soon as they are ready and respond quicker to feedback from our user and developer community.

More detailed information on package content and how to install the updates to your site(s) is available on Episerver World: http://world.episerver.com/releases/

Versioning

We will use semantic versioning according to semver.org to version our packages and specify dependencies: [Major].[Minor].[Patch].[Build]
Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes,
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes.

Note: Microsoft names the version numbers: MAJOR.MINOR.BUILD.REVISION

Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Configuring the Episerver NuGet package source

Episerver NuGet feed

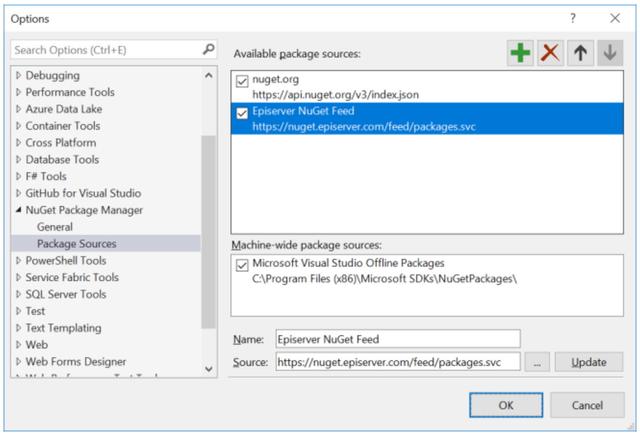
https://nuget.episerver.com/feed/packages.svc/

Use **HTTPS** for enhanced security.

Episerver NuGet Feed Explorer

Filter by creator e.g. Episerver

http://www.david-tec.com/episerver-nuget-feed-explorer/



Available package sources:

- [✓] nuget.org
  https://api.nuget.org/v3/index.json
- [✓] Episerver NuGet Feed
  https://nuget.episerver.com/feed/packages.svc

Machine-wide package sources:

- [✓] Microsoft Visual Studio Offline Packages
  C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\

Name: Episerver NuGet Feed

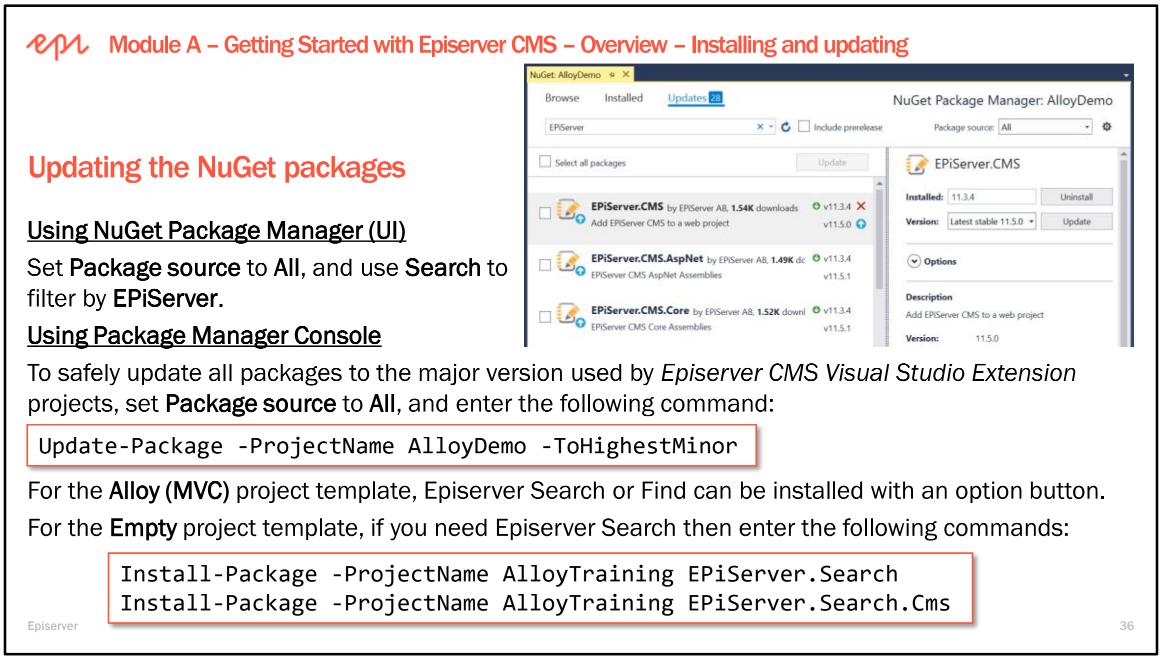
Source: https://nuget.episerver.com/feed/packages.svc

Episerver 35

What does continuous delivery mean?

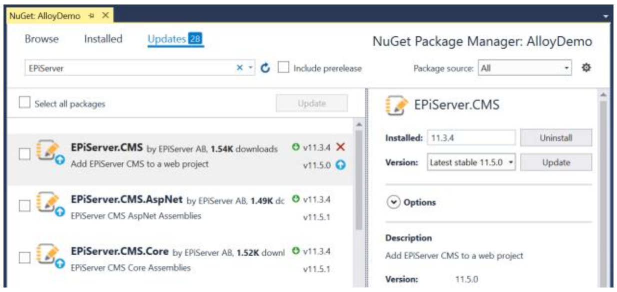
From version 7.5, Episerver has implemented **continuous delivery**. This means:

- Weekly updates for Episerver products.
- Market releases a few times every year that summarize the updates since the previous market release, with mainly a user interface focus.
- Installation via NuGet only; Deployment Center is gone.
- Installed version can be seen in the Plug-in Manager.
- One licence file, Licence.config, that covers all products.
- Customers are recommended to keep their Episerver websites up-to-date.



Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Updating the NuGet packages



NuGet: AlloyDemo

Browse Installed Updates 28

NuGet Package Manager: AlloyDemo

EPiServer

Include prerelease

Package source: All

Select all packages

Update

EPiServer.CMS by EPiServer AB, 1.54K downloads	v11.3.4	v11.5.0
Add EPiServer CMS to a web project		
EPiServer.CMS.AspNet by EPiServer AB, 1.49K dc	v11.3.4	v11.5.1
EPiServer CMS AspNet Assemblies		
EPiServer.CMS.Core by EPiServer AB, 1.52K downl	v11.3.4	v11.5.1
EPiServer CMS Core Assemblies		

EPiServer.CMS

Installed: 11.3.4 Uninstall

Version: Latest stable 11.5.0 Update

Options

Description

Add EPiServer CMS to a web project

Version: 11.5.0

Using NuGet Package Manager (UI)

Set **Package source** to **All**, and use **Search** to filter by **EPiServer**.

Using Package Manager Console

To safely update all packages to the major version used by *Episerver CMS Visual Studio Extension* projects, set **Package source** to **All**, and enter the following command:

```
Update-Package -ProjectName AlloyDemo -ToHighestMinor
```

For the **Alloy (MVC)** project template, Episerver Search or Find can be installed with an option button.

For the **Empty** project template, if you need Episerver Search then enter the following commands:

```
Install-Package -ProjectName AlloyTraining EPiServer.Search
Install-Package -ProjectName AlloyTraining EPiServer.Search.Cms
```

Episerver 36

NuGet Package Manager Console guide
https://docs.microsoft.com/en-us/nuget/tools/package-manager-console

Installing the latest NuGet CLI: https://docs.microsoft.com/en-us/nuget/guides/install-nuget

Update-Package command
https://docs.microsoft.com/en-us/nuget/tools/ps-ref-update-package

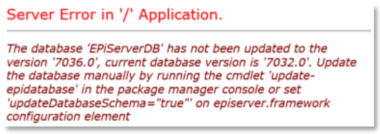
```
Update-Package –ProjectName AlloyDemo –ToHighestMinor
Update-Package –ProjectName AlloyDemo –ToHighestPatch
Update-Package –ProjectName AlloyDemo EPiServer.CMS –Version 10.3
```

To remove a package that is causing conflicts:

```
Uninstall-Package Newtonsoft.Json -ProjectName AlloyDemo –Force
```

Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Updating the Episerver database schema

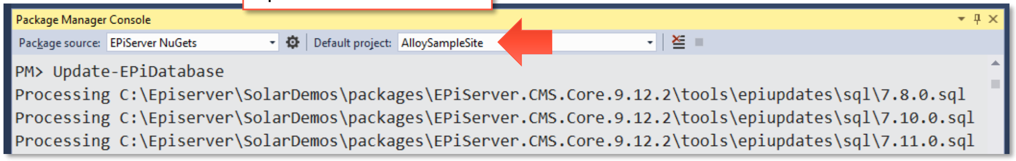

Server Error in '/' Application.

The database 'EPiServerDB' has not been updated to the version '7036.0', current database version is '7032.0'. Update the database manually by running the cmdlet 'update-epidatabase' in the package manager console or set 'updateDatabaseSchema="true"' on episerver.framework configuration element

After updating packages, if you run the site, you might see an exception like this:

In the **Package Manager Console**, select the correct **Default project** and then enter the following PowerShell Cmdlet:

```
Update-EPiDatabase
```


Package Manager Console
Package source: EPiServer NuGets
Default project: AlloySampleSite

```
PM> Update-EPiDatabase
Processing C:\Episerver\SolarDemos\packages\EPiServer.CMS.Core.9.12.2\tools\epiupdates\sql\7.8.0.sql
Processing C:\Episerver\SolarDemos\packages\EPiServer.CMS.Core.9.12.2\tools\epiupdates\sql\7.10.0.sql
Processing C:\Episerver\SolarDemos\packages\EPiServer.CMS.Core.9.12.2\tools\epiupdates\sql\7.11.0.sql
```

Or configure automatic schema changes with attributes in the Web.config. The account used in the database connection string must have suitable rights to EPiServerDB:

```
<episerver.framework updateDatabaseSchema="true" createDatabaseSchema="true" ...
```

Episerver 37

Module A – Getting Started with Episerver CMS – Overview – Installing and updating

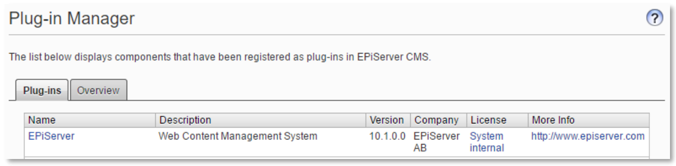
Upgrading Episerver CMS

Upgrading to CMS 11
http://world.episerver.com/documentation/upgrading/Episerver-CMS/cms-11/

Episerver World has a great section that contains information about upgrading between Episerver CMS major versions. Select a version to see specific information regarding breaking changes, and upgrading and migration steps, if any. http://world.episerver.com/documentation/upgrading/Episerver-CMS/

To check the version of Episerver CMS:

- **Admins**: Navigate to **CMS** | **Admin** | **Config** | **Plug-in Manager**



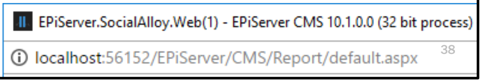
Plug-in Manager

The list below displays components that have been registered as plug-ins in EPiServer CMS.

Plug-ins | Overview

Name	Description	Version	Company	License	More Info
EPiServer	Web Content Management System	10.1.0.0	EPiServer AB	System internal	http://www.episerver.com

- **Editors**: navigate to **CMS** | **Reports** and look at title bar:

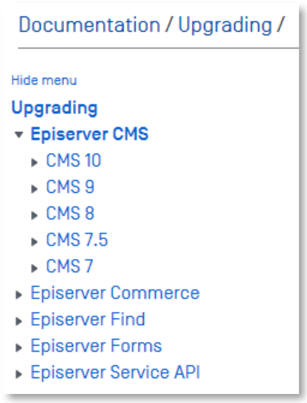


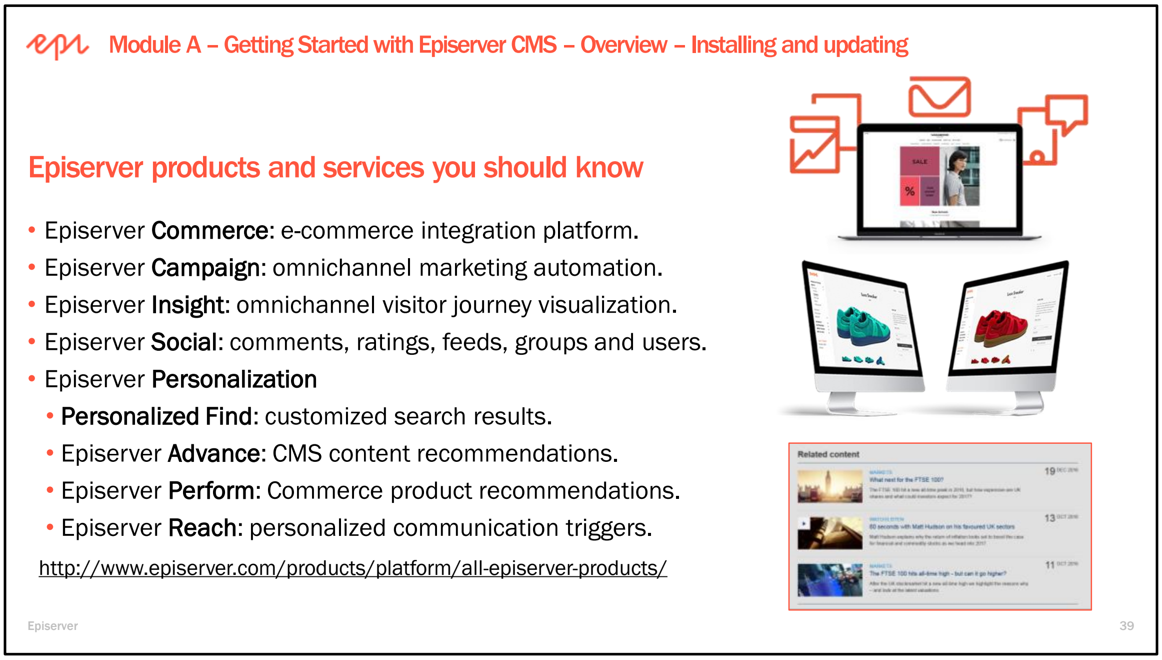
EPiServer.SocialAlloy.Web(1) - EPiServer CMS 10.1.0.0 (32 bit process)
localhost:56152/EPiServer/CMS/Report/default.aspx

Episerver 38

Upgrading is carried out manually and each organization decides whether or not to upgrade. When you upgrade the platform, you do so for all the editors at once.

Ted Gustaf has a blog post about how to upgrade to CMS 10:
https://tedgustaf.com/blog/2016/upgrade-to-episerver-10/





Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Episerver products and services you should know

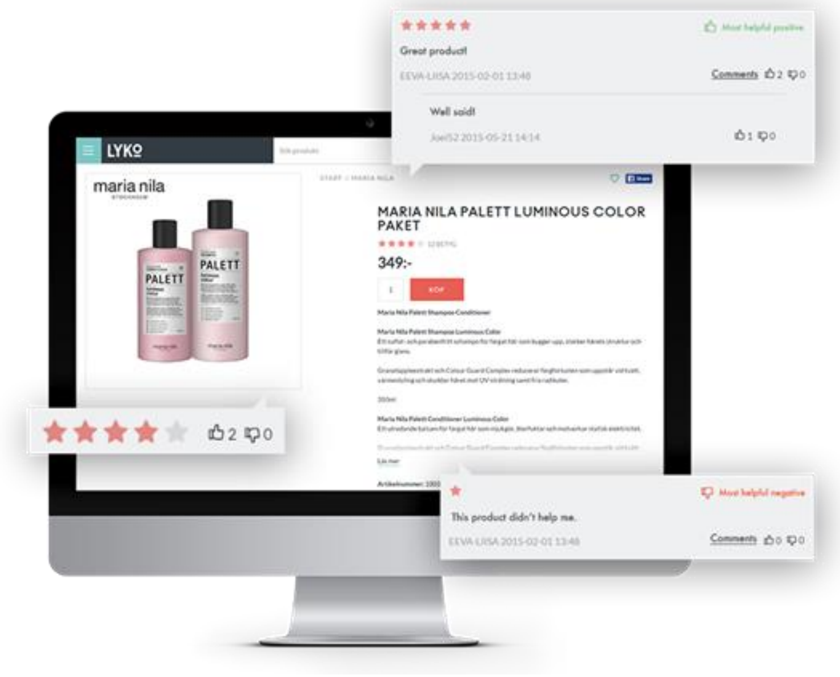
- Episerver **Commerce**: e-commerce integration platform.
- Episerver **Campaign:** omnichannel marketing automation.
- Episerver **Insight**: omnichannel visitor journey visualization.
- Episerver **Social**: comments, ratings, feeds, groups and users.
- Episerver **Personalization**
  - **Personalized Find**: customized search results.
  - Episerver **Advance**: CMS content recommendations.
  - Episerver **Perform**: Commerce product recommendations.
  - Episerver **Reach**: personalized communication triggers.

http://www.episerver.com/products/platform/all-episerver-products/

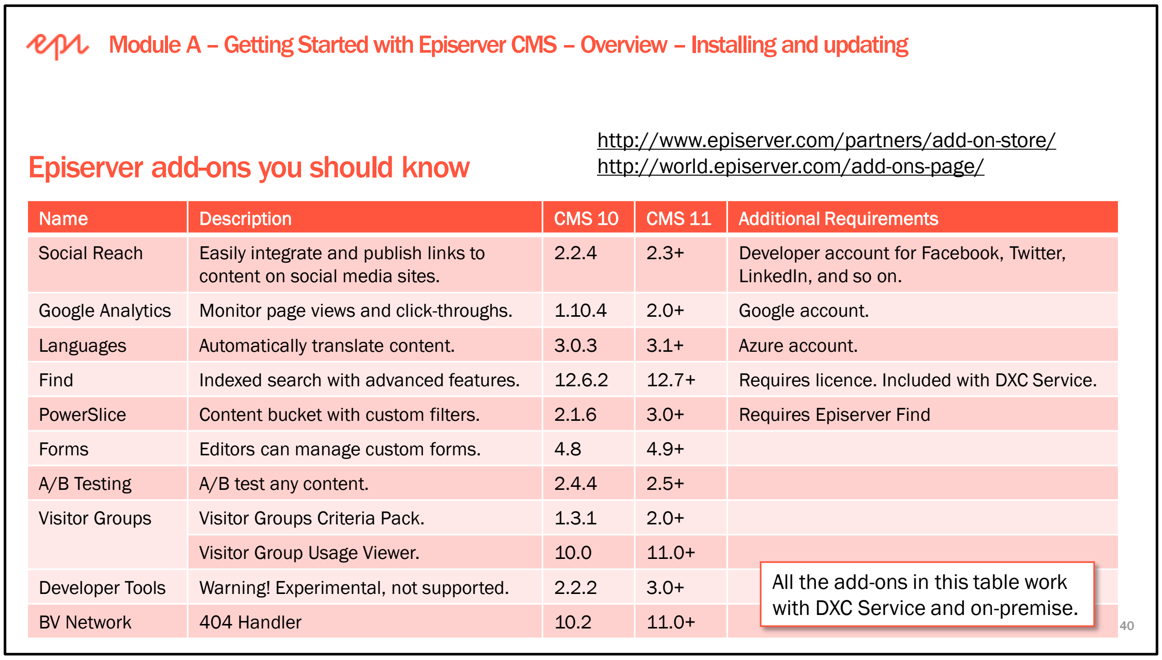
Episerver 39

Episerver Social

User-generated content has proved to be one of the most effective ways to market your products or services, to increase conversions rates, and to enhance employee productivity. Episerver Social offers extremely high performance and reliability with a fluent and easy-to-use API – without the bloat and complexity of other platforms.



https://www.episerver.com/solutions/our-customers/by-industry/ztable/



Module A – Getting Started with Episerver CMS – Overview – Installing and updating

Episerver add-ons you should know

http://www.episerver.com/partners/add-on-store/
http://world.episerver.com/add-ons-page/

Name	Description	CMS 10	CMS 11	Additional Requirements
Social Reach	Easily integrate and publish links to content on social media sites.	2.2.4	2.3+	Developer account for Facebook, Twitter, LinkedIn, and so on.
Google Analytics	Monitor page views and click-throughs.	1.10.4	2.0+	Google account.
Languages	Automatically translate content.	3.0.3	3.1+	Azure account.
Find	Indexed search with advanced features.	12.6.2	12.7+	Requires licence. Included with DXC Service.
PowerSlice	Content bucket with custom filters.	2.1.6	3.0+	Requires Episerver Find
Forms	Editors can manage custom forms.	4.8	4.9+	
A/B Testing	A/B test any content.	2.4.4	2.5+	
Visitor Groups	Visitor Groups Criteria Pack.	1.3.1	2.0+	
	Visitor Group Usage Viewer.	10.0	11.0+	
Developer Tools	Warning! Experimental, not supported.	2.2.2	3.0+	
BV Network	404 Handler	10.2	11.0+	

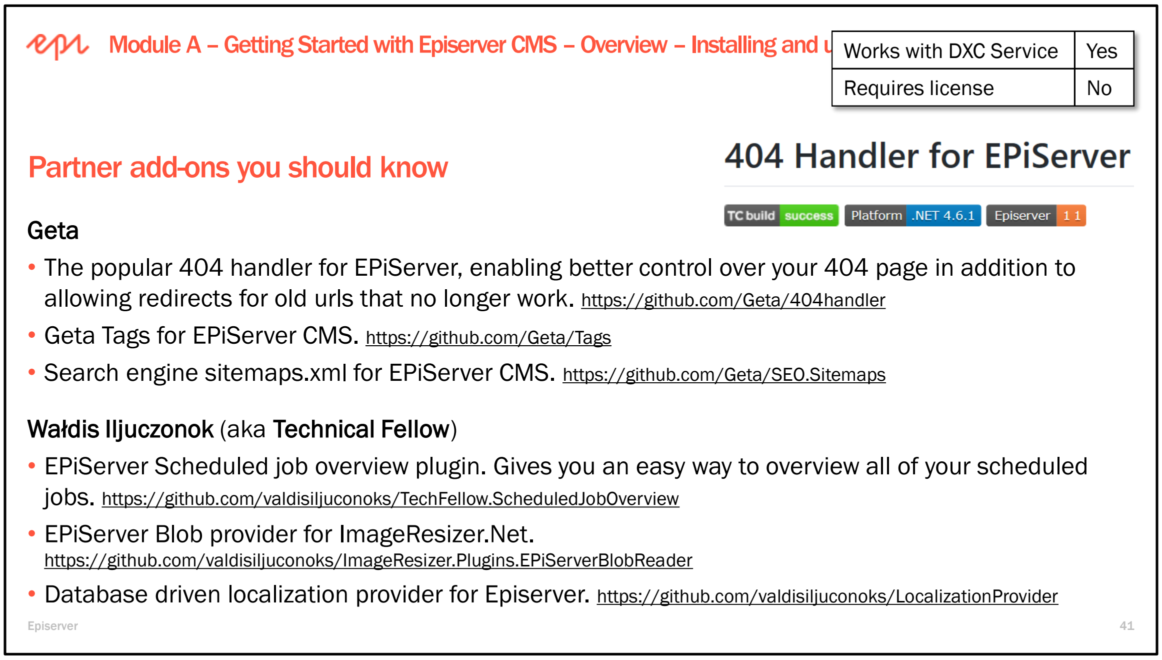
All the add-ons in this table work with DXC Service and on-premise.

40

To install an add-on, enter the following command in Package Manager Console:

```
Install-Package -ProjectName AlloyDemo packagename –Version versionnumber
```

Name	Package
Social Reach	EPiServer.Social
Social	EPiServer.Social.Comments.Site EPiServer.Social.Ratings.Site EPiServer.Social.Groups.Site EPiServer.Social.Moderation.Site EPiServer.Social.ActivityStreams.Site
Google Analytics	EPiServer.GoogleAnalytics
Languages	EPiServer.Labs.Languages
Find	EPiServer.Find.Cms
PowerSlice	PowerSlice
Search (Indexing Service)	EPiServer.Search
Search (CMS Integration)	EPiServer.Search.Cms
Forms	EPiServer.Forms
A/B Testing	EPiServer.Marketing.Testing
Episerver Developer Tools	EPiServer.DeveloperTools
Visitor Groups Criteria Pack	EPiServer.VisitorGroupsCriteriaPack
Visitor Group Usage Viewer	VisitorGroupUsage
Live Monitor	EPiServer.LiveMonitor
BV Network 404 Handler	BVN.404Handler



Module A – Getting Started with Episerver CMS – Overview – Installing and u

Works with DXC Service	Yes
Requires license	No

Partner add-ons you should know

404 Handler for EPiServer

TC build success | Platform .NET 4.6.1 | Episerver 11

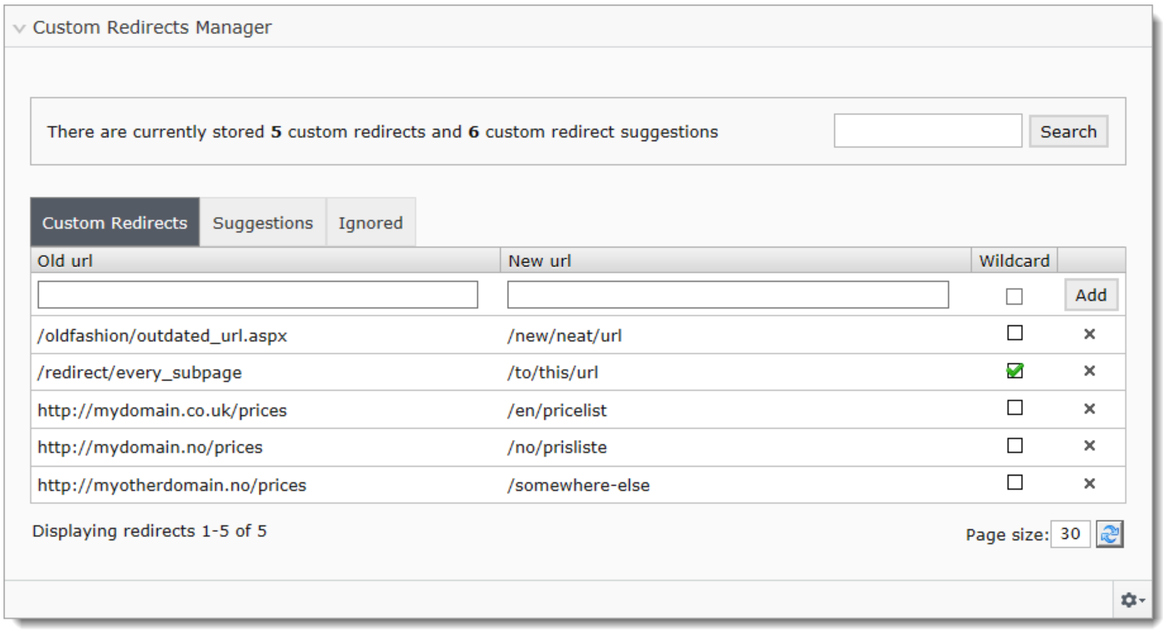
Geta

- The popular 404 handler for EPiServer, enabling better control over your 404 page in addition to allowing redirects for old urls that no longer work. https://github.com/Geta/404handler
- Geta Tags for EPiServer CMS. https://github.com/Geta/Tags
- Search engine sitemaps.xml for EPiServer CMS. https://github.com/Geta/SEO.Sitemaps

Wałdis Iljuczonok (aka **Technical Fellow**)

- EPiServer Scheduled job overview plugin. Gives you an easy way to overview all of your scheduled jobs. https://github.com/valdisiljuconoks/TechFellow.ScheduledJobOverview
- EPiServer Blob provider for ImageResizer.Net. https://github.com/valdisiljuconoks/ImageResizer.Plugins.EPiServerBlobReader
- Database driven localization provider for Episerver. https://github.com/valdisiljuconoks/LocalizationProvider

Episerver 41



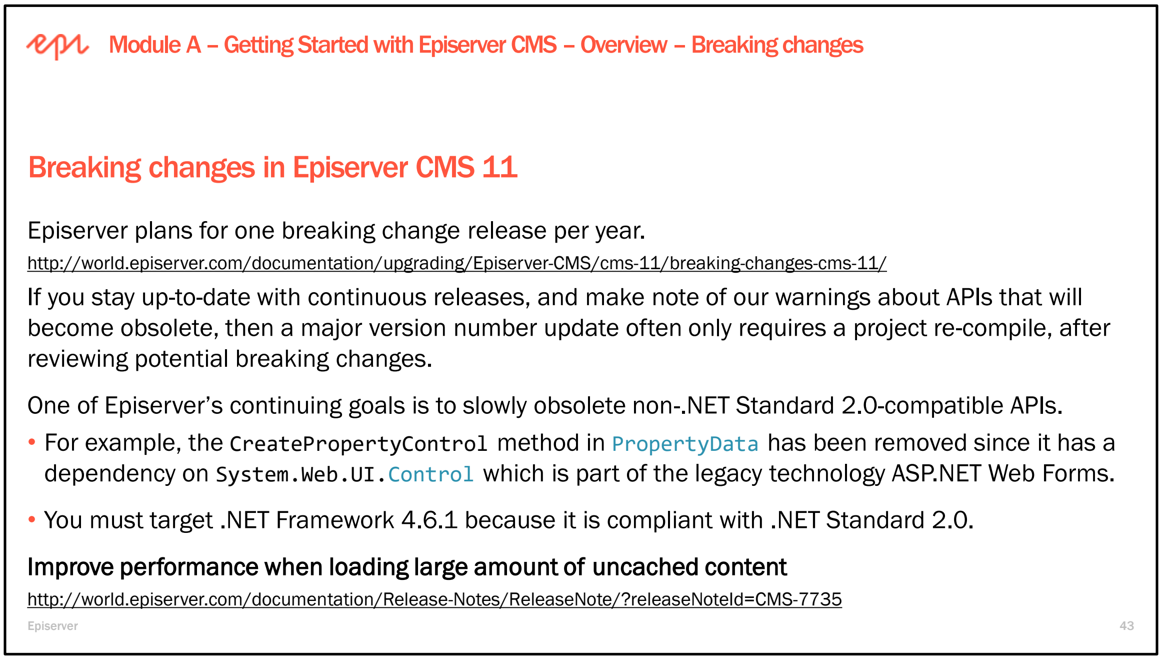
Custom Redirects Manager

There are currently stored **5** custom redirects and **6** custom redirect suggestions

Old url	New url	Wildcard
/oldfashion/outdated_url.aspx	/new/neat/url	☐
/redirect/every_subpage	/to/this/url	✓
http://mydomain.co.uk/prices	/en/pricelist	☐
http://mydomain.no/prices	/no/prisliste	☐
http://myotherdomain.no/prices	/somewhere-else	☐

Displaying redirects 1-5 of 5

Page size: 30



Module A – Getting Started with Episerver CMS – Overview – Breaking changes

Breaking changes in Episerver CMS 11

Episerver plans for one breaking change release per year.
http://world.episerver.com/documentation/upgrading/Episerver-CMS/cms-11/breaking-changes-cms-11/

If you stay up-to-date with continuous releases, and make note of our warnings about APIs that will become obsolete, then a major version number update often only requires a project re-compile, after reviewing potential breaking changes.

One of Episerver's continuing goals is to slowly obsolete non-.NET Standard 2.0-compatible APIs.

- For example, the `CreatePropertyControl` method in `PropertyData` has been removed since it has a dependency on `System.Web.UI.Control` which is part of the legacy technology ASP.NET Web Forms.
- You must target .NET Framework 4.6.1 because it is compliant with .NET Standard 2.0.

Improve performance when loading large amount of uncached content
http://world.episerver.com/documentation/Release-Notes/ReleaseNote/?releaseNoteId=CMS-7735

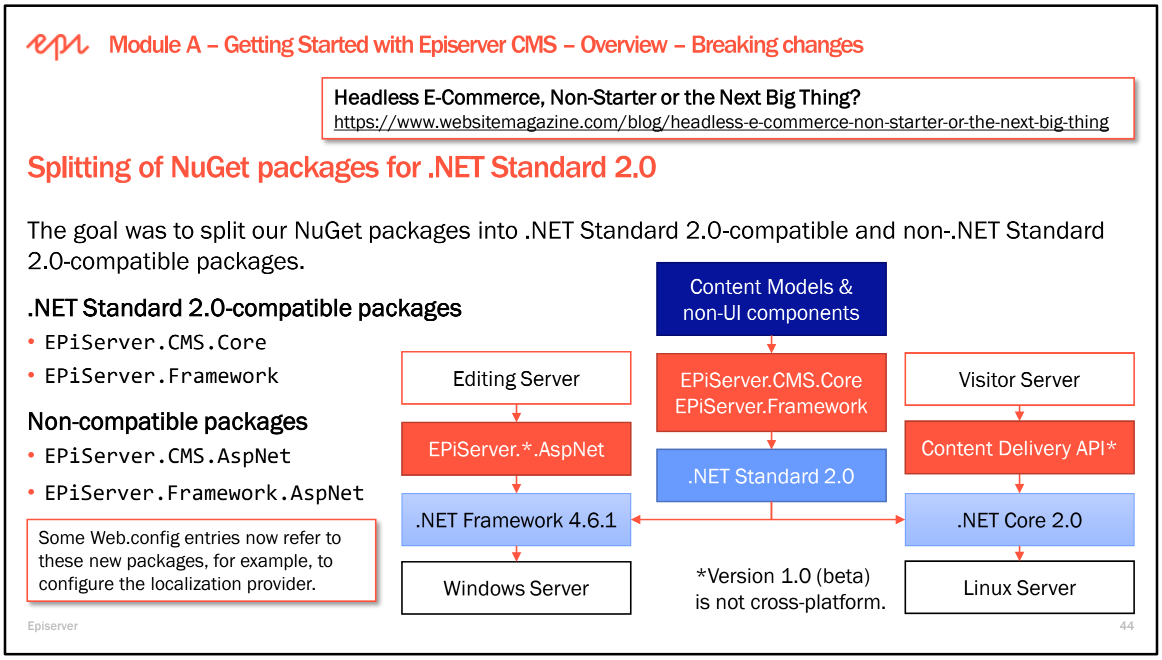
Episerver 43

Resetting passwords with ASP.NET Identity

It was previously not possible to reset a user's password programmatically when using the ASP.NET Identity provider. Due to a bug, the method for resetting the password simply generated a reset token but never changed the password.

ApplicationUIUserManager<TUser>.ResetPassword(IUIUser user) will now throw a not supported exception. This is because ASP.NET Identity does not support generating new passwords for security reasons, i.e. it is bad practice to send a new password to the user in plain text. The new method ResetPassword(IUIUser user, string newPassword) should be used instead.

Both methods will still work when using the API with older membership providers.



Module A – Getting Started with Episerver CMS – Overview – Breaking changes

Headless E-Commerce, Non-Starter or the Next Big Thing?
https://www.websitemagazine.com/blog/headless-e-commerce-non-starter-or-the-next-big-thing

Splitting of NuGet packages for .NET Standard 2.0

The goal was to split our NuGet packages into .NET Standard 2.0-compatible and non-.NET Standard 2.0-compatible packages.

.NET Standard 2.0-compatible packages
- `EPiServer.CMS.Core`
- `EPiServer.Framework`

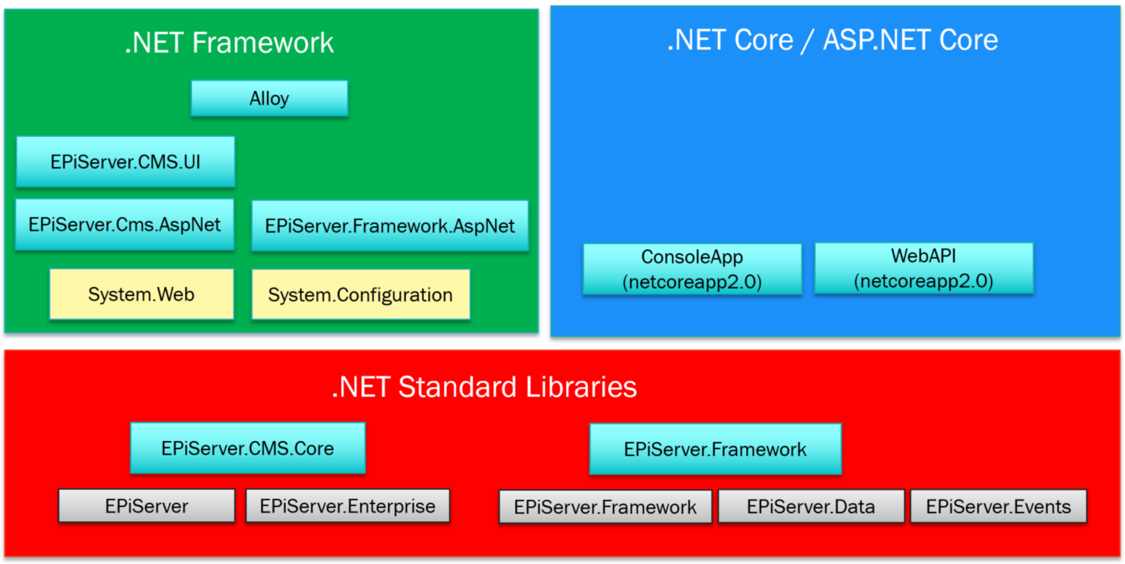
Non-compatible packages
- `EPiServer.CMS.AspNet`
- `EPiServer.Framework.AspNet`

Some Web.config entries now refer to these new packages, for example, to configure the localization provider.

*Version 1.0 (beta) is not cross-platform.

Episerver 44

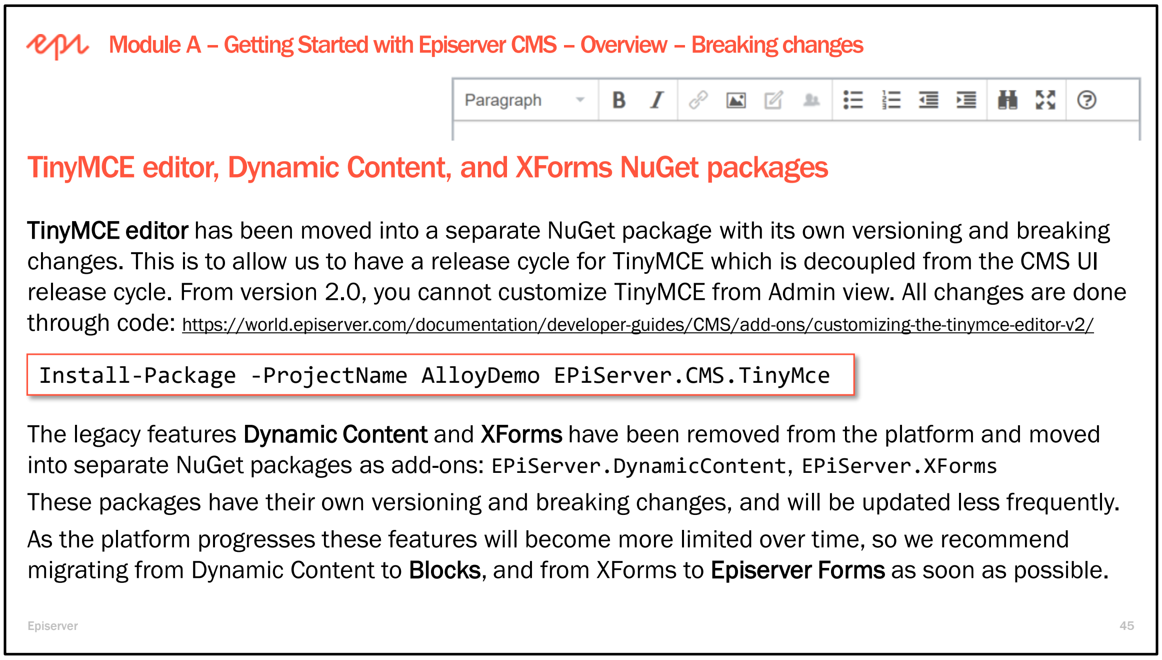
EPiServer CMS 11



StructureMap NuGet package

A new NuGet package, `EPiServer.ServiceLocation.StructureMap`, supports:
- Existing signed StructureMap 3, and
- New unsigned StructureMap 4.

The package only contains the integration, so it has NuGet dependencies on the official StructureMap packages. Moving the dependency to a NuGet package is the same approach we have for logging, where we have abstractions in the platform and an integration in a separate NuGet package. This will allow us to more easily swap to the dependency injection system that is shipped with .NET Core, if we choose to do so in the future.



Module A – Getting Started with Episerver CMS – Overview – Breaking changes

TinyMCE editor, Dynamic Content, and XForms NuGet packages

TinyMCE editor has been moved into a separate NuGet package with its own versioning and breaking changes. This is to allow us to have a release cycle for TinyMCE which is decoupled from the CMS UI release cycle. From version 2.0, you cannot customize TinyMCE from Admin view. All changes are done through code: https://world.episerver.com/documentation/developer-guides/CMS/add-ons/customizing-the-tinymce-editor-v2/

```
Install-Package -ProjectName AlloyDemo EPiServer.CMS.TinyMce
```

The legacy features **Dynamic Content** and **XForms** have been removed from the platform and moved into separate NuGet packages as add-ons: `EPiServer.DynamicContent`, `EPiServer.XForms`

These packages have their own versioning and breaking changes, and will be updated less frequently.

As the platform progresses these features will become more limited over time, so we recommend migrating from Dynamic Content to **Blocks**, and from XForms to **Episerver Forms** as soon as possible.

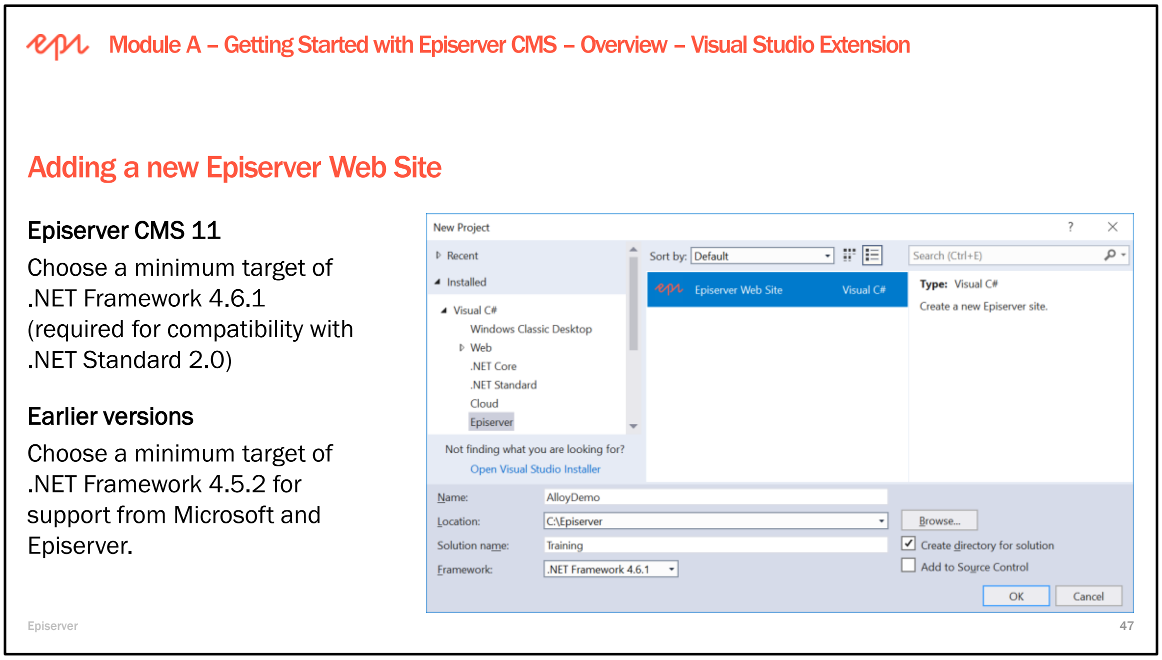
Episerver 45

Deprecations and other changes

jQuery
The jQuery library that is bundled with the CMS UI is being deprecated and should no longer be used.

Gadget Framework
The gadget framework has been deprecated but will remain in the product for CMS 11. We recommend that you convert your [Gadget]s to [Component]s now.

Other Changes
Read this blog article for a detailed list of other changes:
http://world.episerver.com/blogs/Ben-McKernan/Dates/2017/9/planned-breaking-changes-2017-cms-ui/



Module A – Getting Started with Episerver CMS – Overview – Visual Studio Extension

Adding a new Episerver Web Site

Episerver CMS 11

Choose a minimum target of .NET Framework 4.6.1 (required for compatibility with .NET Standard 2.0)

Earlier versions

Choose a minimum target of .NET Framework 4.5.2 for support from Microsoft and Episerver.

New Project

Installed > Visual C# > Episerver

Episerver Web Site — Visual C#

Type: Visual C#

Create a new Episerver site.

Name: AlloyDemo

Location: C:\Episerver

Solution name: Training

Framework: .NET Framework 4.6.1

☑ Create directory for solution

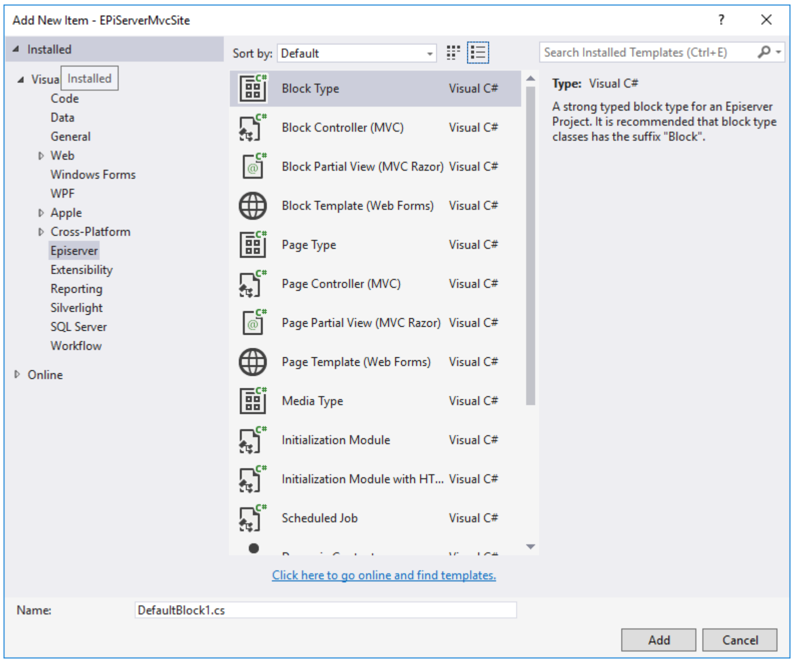
☐ Add to Source Control

Episerver 47

Episerver CMS Visual Studio Extension version 11.3.0.359 was released on 30th April 2018. It includes EPiServer.CMS.Core 11.5.4 and fixes an issue with support for Visual Studio 2017 version 15.6 or later. Alloy is now based on open source version from GitHub: https://github.com/episerver/alloy-mvc-template

When the Episerver CMS Visual Studio Extension has been installed, the Episerver project and item templates will be available when using the Add > New Item option in Visual Studio. A current list of included templates can be found in the Visual Studio Gallery: http://visualstudiogallery.msdn.microsoft.com

- Block Controller (MVC)
- Block Razor View (MVC)
- Block Template
- Block Type
- Block View
- Custom Property
- Initialization Module
- Initialization Module with HTTP events
- Page Controller (MVC)
- Page Razor View (MVC)
- Page Template
- Page Type
- Scheduled job
- User Control
- Visitor Group Criterion
- Media Type



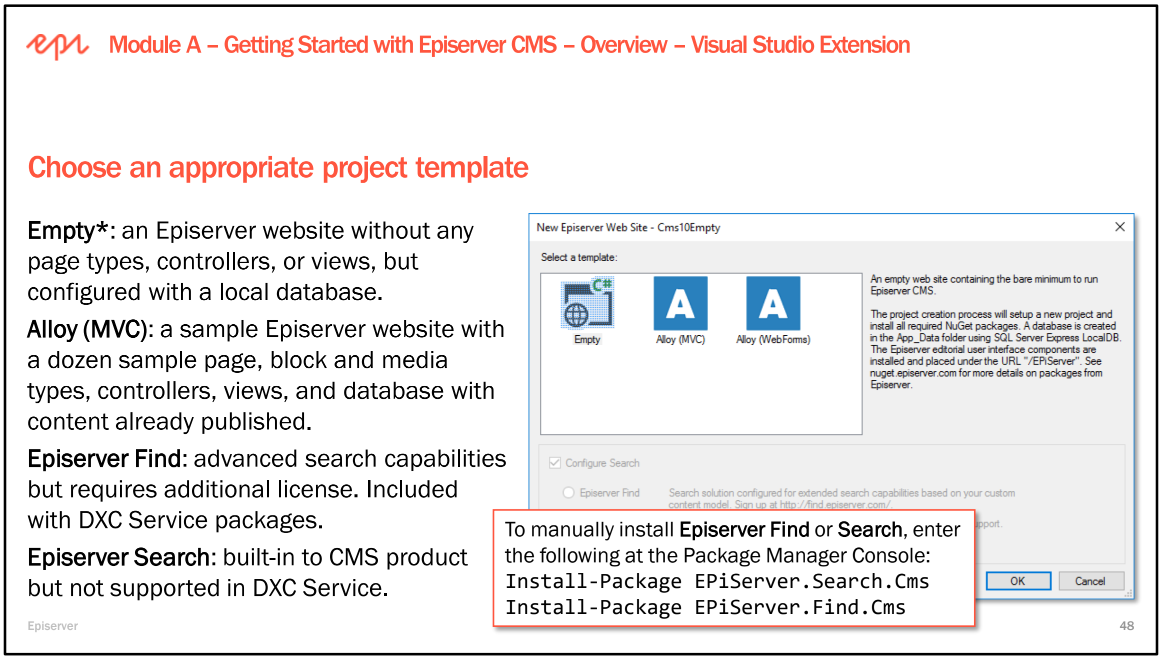
Add New Item - EPiServerMvcSite

Block Type	Visual C#
Block Controller (MVC)	Visual C#
Block Partial View (MVC Razor)	Visual C#
Block Template (Web Forms)	Visual C#
Page Type	Visual C#
Page Controller (MVC)	Visual C#
Page Partial View (MVC Razor)	Visual C#
Page Template (Web Forms)	Visual C#
Media Type	Visual C#
Initialization Module	Visual C#
Initialization Module with HT...	Visual C#
Scheduled Job	Visual C#

Type: Visual C#

A strong typed block type for an Episerver Project. It is recommended that block type classes has the suffix "Block".

Name: DefaultBlock1.cs



Module A – Getting Started with Episerver CMS – Overview – Visual Studio Extension

Choose an appropriate project template

Empty*: an Episerver website without any page types, controllers, or views, but configured with a local database.

Alloy (MVC): a sample Episerver website with a dozen sample page, block and media types, controllers, views, and database with content already published.

Episerver Find: advanced search capabilities but requires additional license. Included with DXC Service packages.

Episerver Search: built-in to CMS product but not supported in DXC Service.

New Episerver Web Site - Cms10Empty

Select a template:

An empty web site containing the bare minimum to run Episerver CMS.

The project creation process will setup a new project and install all required NuGet packages. A database is created in the App_Data folder using SQL Server Express LocalDB. The Episerver editorial user interface components are installed and placed under the URL "/EPiServer". See nuget.episerver.com for more details on packages from Episerver.

☑ Configure Search

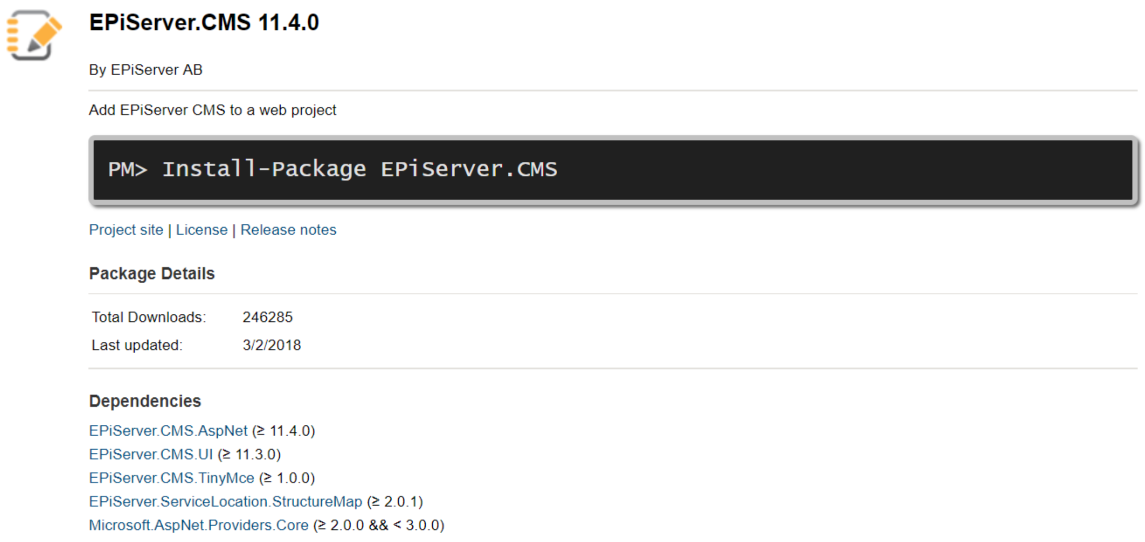
○ Episerver Find — Search solution configured for extended search capabilities based on your custom content model. Sign up at http://find.episerver.com/.

To manually install **Episerver Find** or **Search**, enter the following at the Package Manager Console:

```
Install-Package EPiServer.Search.Cms
Install-Package EPiServer.Find.Cms
```

Episerver 48

To add Episerver to an existing ASP.NET MVC project



EPiServer.CMS 11.4.0

By EPiServer AB

Add EPiServer CMS to a web project

```
PM> Install-Package EPiServer.CMS
```

Project site | License | Release notes

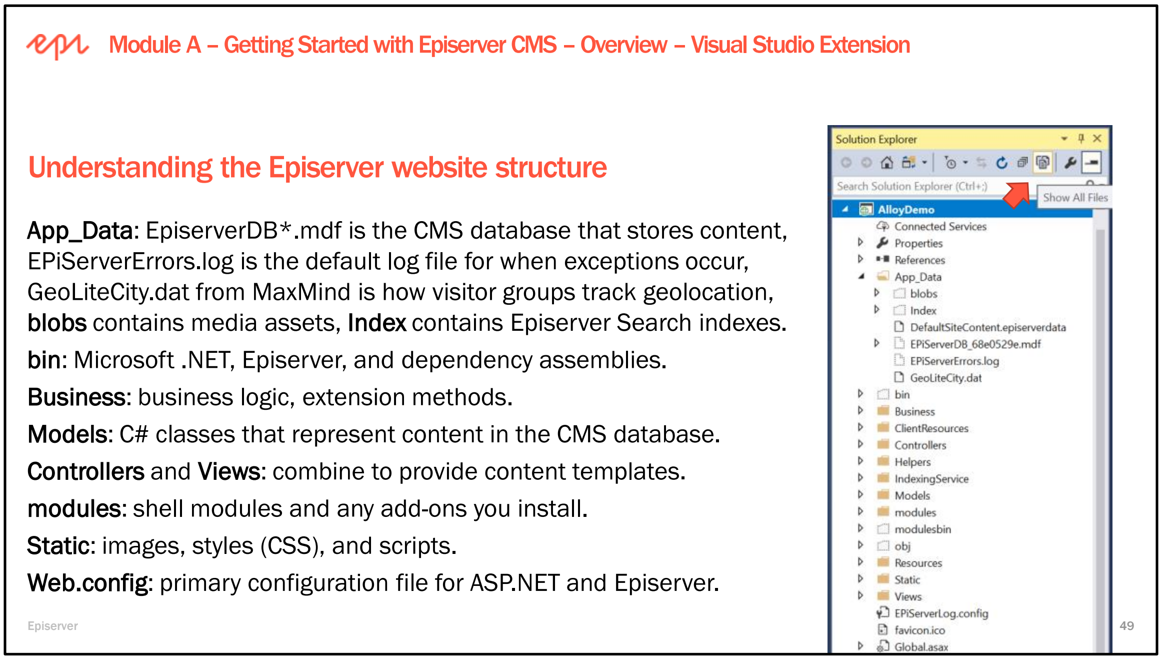
Package Details

Total Downloads:	246285
Last updated:	3/2/2018

Dependencies

EPiServer.CMS.AspNet (≥ 11.4.0)
EPiServer.CMS.UI (≥ 11.3.0)
EPiServer.CMS.TinyMce (≥ 1.0.0)
EPiServer.ServiceLocation.StructureMap (≥ 2.0.1)
Microsoft.AspNet.Providers.Core (≥ 2.0.0 && < 3.0.0)

https://nuget.episerver.com/en/OtherPages/Package/?packageId=EPiServer.CMS



Module A – Getting Started with Episerver CMS – Overview – Visual Studio Extension

Understanding the Episerver website structure

App_Data: EpiserverDB*.mdf is the CMS database that stores content, EPiServerErrors.log is the default log file for when exceptions occur, GeoLiteCity.dat from MaxMind is how visitor groups track geolocation, **blobs** contains media assets, **Index** contains Episerver Search indexes.

bin: Microsoft .NET, Episerver, and dependency assemblies.

Business: business logic, extension methods.

Models: C# classes that represent content in the CMS database.

Controllers and **Views**: combine to provide content templates.

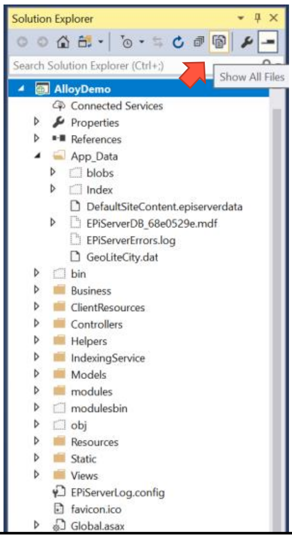
modules: shell modules and any add-ons you install.

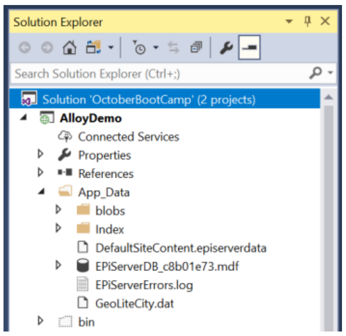
Static: images, styles (CSS), and scripts.

Web.config: primary configuration file for ASP.NET and Episerver.

Episerver

49





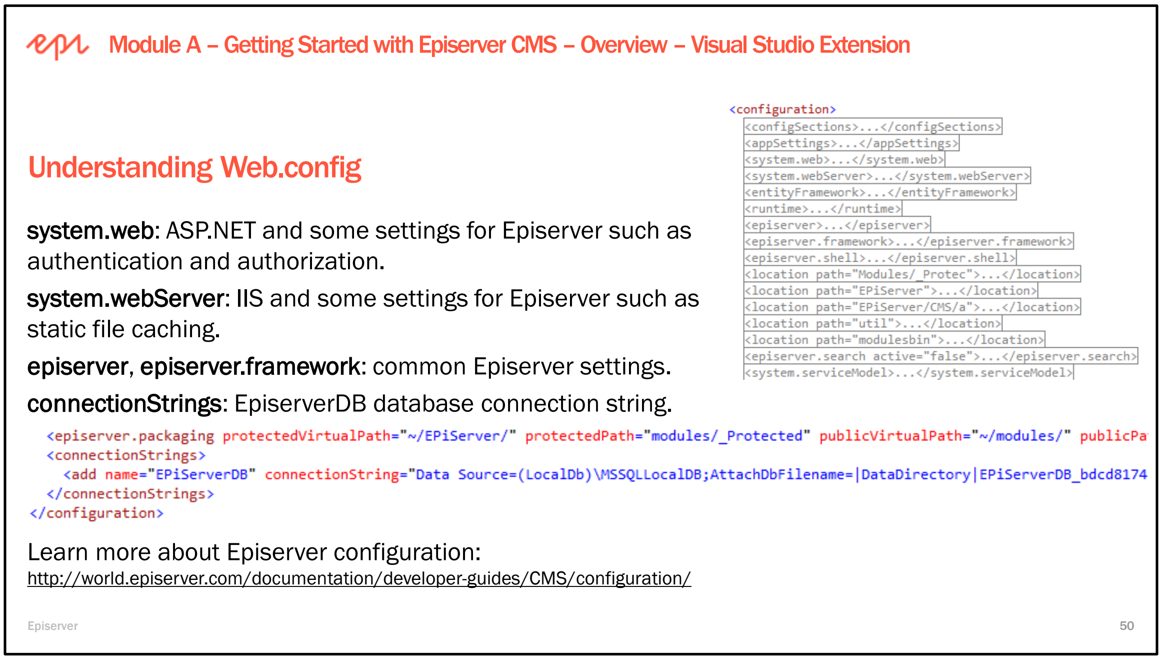
An empty project folder structure:

- **App-Data** for content e.g. relational database, BLOBs, index.
- **Business** for business logic and helper libraries added during development. For example, for containing subfolders: /Channels, /Rendering, /Initialization, etc.
- **Controllers** for controller classes handling user input and responses.
- **Models** for content classes representing and manipulating data.
- **Static** for design and layout files such as scripts, images, and style sheets. For example, /gfx, /css, /js
- **Views** for renderers (MVC), user controls, templates, and master pages.

Other folders in the structure, (not shown on this slide):
/**ClientResources** – for containing subfolders for images, scripts and files
/**Resources** – language files

Global.asax and **App_Start**: in a traditional ASP.NET MVC site, the **Application_Start** method calls multiple types in the **App_Start** folder to set up routes, bundles, and so on. Episerver's **Global** type sets up its own custom routing. To do the equivalent, create Episerver Initialization Modules.

```
public class EpiserverApplication : EPiServer.Global
{
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();

        //Tip: Want to call the Episerver API on startup?
        //Add an initialization module instead
    }
}
```



Module A – Getting Started with Episerver CMS – Overview – Visual Studio Extension

Understanding Web.config

system.web: ASP.NET and some settings for Episerver such as authentication and authorization.

system.webServer: IIS and some settings for Episerver such as static file caching.

episerver, **episerver.framework**: common Episerver settings.

connectionStrings: EpiserverDB database connection string.

```
<configuration>
  <configSections>...</configSections>
  <appSettings>...</appSettings>
  <system.web>...</system.web>
  <system.webServer>...</system.webServer>
  <entityFramework>...</entityFramework>
  <runtime>...</runtime>
  <episerver>...</episerver>
  <episerver.framework>...</episerver.framework>
  <episerver.shell>...</episerver.shell>
  <location path="Modules/_Protec">...</location>
  <location path="EPiServer">...</location>
  <location path="EPiServer/CMS/a">...</location>
  <location path="util">...</location>
  <location path="modulesbin">...</location>
  <episerver.search active="false">...</episerver.search>
  <system.serviceModel>...</system.serviceModel>
```

```
  <episerver.packaging protectedVirtualPath="~/EPiServer/" protectedPath="modules/_Protected" publicVirtualPath="~/modules/" publicPa
  <connectionStrings>
    <add name="EPiServerDB" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|EPiServerDB_bdcd8174
  </connectionStrings>
</configuration>
```

Learn more about Episerver configuration:
http://world.episerver.com/documentation/developer-guides/CMS/configuration/

Episerver

50

```
<episerver>
  <applicationSettings
    httpCacheability="Public"
    pageValidateTemplate="false"
    uiShowGlobalizationUserInterface="true"
    uiUrl="~/EPiServer/CMS/"
    urlRebaseKind="ToRootRelative" />
```

Path to access Episerver CMS user interface

```
<episerver.framework>
  <appData basePath="App_Data" />
  <scanAssembly forceBinFolderScan="true" />
  <virtualPathProviders>
    <clear />
    <add name="ProtectedModules"
          virtualPath="~/EPiServer/"
          physicalPath="Modules\_Protected"
          type="EPiServer.Web.Hosting.VirtualPathNonUnifiedProvider, EPiServer.Framework" />
  </virtualPathProviders>
  <geolocation defaultProvider="maxmind">
    <providers>
      <add name="maxmind"
            type="EPiServer.Personalization.Providers.MaxMind.GeolocationProvider, EPiServer..."
            databaseFileName="App_Data\GeoLiteCity.dat" />
    </providers>
  </geolocation>
```

Path to store content, including SQL database, BLOBs, logs, and index

Paths to the Episerver user interface code

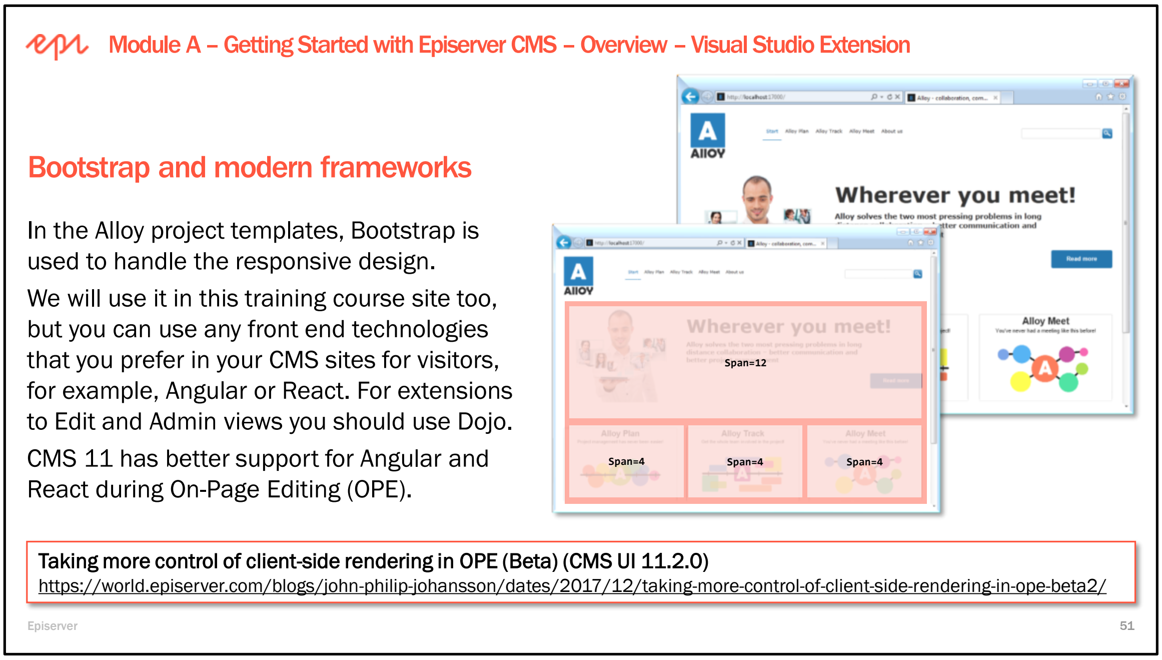
Configuring geolocation provider (used by visitor group criteria)

MaxMind GeoIP2

Episerver's geolocation visitor group criteria depends on MaxMind's legacy database that they are not updating any more. To use MaxMind's latest database, install the following package and update your Web.config as described at the following link:
https://world.episerver.com/blogs/K-Khan-/Dates/2016/10/maxmind-geolite2-on-nuget/

```
Install-Package PixieEPiServerExtensionMaxMindGeoIP2
```

http://nuget.episerver.com/en/OtherPages/Package/?packageId=PixieEPiServerExtensionMaxMindGeoIP2



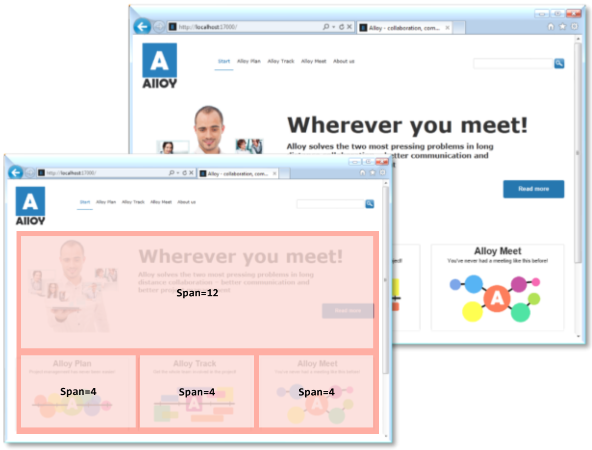
Module A – Getting Started with Episerver CMS – Overview – Visual Studio Extension

Bootstrap and modern frameworks

In the Alloy project templates, Bootstrap is used to handle the responsive design.

We will use it in this training course site too, but you can use any front end technologies that you prefer in your CMS sites for visitors, for example, Angular or React. For extensions to Edit and Admin views you should use Dojo.

CMS 11 has better support for Angular and React during On-Page Editing (OPE).



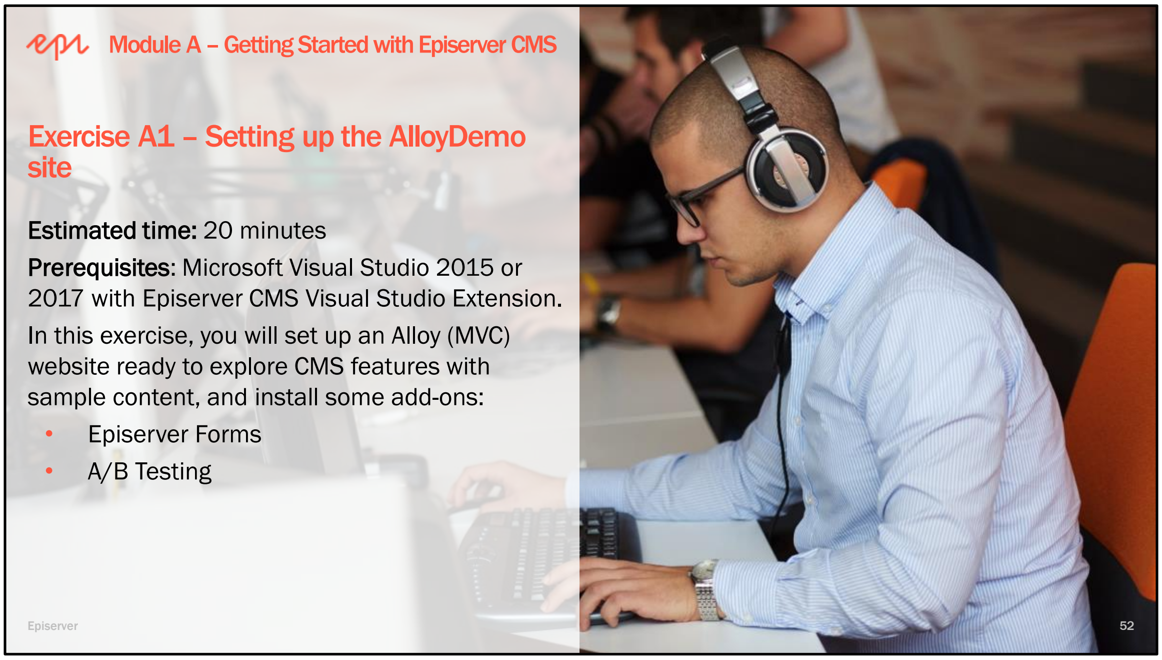
Taking more control of client-side rendering in OPE (Beta) (CMS UI 11.2.0)
https://world.episerver.com/blogs/john-philip-johansson/dates/2017/12/taking-more-control-of-client-side-rendering-in-ope-beta2/

Episerver 51

Ted Nyberg who developed the Alloy templates has written a very good article on Episerver World about how bootstrap is used for the Alloy site markup.
http://world.episerver.com/Articles/Items/Alloy-Templates-for-Episerver-CMS-7/

Valdis Iljuconoks wrote a nice article on bootstrap aware Content Area
http://tech-fellow.net/2015/04/02/bootstrap-aware-content-area-for-episerver-8-0/

Taking control of client-side rendering in OPE (Beta)
https://world.episerver.com/blogs/john-philip-johansson/dates/2017/10/taking-control-of-client-side-rendering-in-ope-beta/



Module A – Getting Started with Episerver CMS

Exercise A1 – Setting up the AlloyDemo site

Estimated time: 20 minutes

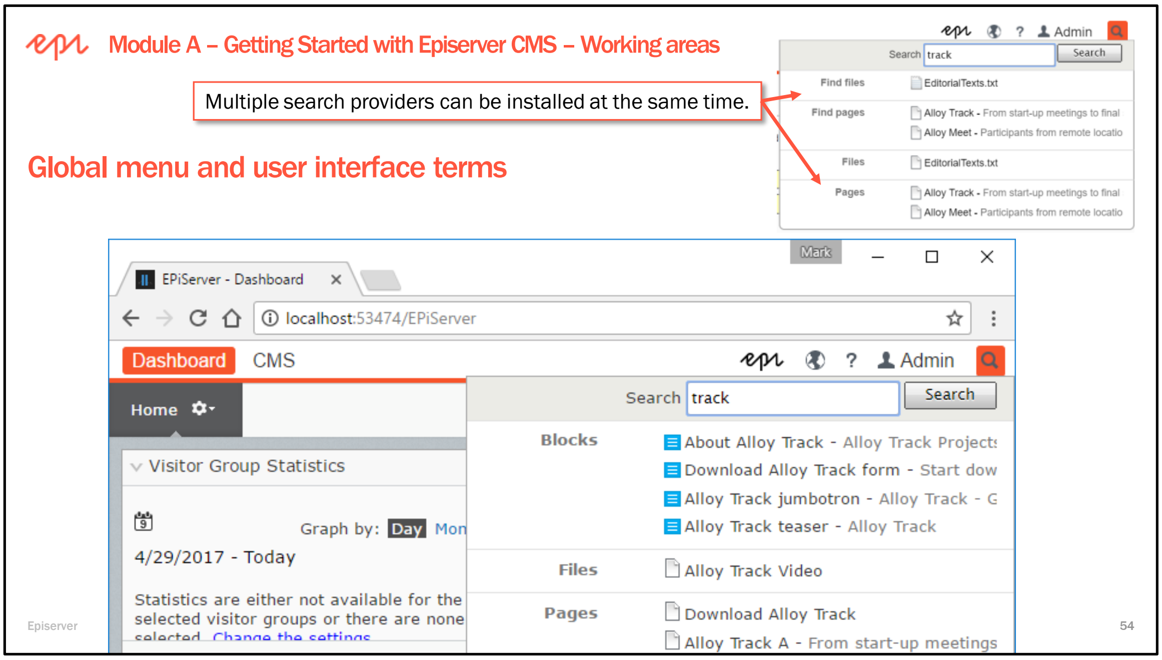
Prerequisites: Microsoft Visual Studio 2015 or 2017 with Episerver CMS Visual Studio Extension.

In this exercise, you will set up an Alloy (MVC) website ready to explore CMS features with sample content, and install some add-ons:

- Episerver Forms
- A/B Testing

Episerver

52



Module A – Getting Started with Episerver CMS – Working areas

Multiple search providers can be installed at the same time.

Global menu and user interface terms

Admin
Search track
Find files EditorialTexts.txt
Find pages Alloy Track - From start-up meetings to final
Alloy Meet - Participants from remote locatio
Files EditorialTexts.txt
Pages Alloy Track - From start-up meetings to final
Alloy Meet - Participants from remote locatio
EPiServer - Dashboard
localhost:53474/EPiServer
Dashboard
CMS
Home
Visitor Group Statistics
Graph by: Day Mon
4/29/2017 - Today
Statistics are either not available for the selected visitor groups or there are none selected. Change the settings
Blocks
About Alloy Track - Alloy Track Projects
Download Alloy Track form - Start dow
Alloy Track jumbotron - Alloy Track - G
Alloy Track teaser - Alloy Track
Files
Alloy Track Video
Pages
Download Alloy Track
Alloy Track A - From start-up meetings
Episerver
54

Episerver CMS user interface terms

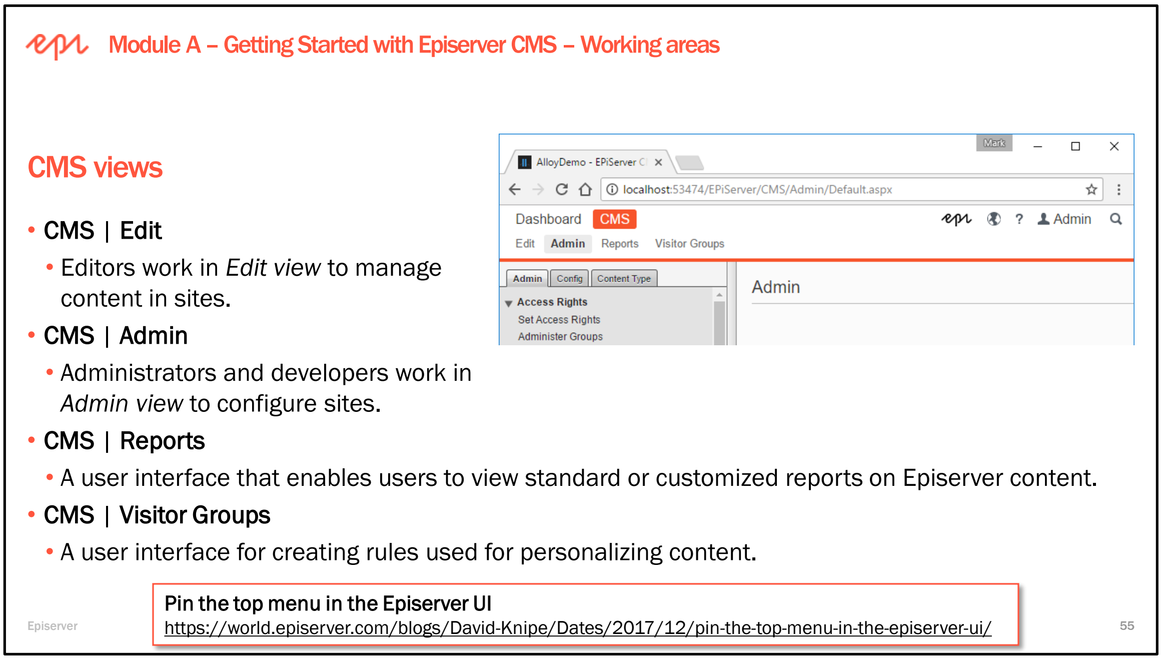
- **Quick Access menu**: once a user is logged in, gives quick access to Edit view and Dashboard.
- **Global menu**: Dashboard, CMS, Commerce, Find, based on access rights, links to Episerver site, Live view, help, user settings, and global search.
- **Dashboard** containing **Gadgets** in one, two, or three columns.
- **Edit view**: two customizable panes on left and right with optional gadgets; **Navigation** on left, **Assets** on right.
- **Page Information Area**: path to page, save information, notifications, publish options, view toggle, and a **Toolbar** to: add content, toggle view settings, toggle preview, toggle compare versions.
- **Page Tree**: hierarchical structure for site.
- **Context menus**: "hamburger" menus for access to actions for the selected item.
- **Search**: at the top of the **Navigation** and **Assets** panes are search boxes filtered by those areas.

Dashboard

- Central customizable workspace used for easy navigation and integration of different products
- Dashboard is personalized with user preferences
- Dashboard Tabs and Components can be made available to certain users or groups

Global Search searches for ID, name, or keywords

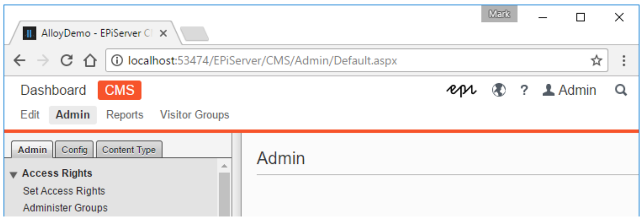
- Customizable: create your own search-providers by implementing the ISearchProvider interface.



Module A – Getting Started with Episerver CMS – Working areas

CMS views

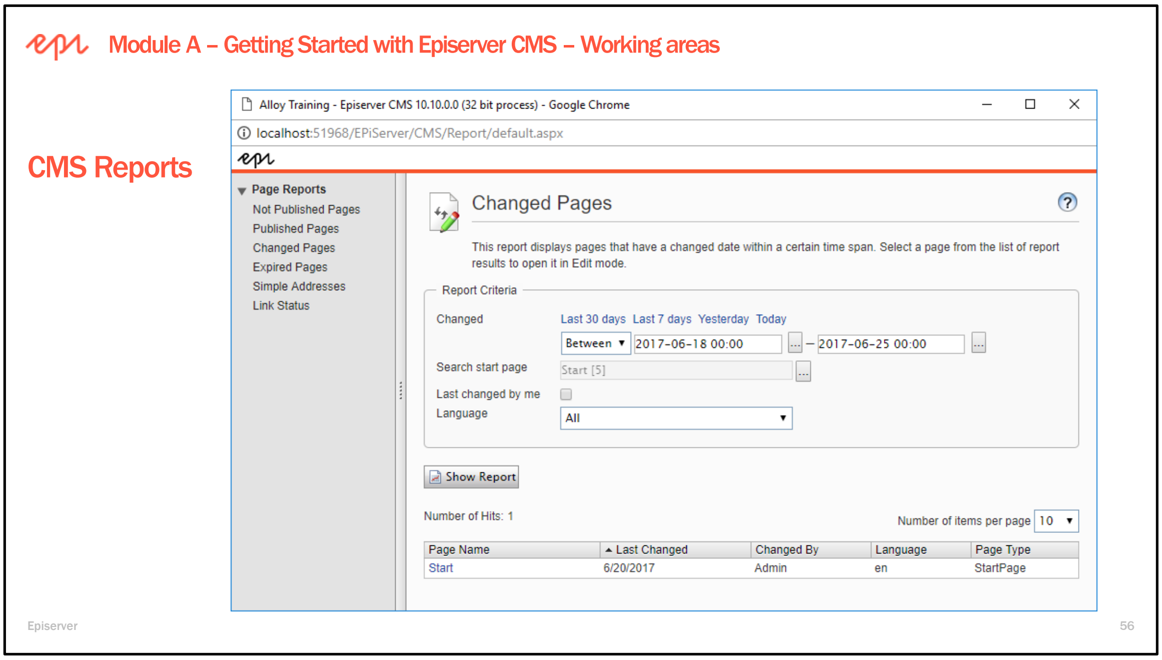
- CMS | Edit
  - Editors work in *Edit view* to manage content in sites.
- CMS | Admin
  - Administrators and developers work in *Admin view* to configure sites.
- CMS | Reports
  - A user interface that enables users to view standard or customized reports on Episerver content.
- CMS | Visitor Groups
  - A user interface for creating rules used for personalizing content.



Pin the top menu in the Episerver UI
https://world.episerver.com/blogs/David-Knipe/Dates/2017/12/pin-the-top-menu-in-the-episerver-ui/

Episerver 55

To learn how to build extensions to Episerver CMS, for example, Edit view gadgets, Admin view plug-ins, custom reports, and custom Visitor Group criteria, attend the *Episerver CMS Advanced Development* training course.



Module A – Getting Started with Episerver CMS – Working areas

CMS Reports

Alloy Training - Episerver CMS 10.10.0.0 (32 bit process) - Google Chrome

localhost:51968/EPiServer/CMS/Report/default.aspx

Changed Pages

This report displays pages that have a changed date within a certain time span. Select a page from the list of report results to open it in Edit mode.

Report Criteria

Changed	Last 30 days Last 7 days Yesterday Today
	Between 2017-06-18 00:00 – 2017-06-25 00:00
Search start page	Start [5]
Last changed by me	☐
Language	All

Number of Hits: 1

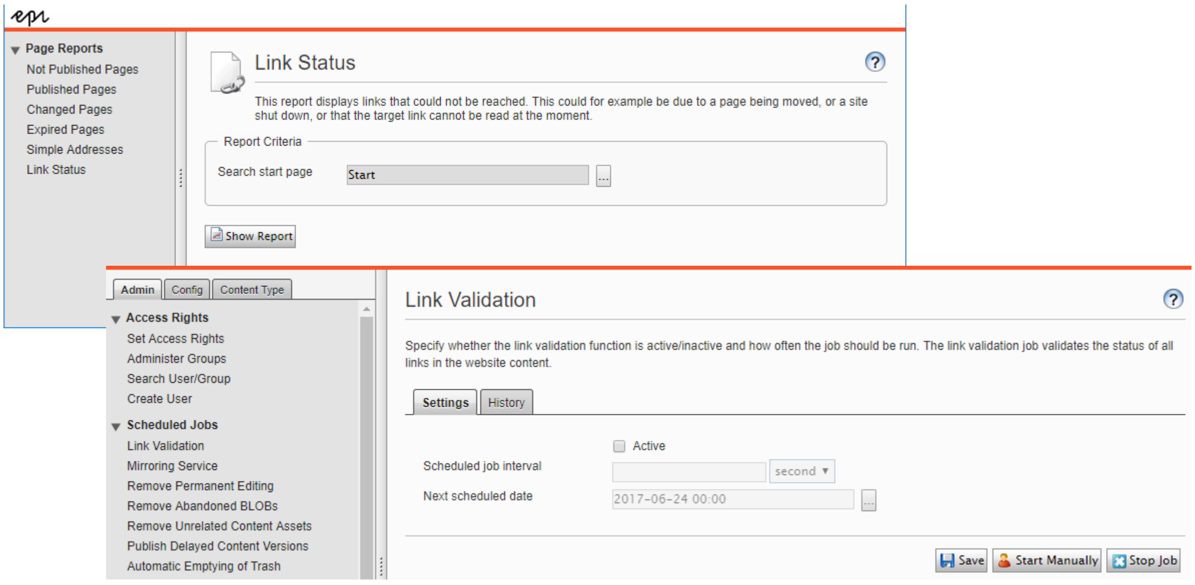
Number of items per page 10

Page Name	Last Changed	Changed By	Language	Page Type
Start	6/20/2017	Admin	en	StartPage

Episerver 56

Six predefined reports

1. Not Published Pages
2. Published Pages
3. Changed Pages
4. Expired Pages
5. Simple Addresses
6. Link Status: requires the **Link Validation** scheduled job to be executed.



Link Status

This report displays links that could not be reached. This could for example be due to a page being moved, or a site shut down, or that the target link cannot be read at the moment.

Report Criteria

Search start page: Start

Link Validation

Specify whether the link validation function is active/inactive and how often the job should be run. The link validation job validates the status of all links in the website content.

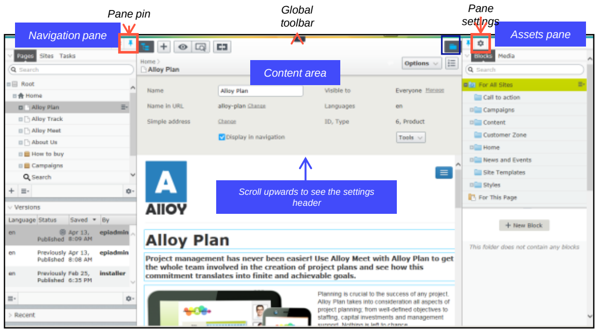
	☐ Active
Scheduled job interval	second
Next scheduled date	2017-06-24 00:00

You can define your own custom reports and add them to report center by using the **GuiPlugInAttribute**. Learn how on the *Episerver CMS Advanced Development* training course.

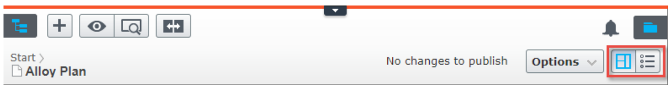
Module A – Getting Started with Episerver CMS – Working areas – Edit view

Edit view

- Editors work in Edit view to edit content and build the website structure.
- Pages can be edited directly on-page.
- Content can be edited when displayed in different resolutions and channels.
- All Properties view with sticky view mode
  - Access to all page properties, not only those visible on-page
  - Reached from Edit view and used by editors to manage advanced page settings



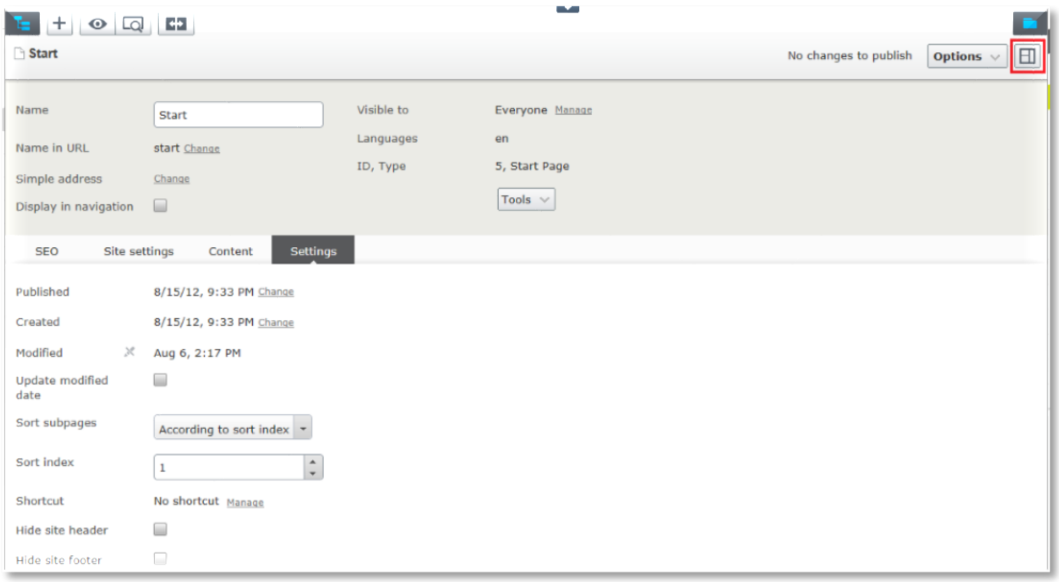
“Sticky” On-Page and All Properties views



Episerver 58

All Properties View is mainly used to work with the page settings and properties that are not accessible via the on-page edit view:

- Dates for Published and Created
- Sort order for child pages
- Shortcut
- Property types that do not have an on-page edit control



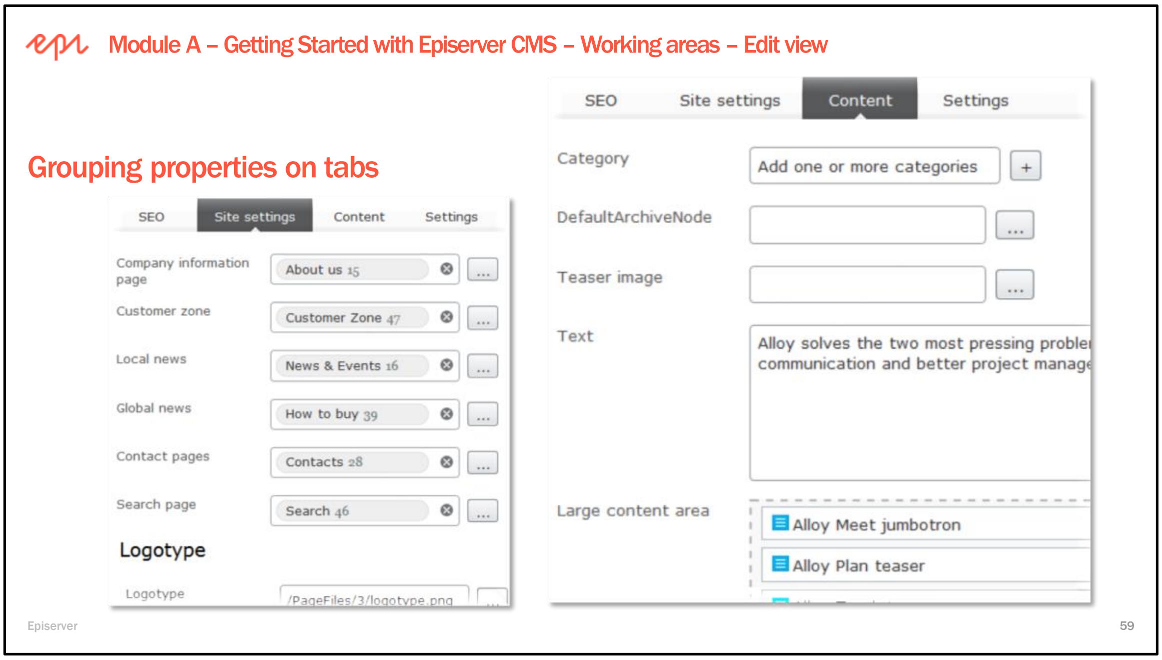
Start

No changes to publish Options

Name	Start	Visible to	Everyone Manage
Name in URL	start Change	Languages	en
Simple address	Change	ID, Type	5, Start Page
Display in navigation	☐		Tools

SEO | Site settings | Content | Settings

Published	8/15/12, 9:33 PM Change
Created	8/15/12, 9:33 PM Change
Modified	Aug 6, 2:17 PM
Update modified date	☐
Sort subpages	According to sort index
Sort index	1
Shortcut	No shortcut Manage
Hide site header	☐
Hide site footer	☐



Module A – Getting Started with Episerver CMS – Working areas – Edit view

Grouping properties on tabs

SEO | Site settings | Content | Settings

Company information page	About us 15
Customer zone	Customer Zone 47
Local news	News & Events 16
Global news	How to buy 39
Contact pages	Contacts 28
Search page	Search 46

Logotype

Logotype	/PageFiles/3/logotype.png

SEO | Site settings | Content | Settings

Category	Add one or more categories
DefaultArchiveNode	
Teaser image	
Text	Alloy solves the two most pressing proble communication and better project manage
Large content area	Alloy Meet jumbotron Alloy Plan teaser

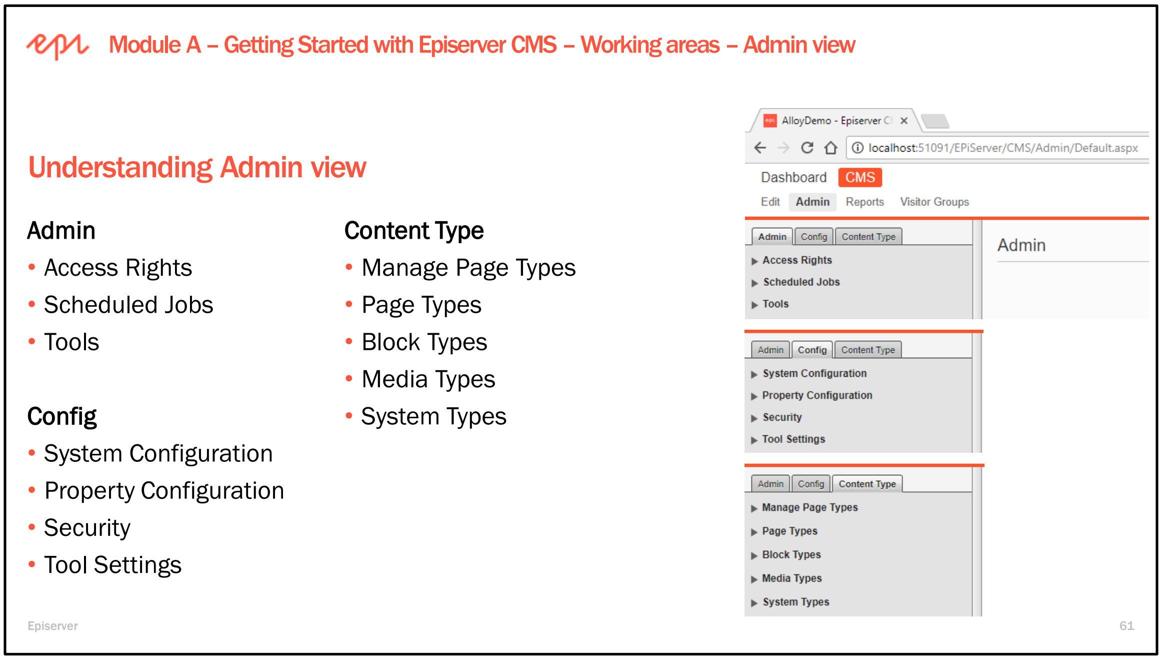
Episerver

59

Tabs are used to group properties in All Properties View
Grouping properties related to particular functionality on specific tabs makes it easier for Editors to find them. It also makes it easier for the Administrator or Developer to restrict access to specific properties if needed.

- Default tabs that are always available for a page type are Content and Settings.
- Tabs that don't contain properties are not shown.
- You set the access level on tabs via Admin view.
- You can create tabs from Admin view in the **Edit Tabs section** on the **Config** tab.
- You can specify custom GroupName/Tab string constants in a static class.
- You can control which tab a property is placed using code.

In the Alloy sample site a number of custom tabs (or GroupNames as they are called when working with them programmatically) have been added, for example SEO and Site Settings.
The GroupNames are specified as constants in Global.cs and translations for them can be found in the GroupNames.xml language resource file.



Module A – Getting Started with Episerver CMS – Working areas – Admin view

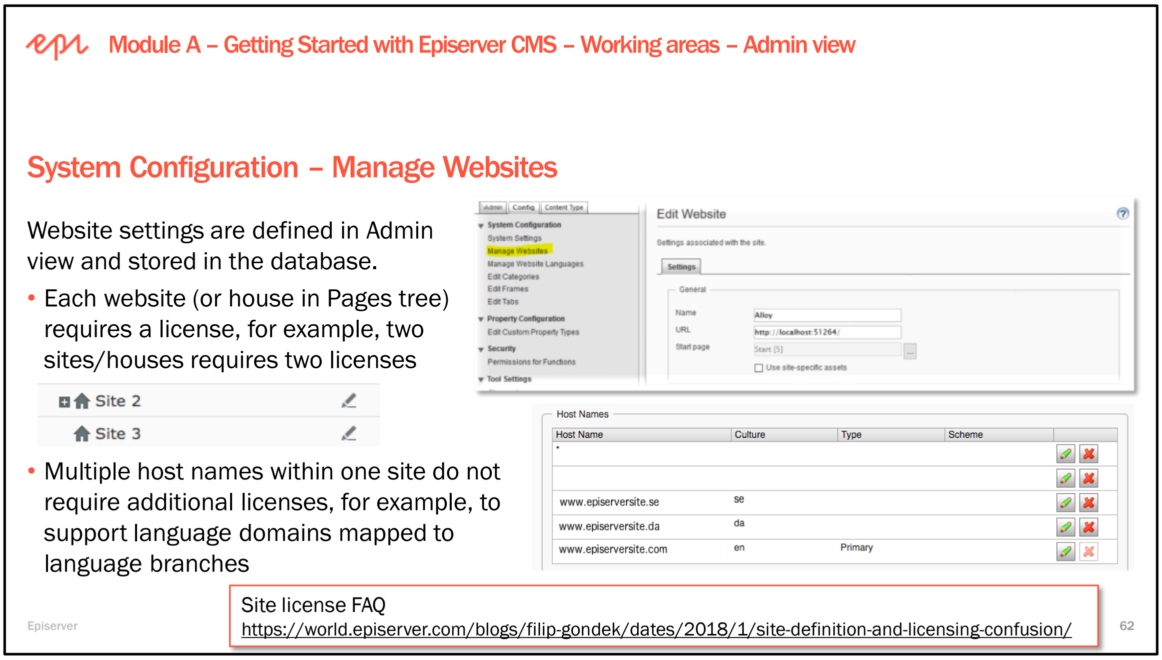
Understanding Admin view

Admin
- Access Rights
- Scheduled Jobs
- Tools

Config
- System Configuration
- Property Configuration
- Security
- Tool Settings

Content Type
- Manage Page Types
- Page Types
- Block Types
- Media Types
- System Types

Episerver 61



Module A – Getting Started with Episerver CMS – Working areas – Admin view

System Configuration – Manage Websites

Website settings are defined in Admin view and stored in the database.

- Each website (or house in Pages tree) requires a license, for example, two sites/houses requires two licenses

Site 2
Site 3

- Multiple host names within one site do not require additional licenses, for example, to support language domains mapped to language branches

Edit Website

Settings associated with the site.

General

Name	Alloy
URL	http://localhost:51264/
Start page	Start [5]

☐ Use site-specific assets

Host Names

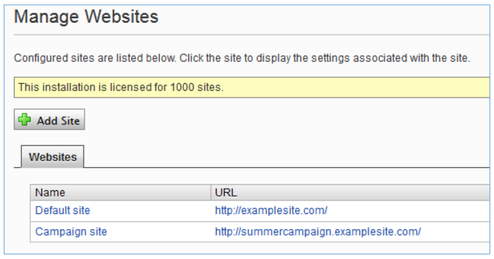
Host Name	Culture	Type	Scheme
*			
www.episerversite.se	se		
www.episerversite.da	da		
www.episerversite.com	en	Primary	

Site license FAQ
https://world.episerver.com/blogs/filip-gondek/dates/2018/1/site-definition-and-licensing-confusion/

Episerver 62

Episerver websites can be defined and managed from Admin. From version 7.5 of Episerver the setting values specific to a site are stored in the database instead of in the configuration files. Settings that are common for all sites have been moved to a new element applicationSettings in the <episerver> section in web.config. Several Episerver sites can use the same IIS site. If the IIS site is configured to have a wildcard host, new sites can be added to an existing solution from the CMS Admin without any additional configuration needed.
The above example is of a single-site configuration.
References

- The Manage Websites section in the Episerver Web Help
- The Deployment section in the Developer Guide for CMS
- Johan Björnfot's blog on Episerver World: http://world.episerver.com/Blogs/Johan-Bjornfot/Dates1/2013/12/Multisite-feature-in-Episerver-75/

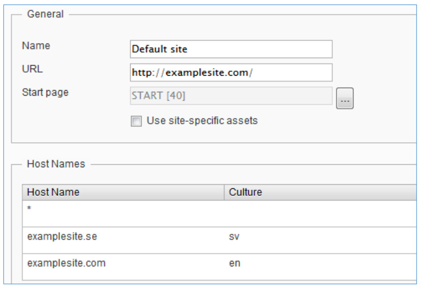


Manage Websites

Configured sites are listed below. Click the site to display the settings associated with the site.

This installation is licensed for 1000 sites.

Name	URL
Default site	http://examplesite.com/
Campaign site	http://summercampaign.examplesite.com/



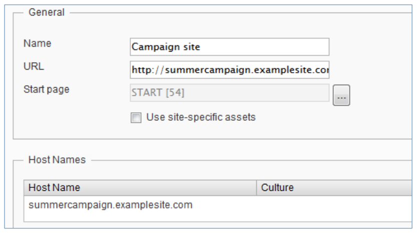
General

Name	Default site
URL	http://examplesite.com/
Start page	START [40]

☐ Use site-specific assets

Host Names

Host Name	Culture
*	
examplesite.se	sv
examplesite.com	en



General

Name	Campaign site
URL	http://summercampaign.examplesite.co
Start page	START [54]

☐ Use site-specific assets

Host Names

Host Name	Culture
summercampaign.examplesite.com	

Module A – Getting Started with Episerver CMS – Working areas – Admin view

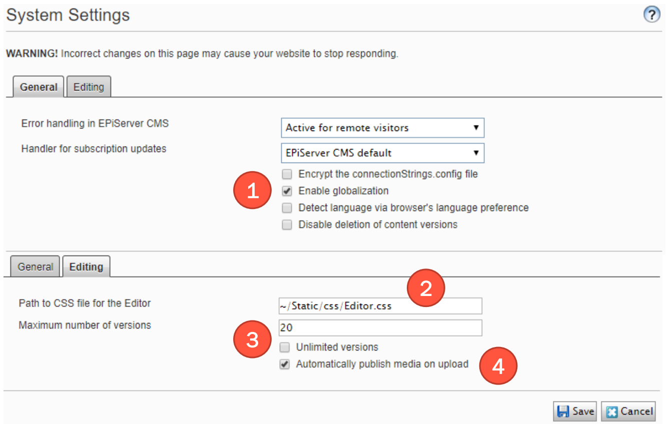
System Configuration – System Settings

General

1. Language support

Editing

2. TinyMCE style list customization
3. Content version limits
4. Media publishing



System Settings

WARNING! Incorrect changes on this page may cause your website to stop responding.

General | Editing

Error handling in EPiServer CMS: Active for remote visitors

Handler for subscription updates: EPiServer CMS default

1
- [ ] Encrypt the connectionStrings.config file
- [x] Enable globalization
- [ ] Detect language via browser's language preference
- [ ] Disable deletion of content versions

General | Editing

2

Path to CSS file for the Editor: ~/Static/css/Editor.css

3

Maximum number of versions: 20

- [ ] Unlimited versions
- [x] Automatically publish media on upload

4

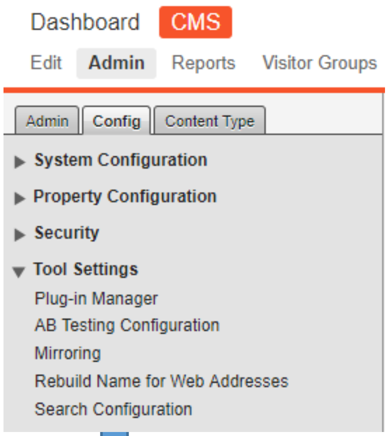
Save | Cancel

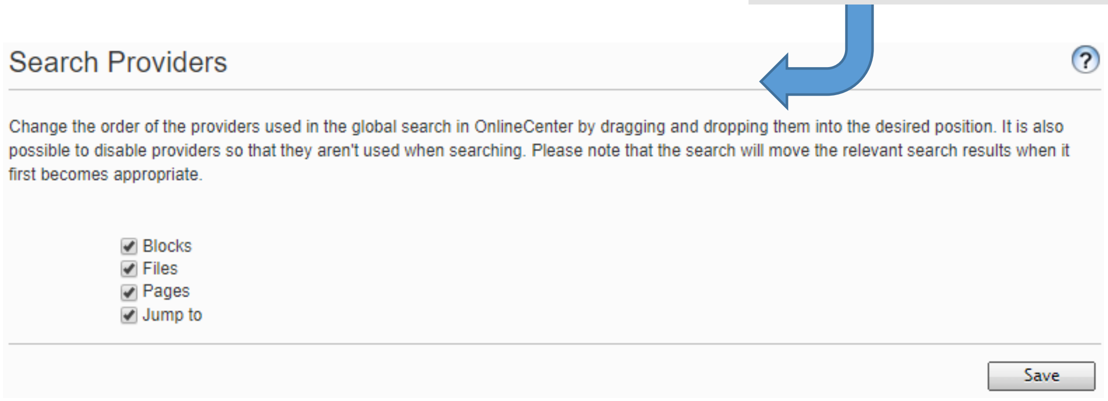
Episerver 63

Tool Settings – Search Configuration

You can install multiple search providers at the same time, and configure the order in which search results are shown, and disable content types for different providers:

- Episerver Search
- Episerver Find
- Others



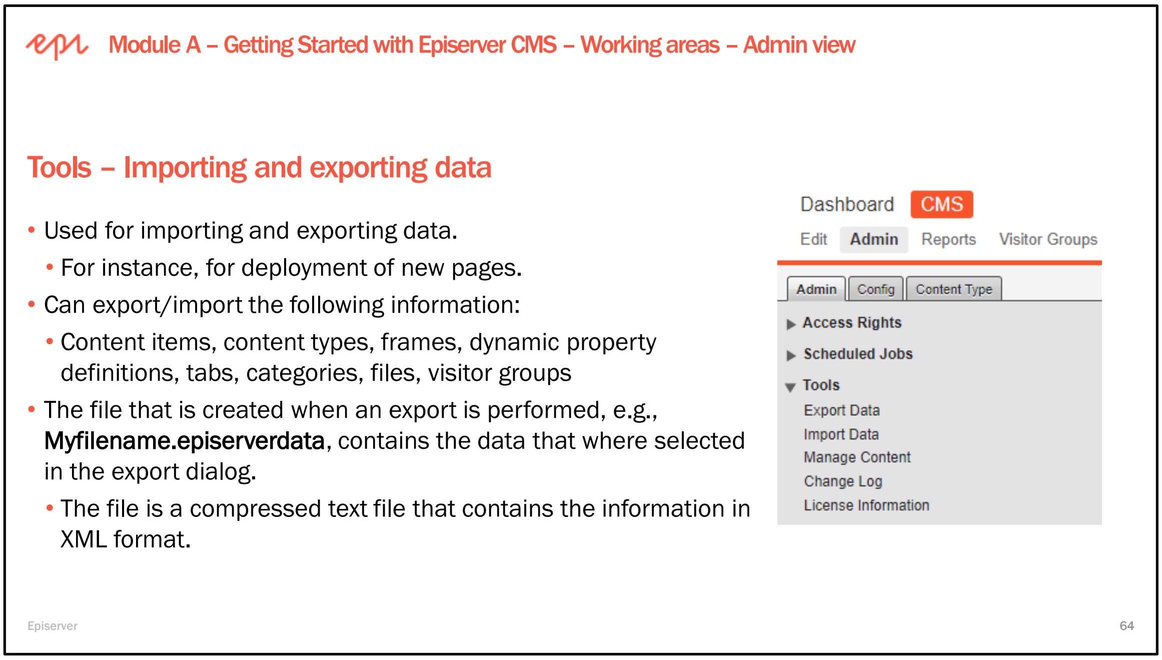


Search Providers

Change the order of the providers used in the global search in OnlineCenter by dragging and dropping them into the desired position. It is also possible to disable providers so that they aren't used when searching. Please note that the search will move the relevant search results when it first becomes appropriate.

- [x] Blocks
- [x] Files
- [x] Pages
- [x] Jump to

Save



Module A – Getting Started with Episerver CMS – Working areas – Admin view

Tools – Importing and exporting data

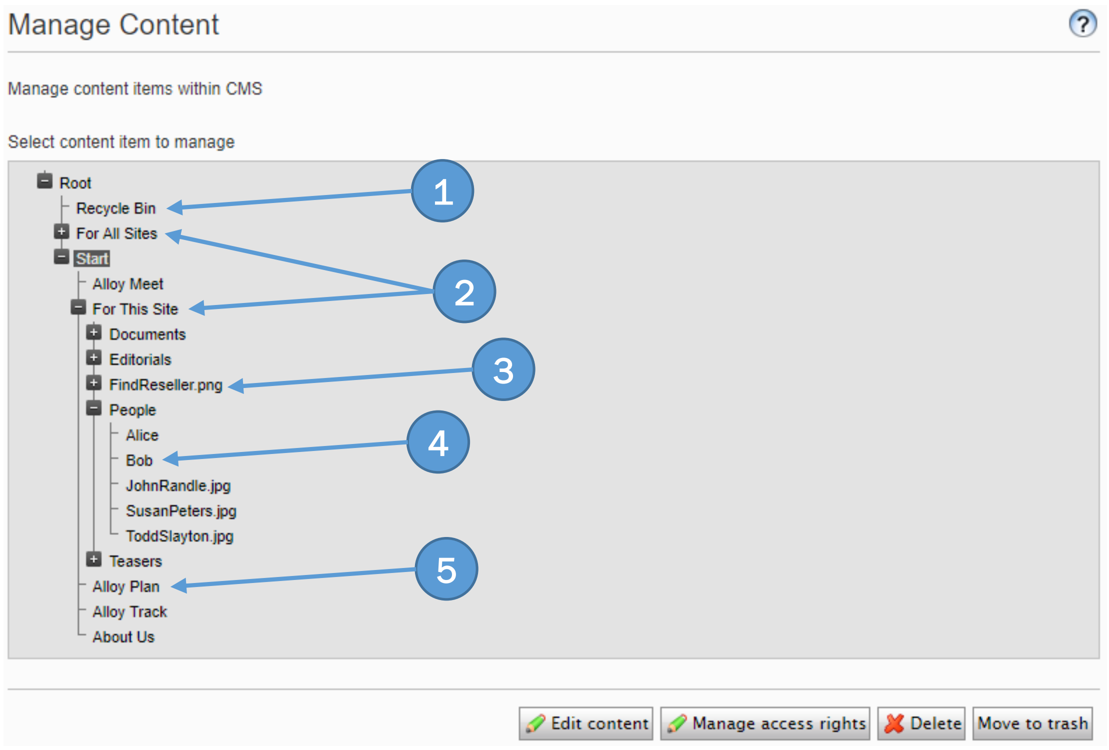
- Used for importing and exporting data.
  - For instance, for deployment of new pages.
- Can export/import the following information:
  - Content items, content types, frames, dynamic property definitions, tabs, categories, files, visitor groups
- The file that is created when an export is performed, e.g., **Myfilename.episerverdata**, contains the data that where selected in the export dialog.
  - The file is a compressed text file that contains the information in XML format.

Episerver 64

Tools – Managing content

A useful view of the content tree, showing its hierarchical structure, combining pages, blocks, folders, and media assets, with buttons to quickly edit or assign access rights:

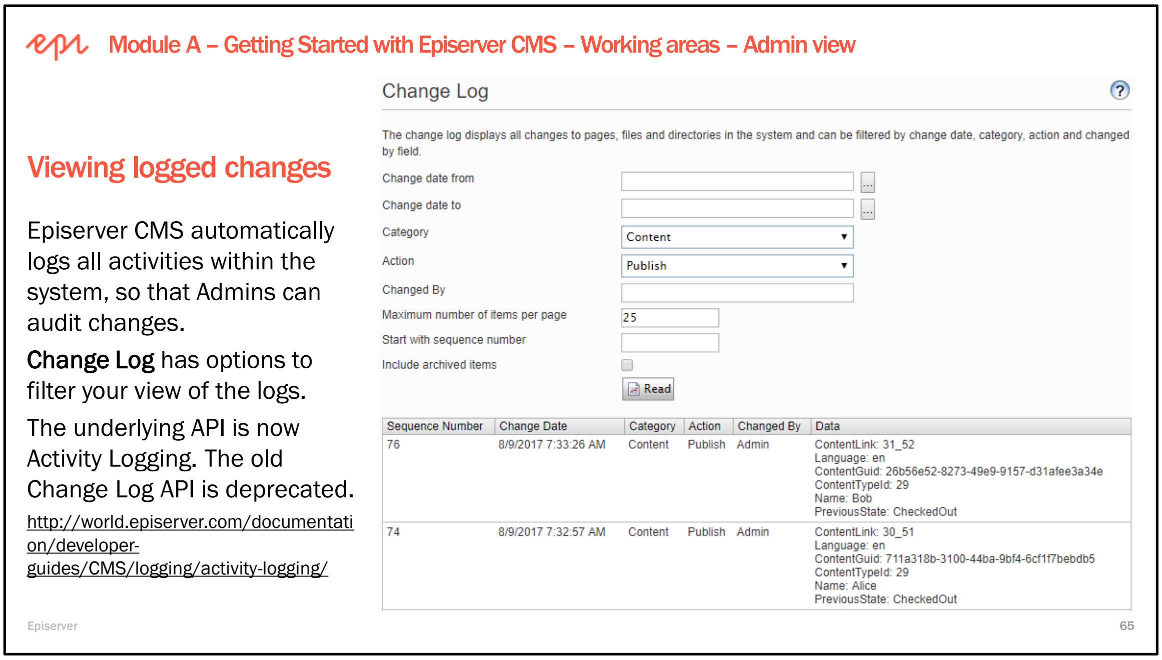
1. System objects like Recycle Bin
2. Folders like For All Sites and For This Site
3. Media assets like FindReseller.png
4. Blocks like Bob
5. Pages like Alloy Plan



Manage Content

Manage content items within CMS

Select content item to manage



Module A – Getting Started with Episerver CMS – Working areas – Admin view

Viewing logged changes

Episerver CMS automatically logs all activities within the system, so that Admins can audit changes.

Change Log has options to filter your view of the logs.

The underlying API is now Activity Logging. The old Change Log API is deprecated.

http://world.episerver.com/documentation/developer-guides/CMS/logging/activity-logging/

Change Log

The change log displays all changes to pages, files and directories in the system and can be filtered by change date, category, action and changed by field.

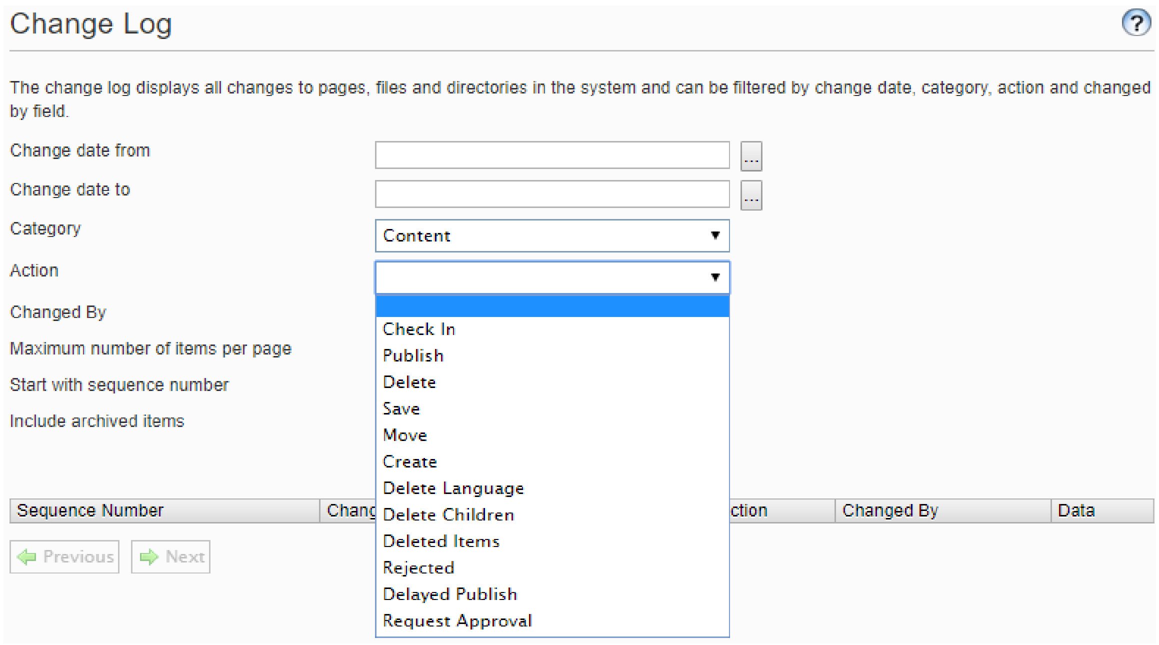
Change date from	
Change date to	
Category	Content
Action	Publish
Changed By	
Maximum number of items per page	25
Start with sequence number	
Include archived items	☐

Read

Sequence Number	Change Date	Category	Action	Changed By	Data
76	8/9/2017 7:33:26 AM	Content	Publish	Admin	ContentLink: 31_52 Language: en ContentGuid: 26b56e52-8273-49e9-9157-d31afee3a34e ContentTypeId: 29 Name: Bob PreviousState: CheckedOut
74	8/9/2017 7:32:57 AM	Content	Publish	Admin	ContentLink: 30_51 Language: en ContentGuid: 711a318b-3100-44ba-9bf4-6cf1f7bebdb5 ContentTypeId: 29 Name: Alice PreviousState: CheckedOut

Episerver 65

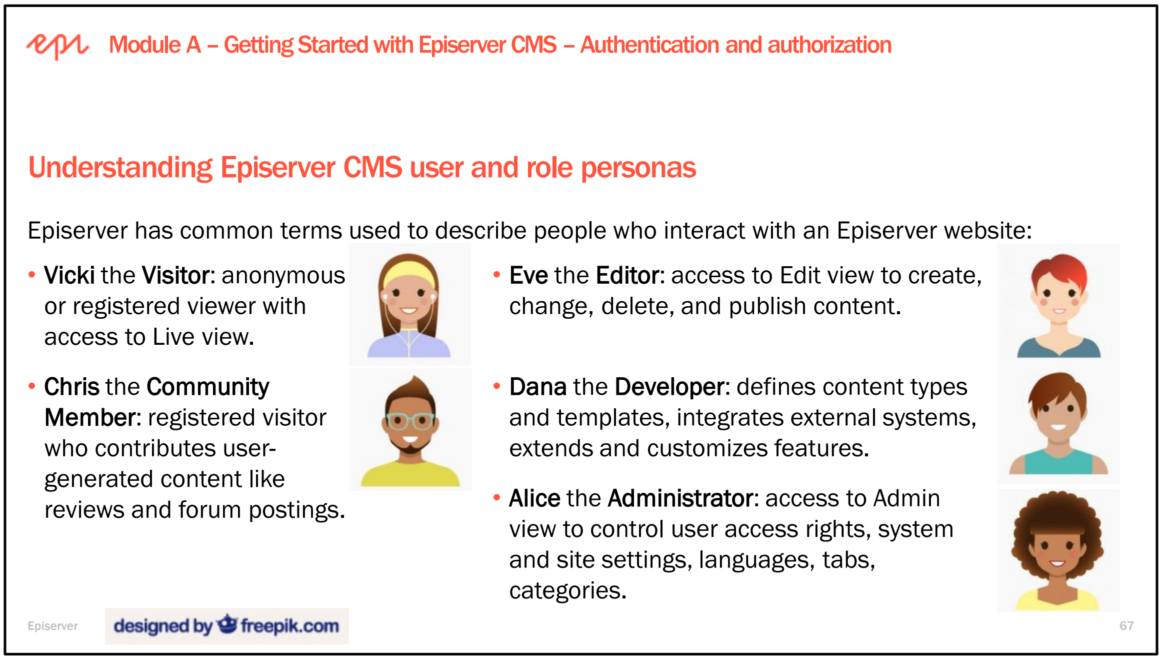
Change Log can filter content by the actions shown in the following screenshot:



Change Log

The change log displays all changes to pages, files and directories in the system and can be filtered by change date, category, action and changed by field.

Change date from	
Change date to	
Category	Content
Action	
Changed By	
Maximum number of items per page	
Start with sequence number	
Include archived items	



Module A – Getting Started with Episerver CMS – Authentication and authorization

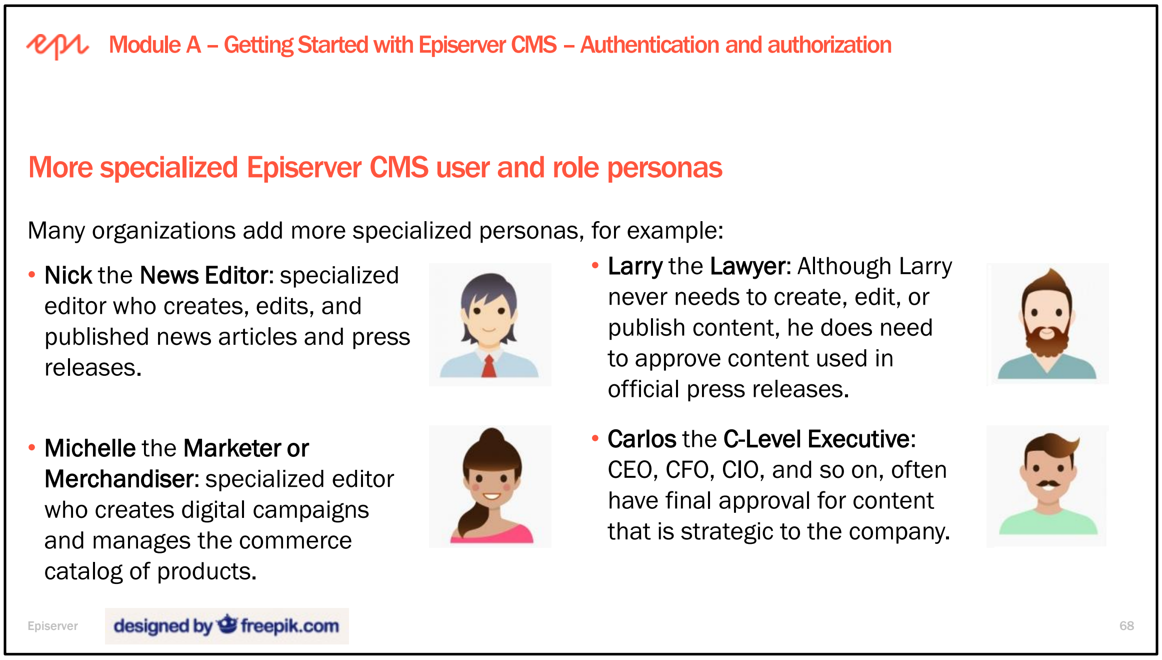
Understanding Episerver CMS user and role personas

Episerver has common terms used to describe people who interact with an Episerver website:

- **Vicki** the **Visitor**: anonymous or registered viewer with access to Live view.
- **Chris** the **Community Member**: registered visitor who contributes user-generated content like reviews and forum postings.
- **Eve** the **Editor**: access to Edit view to create, change, delete, and publish content.
- **Dana** the **Developer**: defines content types and templates, integrates external systems, extends and customizes features.
- **Alice** the **Administrator**: access to Admin view to control user access rights, system and site settings, languages, tabs, categories.

Episerver designed by freepik.com 67

Images created by Freepik:
http://www.freepik.com/free-vector/nice-people-avatars-in-flat-design_844761.htm



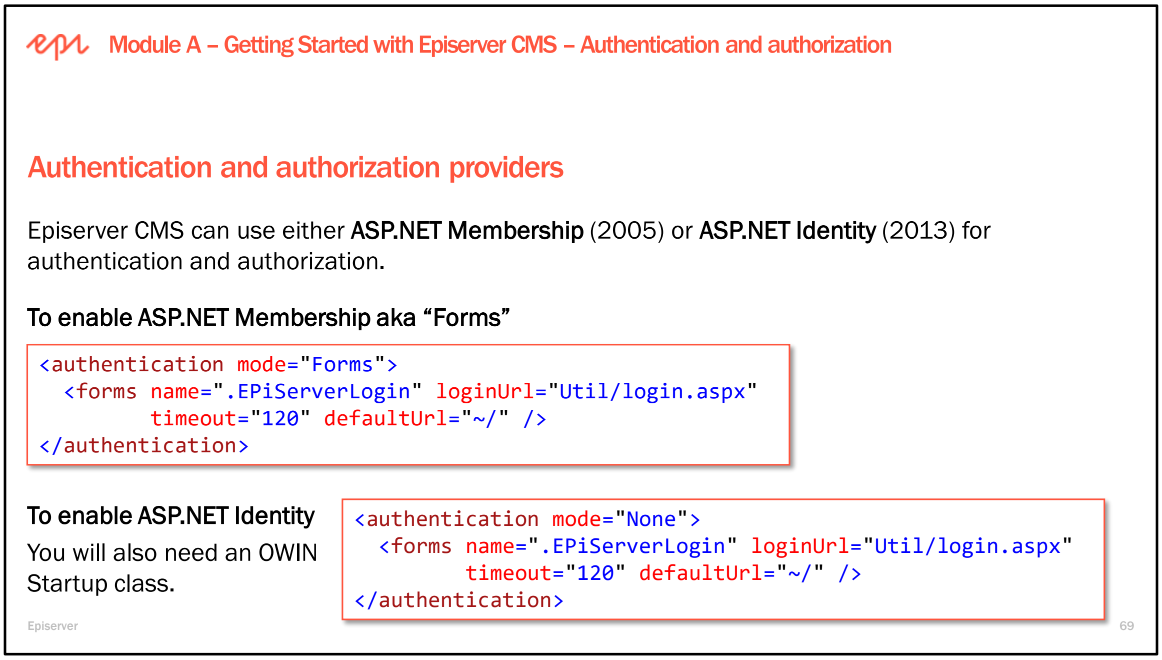
More specialized Episerver CMS user and role personas

Many organizations add more specialized personas, for example:

- **Nick** the **News Editor**: specialized editor who creates, edits, and published news articles and press releases.
- **Michelle** the **Marketer or Merchandiser**: specialized editor who creates digital campaigns and manages the commerce catalog of products.
- **Larry** the **Lawyer**: Although Larry never needs to create, edit, or publish content, he does need to approve content used in official press releases.
- **Carlos** the **C-Level Executive**: CEO, CFO, CIO, and so on, often have final approval for content that is strategic to the company.

Episerver designed by freepik.com 68

Images created by Freepik:
http://www.freepik.com/free-vector/nice-people-avatars-in-flat-design_844761.htm



Module A – Getting Started with Episerver CMS – Authentication and authorization

Authentication and authorization providers

Episerver CMS can use either **ASP.NET Membership** (2005) or **ASP.NET Identity** (2013) for authentication and authorization.

To enable ASP.NET Membership aka "Forms"

```
<authentication mode="Forms">
  <forms name=".EPiServerLogin" loginUrl="Util/login.aspx"
         timeout="120" defaultUrl="~/" />
</authentication>
```

To enable ASP.NET Identity

You will also need an OWIN Startup class.

```
<authentication mode="None">
  <forms name=".EPiServerLogin" loginUrl="Util/login.aspx"
         timeout="120" defaultUrl="~/" />
</authentication>
```

Episerver 69

Terminology

Authentication = "Who": The process of identifying a user. The usual way of doing this is with a username and a password.

- Membership provider. The module that handles authentication in the security model in ASP.NET.

Authorization = "What": The process of determining the specific actions a user is allowed to perform.

- Role provider. The module that gives the base data for authorization in the security model in ASP.NET.

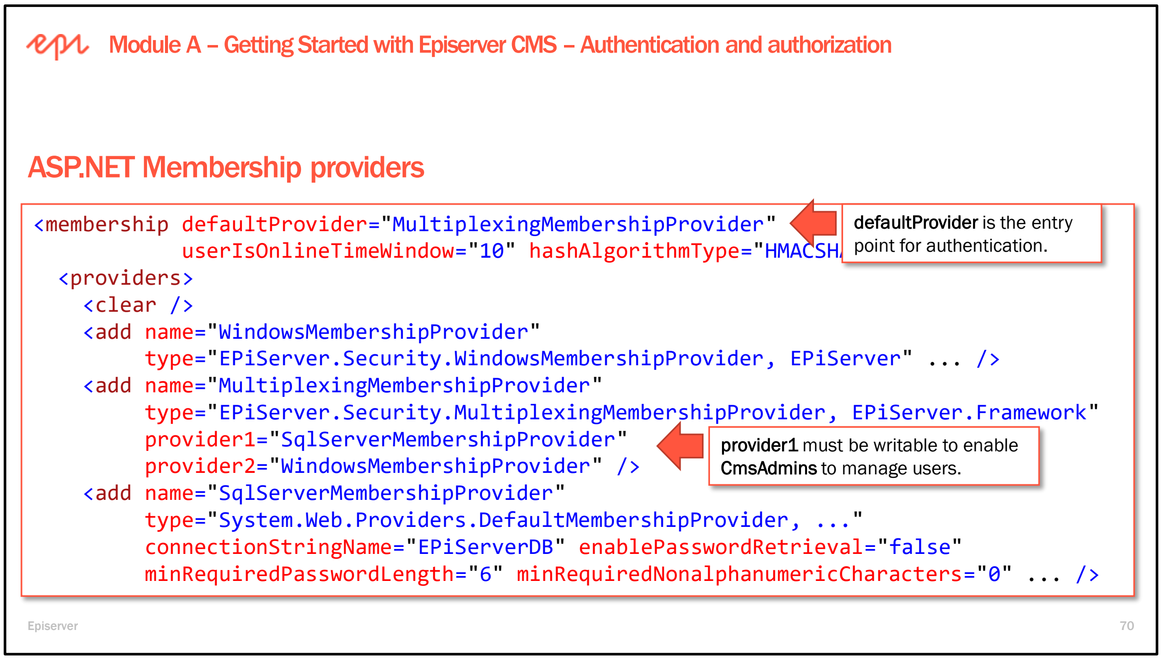
How to configure Episerver to use Active Directory

https://josefottosson.se/how-to-configure-episerver-to-use-active-directory/

EPiServer CMS UI AspNetIdentity OWIN authentication

You can configure the application to use EPiServer AspNetIdentity as the authentication module for managing users and roles. This configuration requires the following NuGet package as a dependency: EPiServer.CMS.UI.AspNetIdentity.

https://world.episerver.com/documentation/developer-guides/CMS/security/episerver-aspnetidentity/



Module A – Getting Started with Episerver CMS – Authentication and authorization

ASP.NET Membership providers

```
<membership defaultProvider="MultiplexingMembershipProvider"
            userIsOnlineTimeWindow="10" hashAlgorithmType="HMACSH
  <providers>
    <clear />
    <add name="WindowsMembershipProvider"
         type="EPiServer.Security.WindowsMembershipProvider, EPiServer" ... />
    <add name="MultiplexingMembershipProvider"
         type="EPiServer.Security.MultiplexingMembershipProvider, EPiServer.Framework"
         provider1="SqlServerMembershipProvider"
         provider2="WindowsMembershipProvider" />
    <add name="SqlServerMembershipProvider"
         type="System.Web.Providers.DefaultMembershipProvider, ..."
         connectionStringName="EPiServerDB" enablePasswordRetrieval="false"
         minRequiredPasswordLength="6" minRequiredNonalphanumericCharacters="0" ... />
```

defaultProvider is the entry point for authentication.

provider1 must be writable to enable **CmsAdmins** to manage users.

Episerver 70

```
<roleManager enabled="true" defaultProvider="MultiplexingRoleProvider"
             cacheRolesInCookie="true">
  <providers>
    <clear />
    <add name="MultiplexingRoleProvider"
         type="EPiServer.Security.MultiplexingRoleProvider, EPiServer.Framework"
         provider1="SqlServerRoleProvider"
         provider2="WindowsRoleProvider"
         providerMap1="SqlServerMembershipProvider"
         providerMap2="WindowsMembershipProvider" />
    <add name="WindowsRoleProvider" applicationName="/"
         type="EPiServer.Security.WindowsRoleProvider, EPiServer" />
    <add name="SqlServerRoleProvider"
         type="System.Web.Providers.DefaultRoleProvider, ..."
         connectionStringName="EPiServerDB" applicationName="/" />
  </providers>
</roleManager>
```

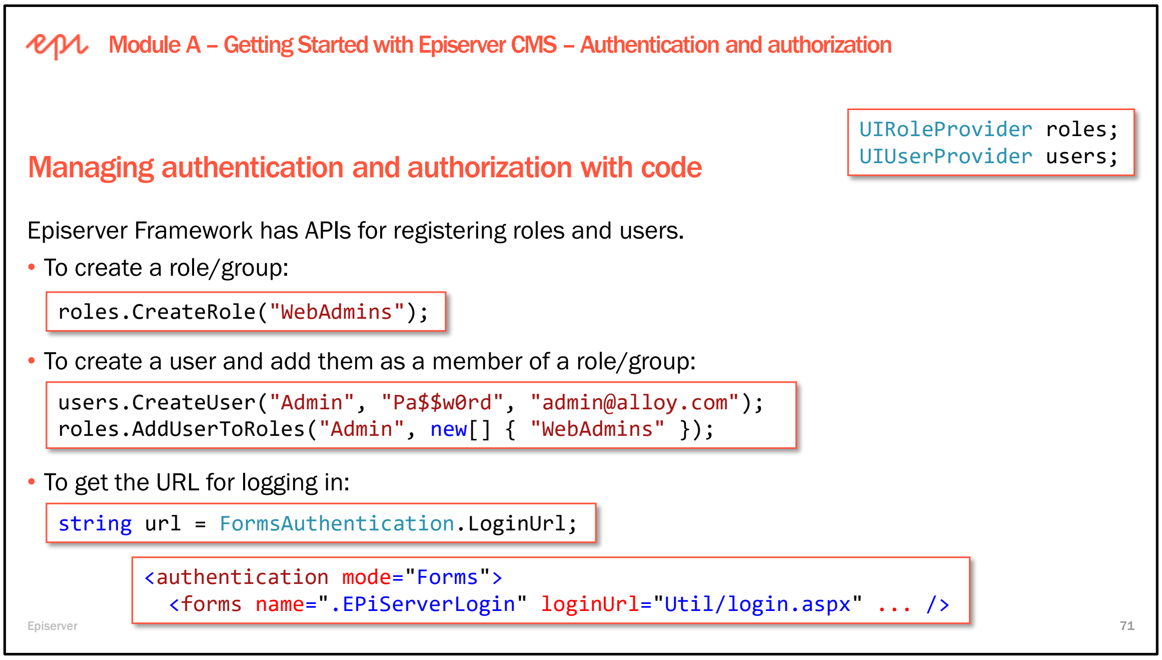
The **Empty** project template configures the **MultiplexingMembershipProvider** to use:

- **SqlMembershipProvider** as **provider1**, and
- **WindowsMembershipProvider** as **provider2**

...because provider1 must support read/write if the Admins need to be able to manage users and roles through the Episerver UI.

The SQL provider is read/write. The Windows provider is read-only.

You can configure additional or alternative providers including **Active Directory** and **ASP.NET Identity**
http://world.episerver.com/documentation/developer-guides/CMS/security/episerver-aspnetidentity/



Module A – Getting Started with Episerver CMS – Authentication and authorization

```
UIRoleProvider roles;
UIUserProvider users;
```

Managing authentication and authorization with code

Episerver Framework has APIs for registering roles and users.

- To create a role/group:

```
roles.CreateRole("WebAdmins");
```

- To create a user and add them as a member of a role/group:

```
users.CreateUser("Admin", "Pa$$w0rd", "admin@alloy.com");
roles.AddUserToRoles("Admin", new[] { "WebAdmins" });
```

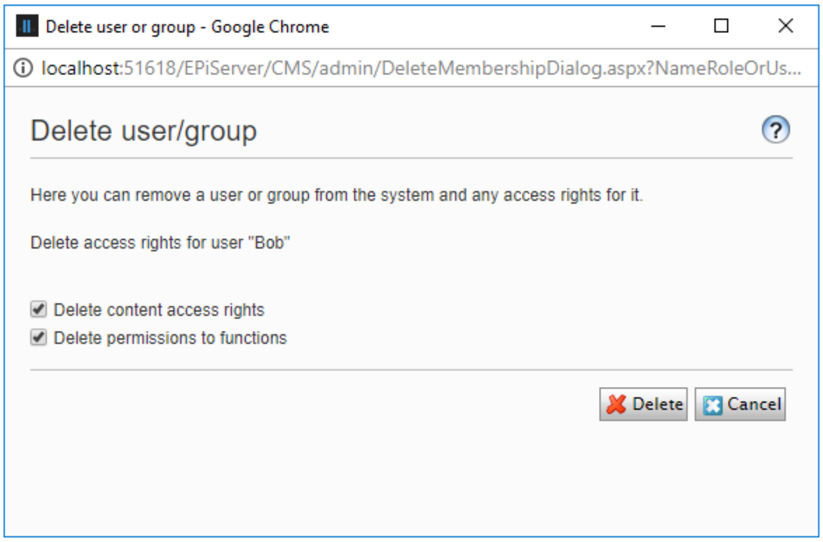
- To get the URL for logging in:

```
string url = FormsAuthentication.LoginUrl;
```

```
<authentication mode="Forms">
  <forms name=".EPiServerLogin" loginUrl="Util/login.aspx" ... />
```

Episerver 71

When you delete a group using the Admin view:



Delete user or group - Google Chrome

localhost:51618/EPiServer/CMS/admin/DeleteMembershipDialog.aspx?NameRoleOrUs...

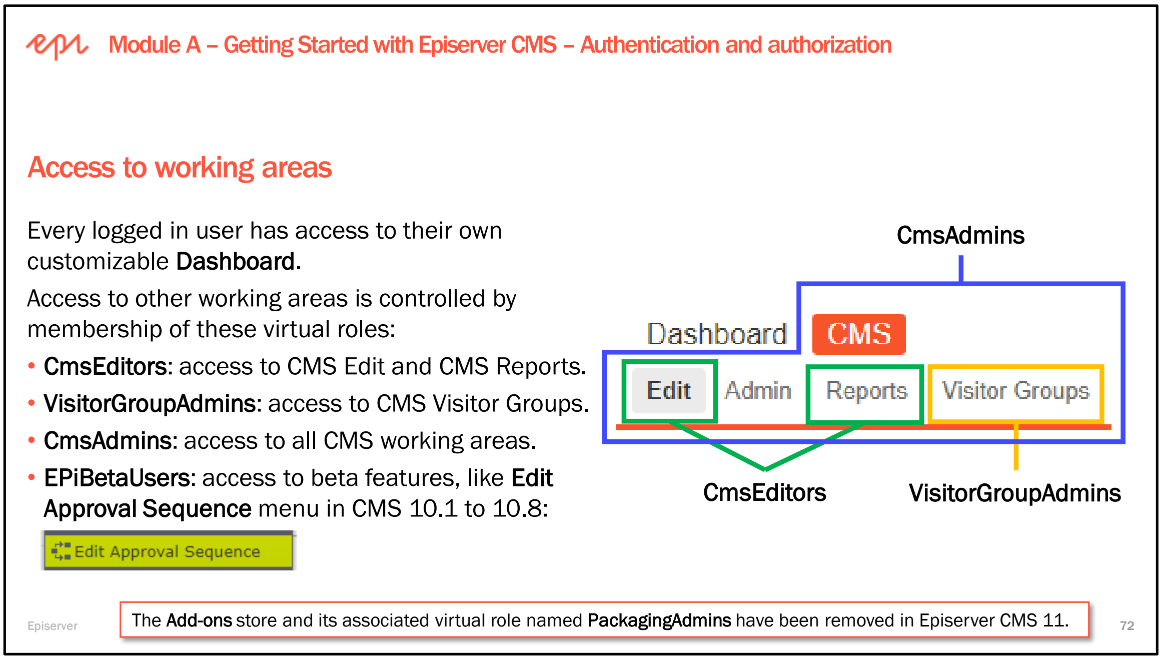
Delete user/group

Here you can remove a user or group from the system and any access rights for it.

Delete access rights for user "Bob"

- [✓] Delete content access rights
- [✓] Delete permissions to functions

Create Episerver admin user by code
https://world.episerver.com/blogs/kristoffer-linden/dates/2017/12/create-episerver-login-account-by-code/



Module A – Getting Started with Episerver CMS – Authentication and authorization

Access to working areas

Every logged in user has access to their own customizable **Dashboard.**

Access to other working areas is controlled by membership of these virtual roles:

- **CmsEditors**: access to CMS Edit and CMS Reports.
- **VisitorGroupAdmins**: access to CMS Visitor Groups.
- **CmsAdmins**: access to all CMS working areas.
- **EPiBetaUsers**: access to beta features, like **Edit Approval Sequence** menu in CMS 10.1 to 10.8:

The **Add-ons** store and its associated virtual role named **PackagingAdmins** have been removed in Episerver CMS 11.

Episerver 72

Privileges for the access groups in a default installation
All access groups can log in and access Reports
CmsEditors (usually WebEditors)

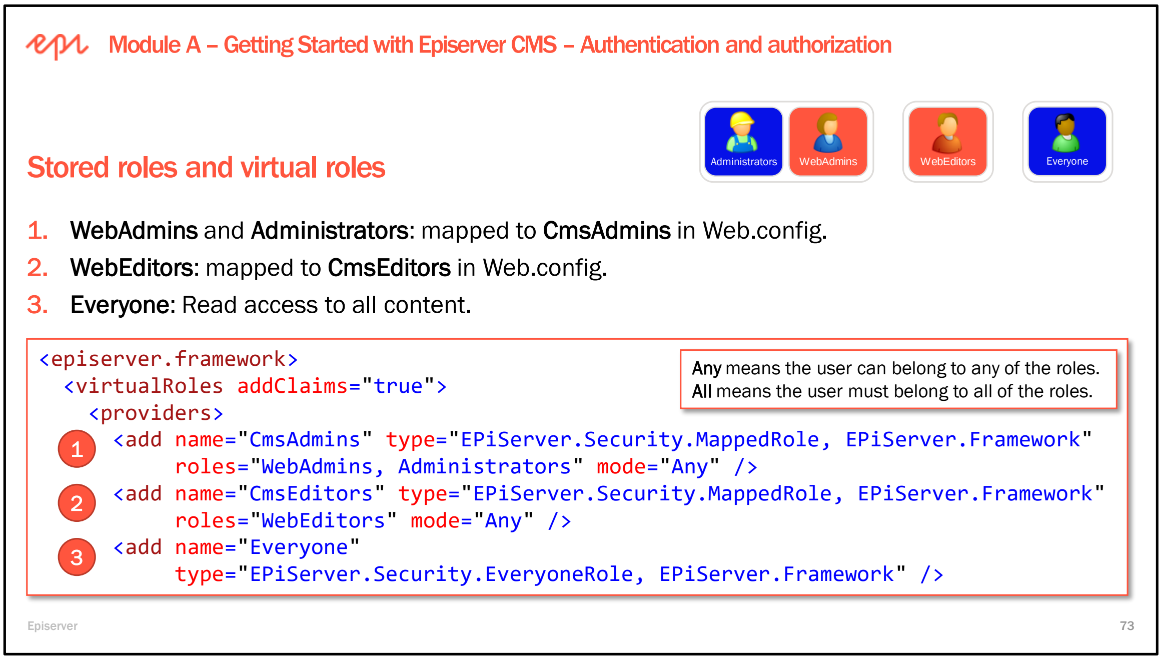
- Can access Edit view and Reports.
- Create, change and publish pages and blocks in Edit view.
- Can give other users access rights in the website structure where Administer access level has been granted.

VisitorGroupAdmins

- Can access the Visitor Groups UI and administer visitor groups.
- Was added to make it possible to give Editors access to Visitor Groups without giving access to the rest of Admin.

CmsAdmins (usually WebAdmins and Administrators)

- Can access Edit, Admin, Reports, Visitor Groups
- Maintain the users and groups for the whole website.
- Set access rights on pages, page types and languages.
- Set access rights on files, folders and blocks in the Assets pane.


Administrators
WebAdmins
WebEditors
Everyone

Module A – Getting Started with Episerver CMS – Authentication and authorization

Stored roles and virtual roles

1. **WebAdmins** and **Administrators**: mapped to **CmsAdmins** in Web.config.
2. **WebEditors**: mapped to **CmsEditors** in Web.config.
3. **Everyone**: Read access to all content.

Any means the user can belong to any of the roles.
All means the user must belong to all of the roles.

```
<episerver.framework>
  <virtualRoles addClaims="true">
    <providers>
1     <add name="CmsAdmins" type="EPiServer.Security.MappedRole, EPiServer.Framework"
           roles="WebAdmins, Administrators" mode="Any" />
2     <add name="CmsEditors" type="EPiServer.Security.MappedRole, EPiServer.Framework"
           roles="WebEditors" mode="Any" />
3     <add name="Everyone"
           type="EPiServer.Security.EveryoneRole, EPiServer.Framework" />
```

Episerver 73

Although the Web.config created by the project template maps Administrators (usually a Windows group) and WebAdmins (usually a SQL-stored role) to **CmsAdmins**, you should not assume this to always be the case. Therefore, always use the virtual role names when applying authorization rules.

For example, when setting access rights, apply them to **CmsEditors**, not **WebEditors**.

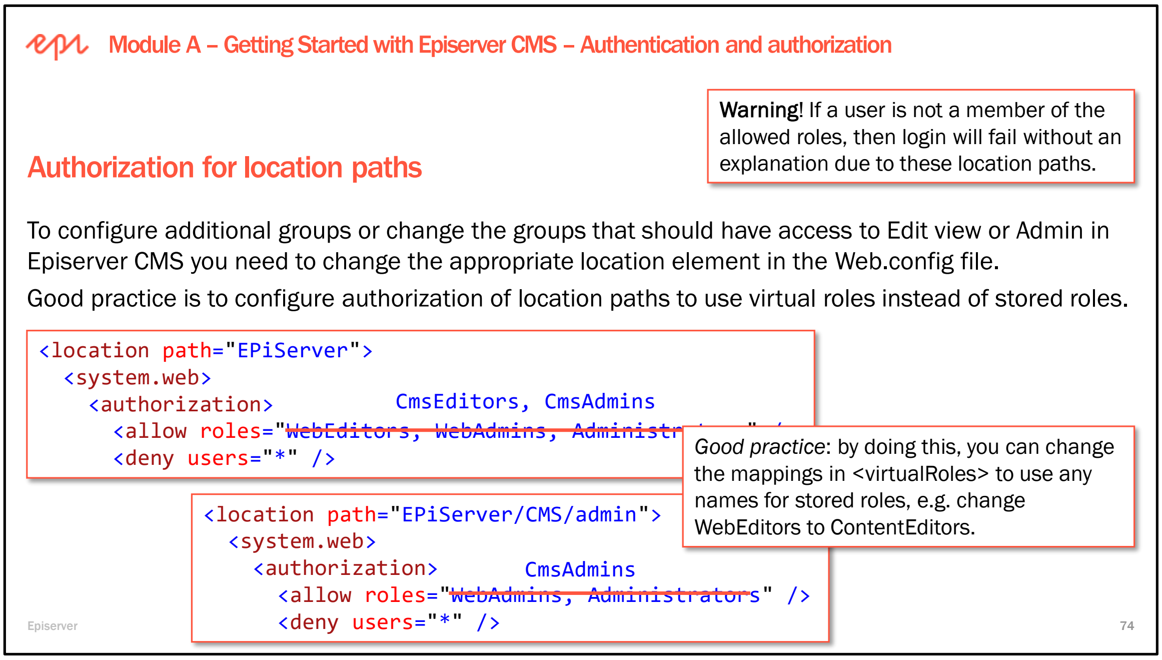
The following virtual roles are delivered with Episerver CMS:

- Anonymous
- Authenticated
- Creator
- Everyone
- Administrator
- CmsAdmins
- CmsEditors

Predefined virtual roles in Episerver CMS that are not pre-configured but worth knowing about:

- EPiBetaUsers (gives access to beta features)
- VisitorGroupAdmins (gives access to the Visitor Groups UI)

In addition to the predefined roles, it is very easy to create new virtual roles to allow access based on business rules, such as only allow access during business hours. A common scenario is to define virtual roles that evaluate to true if the user is a member of role1 and role2. This can be used to reduce the number of groups needed for setting the required permissions in Episerver CMS.



Module A – Getting Started with Episerver CMS – Authentication and authorization

Warning! If a user is not a member of the allowed roles, then login will fail without an explanation due to these location paths.

Authorization for location paths

To configure additional groups or change the groups that should have access to Edit view or Admin in Episerver CMS you need to change the appropriate location element in the Web.config file.

Good practice is to configure authorization of location paths to use virtual roles instead of stored roles.

```
<location path="EPiServer">
  <system.web>
    <authorization>           CmsEditors, CmsAdmins
      <allow roles="~~WebEditors, WebAdmins, Administr~~
      <deny users="*" />
```

Good practice: by doing this, you can change the mappings in <virtualRoles> to use any names for stored roles, e.g. change WebEditors to ContentEditors.

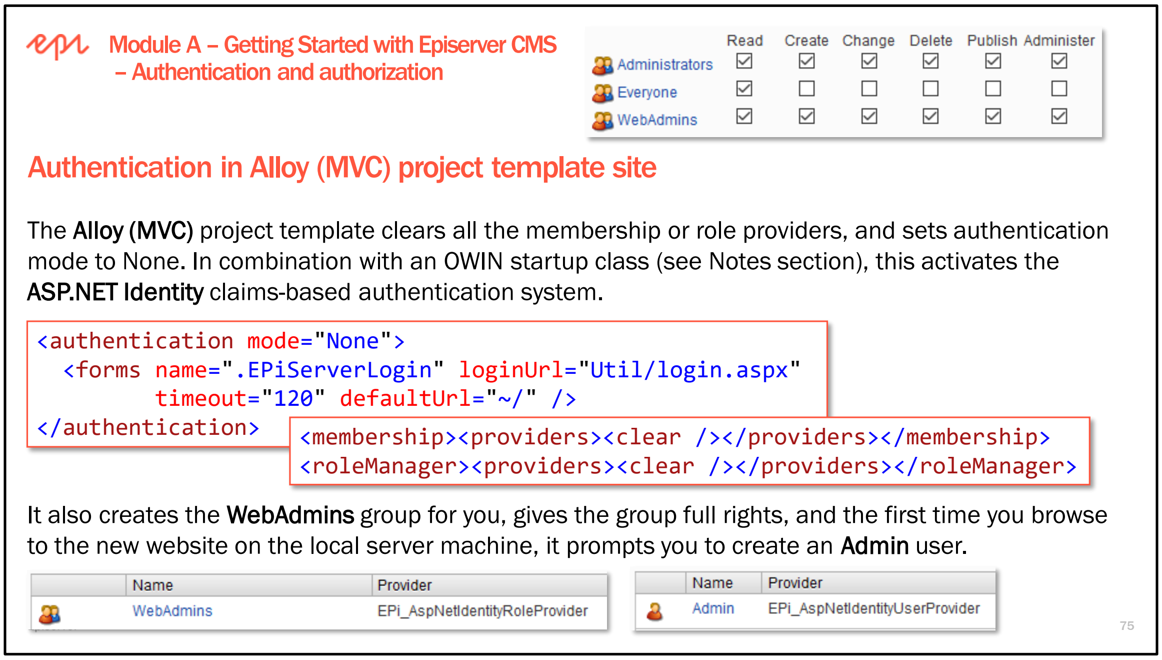
```
<location path="EPiServer/CMS/admin">
  <system.web>
    <authorization>       CmsAdmins
      <allow roles="~~WebAdmins, Administrators~~" />
      <deny users="*" />
```

Episerver

74

In a default installation of Episerver CMS there are preconfigured groups in Web.config that need to correspond to the groups created in Admin.

The groups WebEditors, WebAdmins and VisitorGroupAdmins must be created in Admin before you can give other users access to Edit view and/or Admin.



Module A – Getting Started with Episerver CMS – Authentication and authorization

	Read	Create	Change	Delete	Publish	Administer
Administrators	☑	☑	☑	☑	☑	☑
Everyone	☑	☐	☐	☐	☐	☐
WebAdmins	☑	☑	☑	☑	☑	☑

Authentication in Alloy (MVC) project template site

The **Alloy (MVC)** project template clears all the membership or role providers, and sets authentication mode to None. In combination with an OWIN startup class (see Notes section), this activates the **ASP.NET Identity** claims-based authentication system.

```
<authentication mode="None">
  <forms name=".EPiServerLogin" loginUrl="Util/login.aspx"
         timeout="120" defaultUrl="~/" />
</authentication>
```

```
<membership><providers><clear /></providers></membership>
<roleManager><providers><clear /></providers></roleManager>
```

It also creates the **WebAdmins** group for you, gives the group full rights, and the first time you browse to the new website on the local server machine, it prompts you to create an **Admin** user.

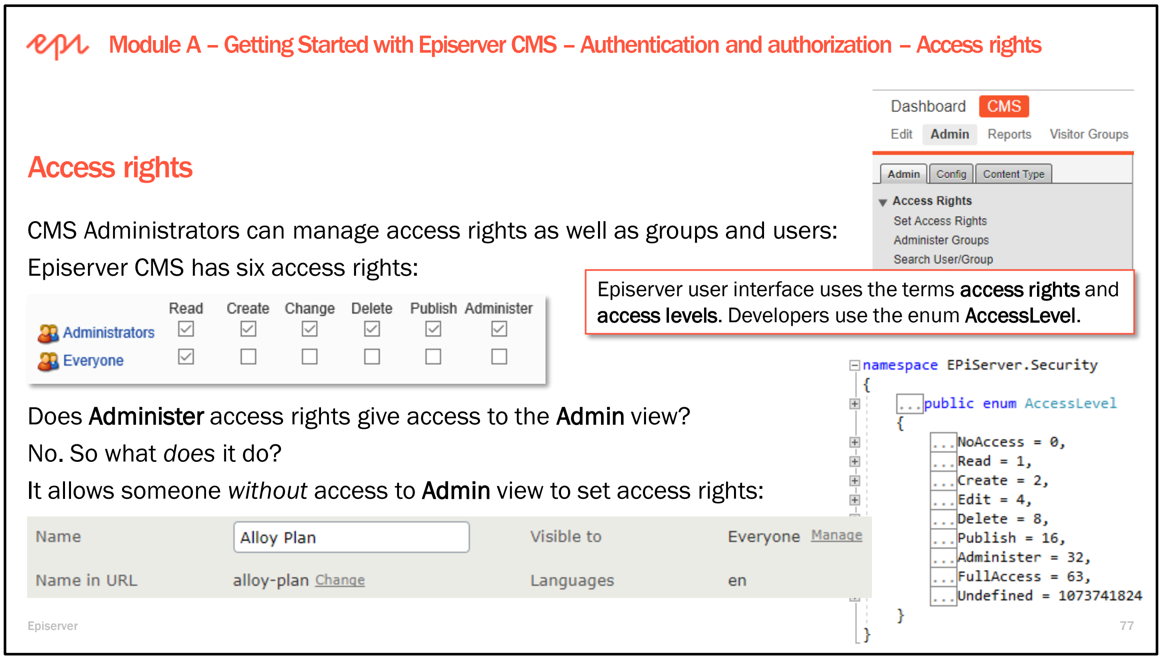
	Name	Provider
	WebAdmins	EPi_AspNetIdentityRoleProvider

	Name	Provider
	Admin	EPi_AspNetIdentityUserProvider

75

```
using EPiServer.Cms.UI.AspNetIdentity;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Owin;
using System;
using System.Web;
```

```
[assembly: OwinStartup(typeof(AlloyDemo.Startup))]
public class Startup
{
    public void Configuration(IAppBuilder app)
    {   // Add CMS integration for ASP.NET Identity
        app.AddCmsAspNetIdentity<ApplicationUser>();
        // prompt to register an admin account if browser is local on server
        app.UseAdministratorRegistrationPage(()
            => HttpContext.Current.Request.IsLocal);
        app.UseCookieAuthentication(new CookieAuthenticationOptions
        {
            AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
            LoginPath = new PathString(Global.LoginPath), // and so on
```



Module A – Getting Started with Episerver CMS – Authentication and authorization – Access rights

Access rights

CMS Administrators can manage access rights as well as groups and users:

Episerver CMS has six access rights:

	Read	Create	Change	Delete	Publish	Administer
Administrators	✓	✓	✓	✓	✓	✓
Everyone	✓	☐	☐	☐	☐	☐

Does **Administer** access rights give access to the **Admin** view?

No. So what *does* it do?

It allows someone *without* access to **Admin** view to set access rights:

Name	Alloy Plan	Visible to	Everyone Manage
Name in URL	alloy-plan Change	Languages	en

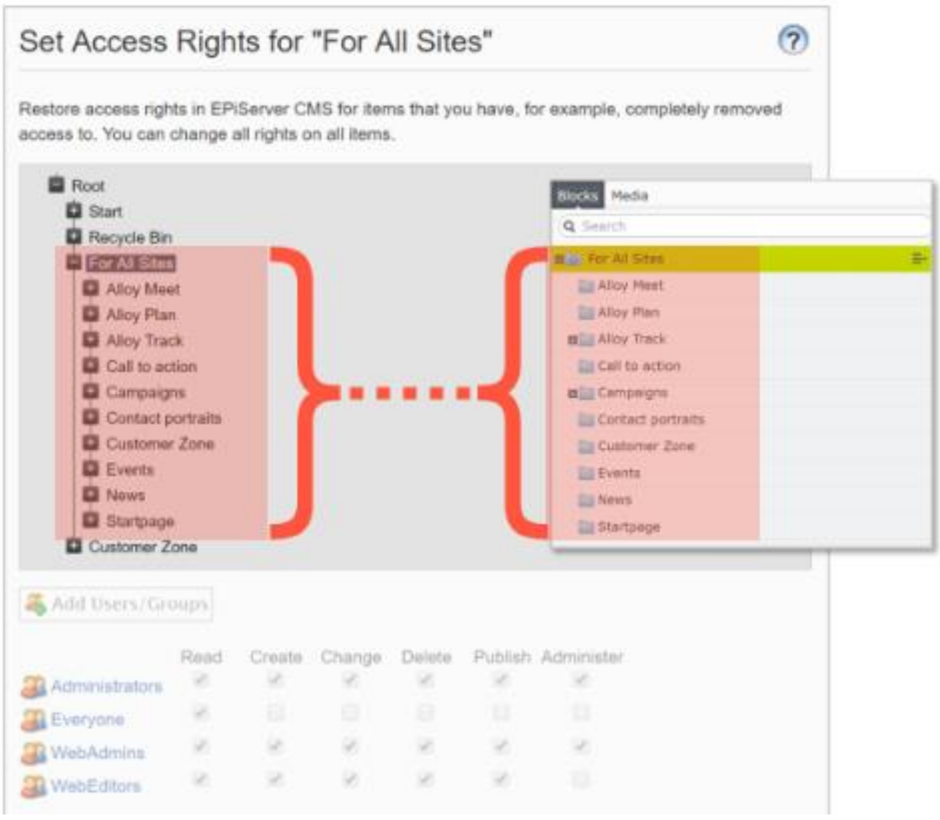
Dashboard CMS
Edit Admin Reports Visitor Groups
Admin Config Content Type
Access Rights
Set Access Rights
Administer Groups
Search User/Group

Episerver user interface uses the terms **access rights** and **access levels**. Developers use the enum **AccessLevel**.

```
namespace EPiServer.Security
{
    ...public enum AccessLevel
    {
        ...NoAccess = 0,
        ...Read = 1,
        ...Create = 2,
        ...Edit = 4,
        ...Delete = 8,
        ...Publish = 16,
        ...Administer = 32,
        ...FullAccess = 63,
        ...Undefined = 1073741824
    }
}
```

Episerver 77

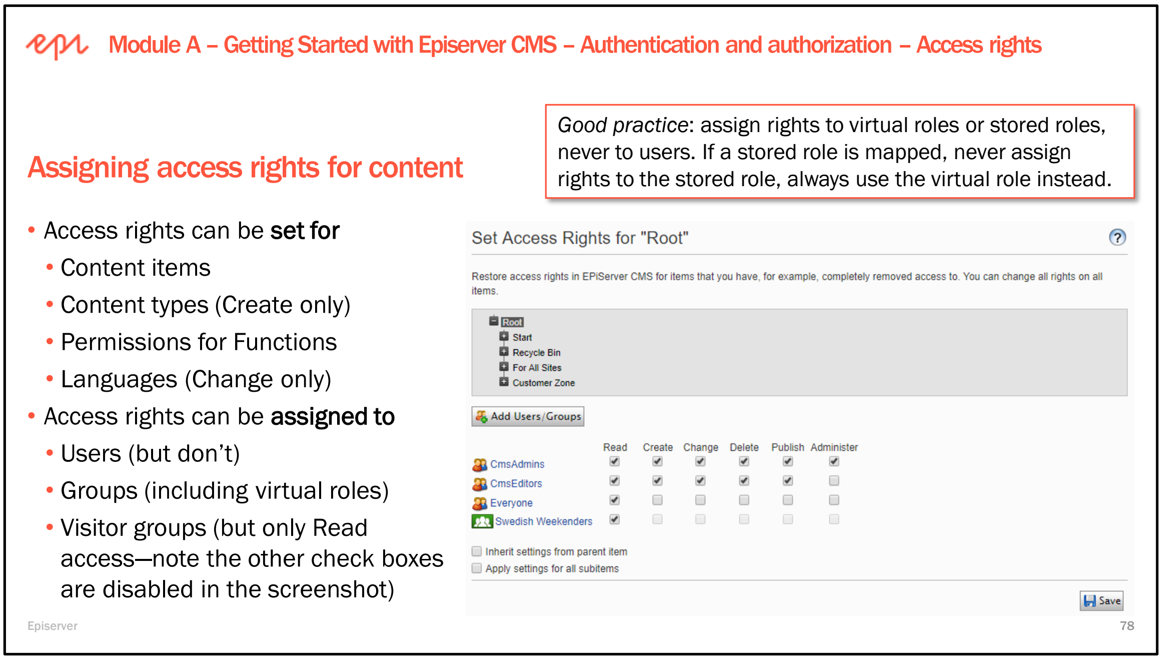
The Access Rights content tree matches content in the Pages tree in Navigation pane, and the Media/Blocks tree in Assets pane.



Set Access Rights for "For All Sites"

Restore access rights in EPiServer CMS for items that you have, for example, completely removed access to. You can change all rights on all items.

	Read	Create	Change	Delete	Publish	Administer
Administrators	✓	✓	✓	✓	✓	✓
Everyone	✓	☐	☐	☐	☐	☐
WebAdmins	✓	✓	✓	✓	✓	✓
WebEditors	✓	✓	✓	✓	✓	☐



Module A – Getting Started with Episerver CMS – Authentication and authorization – Access rights

Assigning access rights for content

Good practice: assign rights to virtual roles or stored roles, never to users. If a stored role is mapped, never assign rights to the stored role, always use the virtual role instead.

- Access rights can be **set for**
  - Content items
  - Content types (Create only)
  - Permissions for Functions
  - Languages (Change only)
- Access rights can be **assigned to**
  - Users (but don't)
  - Groups (including virtual roles)
  - Visitor groups (but only Read access—note the other check boxes are disabled in the screenshot)

Set Access Rights for "Root"

Restore access rights in EPiServer CMS for items that you have, for example, completely removed access to. You can change all rights on all items.

- Root
  - Start
  - Recycle Bin
  - For All Sites
  - Customer Zone

Add Users/Groups

	Read	Create	Change	Delete	Publish	Administer
CmsAdmins	✓	✓	✓	✓	✓	✓
CmsEditors	✓	✓	✓	✓	✓	☐
Everyone	✓	☐	☐	☐	☐	☐
Swedish Weekenders	✓	☐	☐	☐	☐	☐

☐ Inherit settings from parent item
☐ Apply settings for all subitems

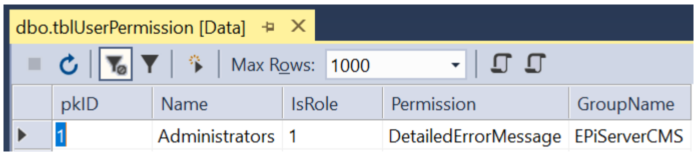
Save

Episerver 78

There are two approaches to setting access rights:

1. Top-down: Everyone can Read the Root, which is inherited, and then remove Read access to specific content.
2. Bottom-up: Everyone *cannot* Read the Root, which is inherited, and then add Read access to specific content.

Permissions for Functions



dbo.tblUserPermission [Data]

Max Rows: 1000

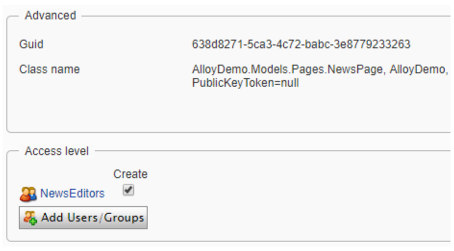
pkID	Name	IsRole	Permission	GroupName
1	Administrators	1	DetailedErrorMessage	EPiServerCMS

Module A – Getting Started with Episerver CMS – Authentication and authorization – Access rights

Assigning access rights for creating content, managing languages, and functions

```
[EPiServer.DataAnnotations.Access(
    Roles = "NewsEditors")]
public class NewsPage : StandardPage
{
```

Default access level for content types and languages is **Everyone**.



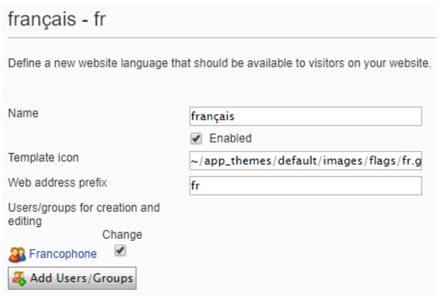
Advanced

Guid: 638d8271-5ca3-4c72-babc-3e8779233263

Class name: AlloyDemo.Models.Pages.NewsPage, AlloyDemo, PublicKeyToken=null

Access level

	Create
NewsEditors	✓



français - fr

Define a new website language that should be available to visitors on your website.

Name: français

✓ Enabled

Template icon: ~/app_themes/default/images/flags/fr.g

Web address prefix: fr

Users/groups for creation and editing

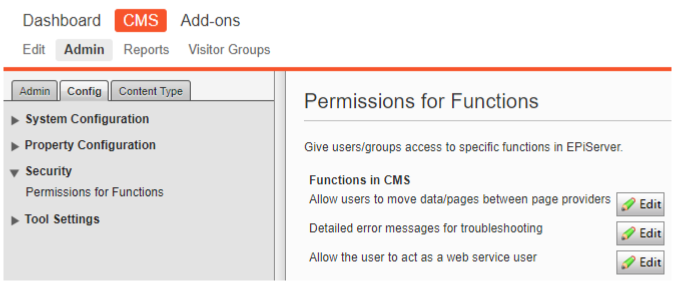
	Change
Francophone	✓

Episerver 79

Creating instances of Content Types
Makes it possible to control who will be able to create items based on a Page-/Block-/ or Media Type
Some Content Types are too difficult, risky, or seldom used for all users.

Languages
Add users and/or groups that should be able to maintain pages in a specific language branch.

Permissions for Functions
It is good practice to enable **Detailed error messages for troubleshooting** for any role that can access Admin or Edit View, i.e. the virtual roles **CmsEditors** and **CmsAdmins**. This will make it easier for the user (both editors and administrators) to understand and report any error that might occur.



Permissions for Functions

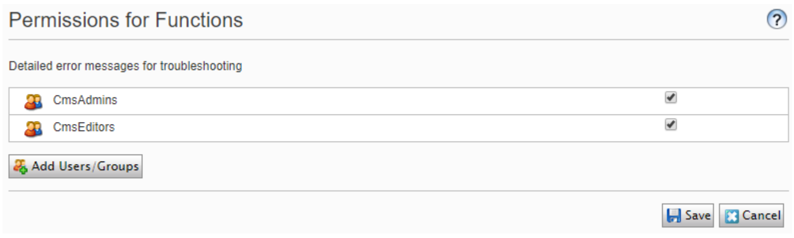
Give users/groups access to specific functions in EPiServer.

Functions in CMS

Allow users to move data/pages between page providers — Edit

Detailed error messages for troubleshooting — Edit

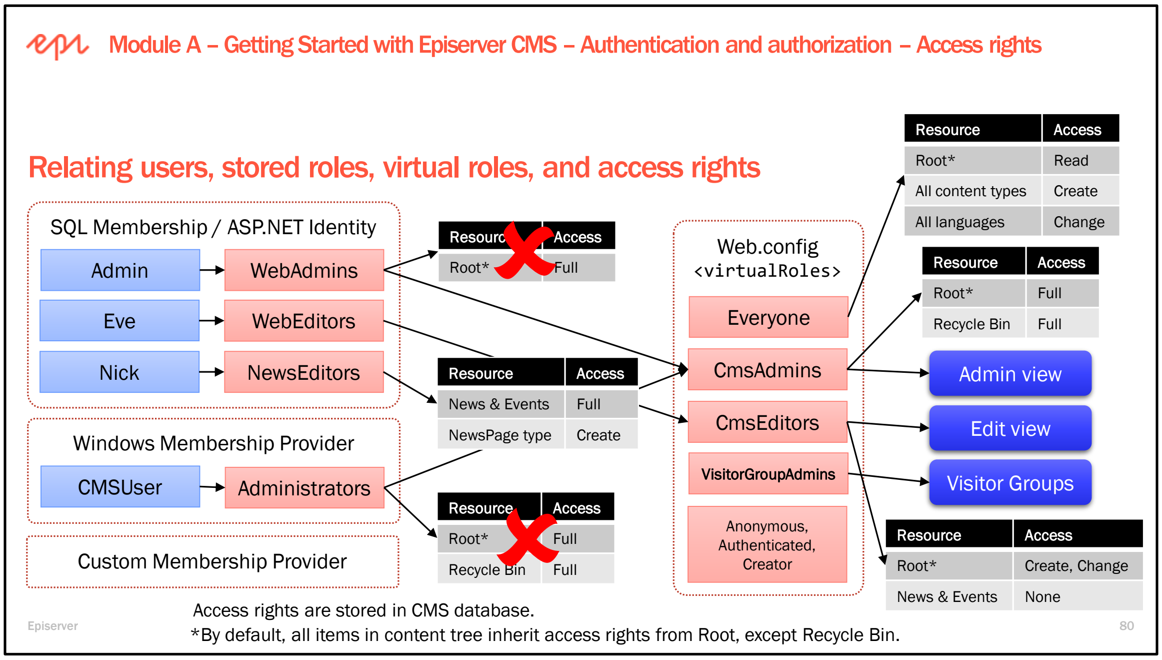
Allow the user to act as a web service user — Edit



Permissions for Functions

Detailed error messages for troubleshooting

CmsAdmins	✓
CmsEditors	✓



Module A – Getting Started with Episerver CMS – Authentication and authorization – Access rights

Relating users, stored roles, virtual roles, and access rights

Resource	Access
Root*	Full

Resource	Access
News & Events	Full
NewsPage type	Create

Resource	Access
Root*	Full
Recycle Bin	Full

Resource	Access
Root*	Read
All content types	Create
All languages	Change

Resource	Access
Root*	Full
Recycle Bin	Full

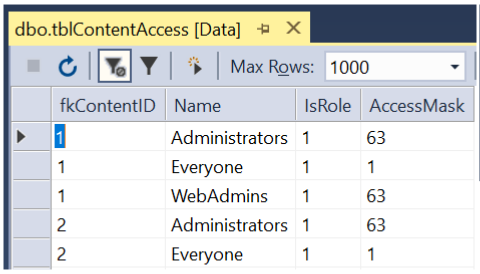
Resource	Access
Root*	Create, Change
News & Events	None

Access rights are stored in CMS database.
*By default, all items in content tree inherit access rights from Root, except Recycle Bin.

Web.config

```
<episerver.framework>
  <appData basePath="App_Data" />
  <scanAssembly forceBinFolderScan="true" />
  <virtualRoles addClaims="true">
    <providers>
      <add name="Administrators"
           type="EPiServer.Security.WindowsAdministratorsRole, EPiServer.Framework" />
      <add name="Everyone" type="EPiServer.Security.EveryoneRole, EPiServer.Framework" />
      <add name="Authenticated" type="EPiServer.Security.AuthenticatedRole, EPiServer.Framework" />
      <add name="Anonymous" type="EPiServer.Security.AnonymousRole, EPiServer.Framework" />
      <add name="CmsAdmins" type="EPiServer.Security.MappedRole, EPiServer.Framework"
           roles="WebAdmins, Administrators" mode="Any" />
      <add name="CmsEditors" type="EPiServer.Security.MappedRole, EPiServer.Framework"
           roles="WebEditors" mode="Any" />
      <add name="Creator" type="EPiServer.Security.CreatorRole, EPiServer" />
    </providers>
  </virtualRoles>
```

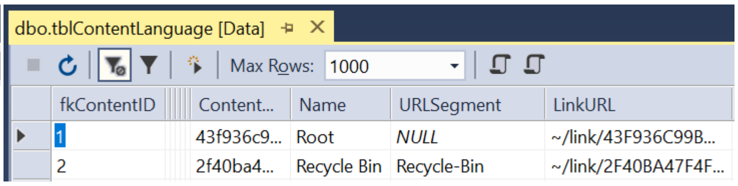
EPiServerDB



dbo.tblContentAccess [Data]

Max Rows: 1000

fkContentID	Name	IsRole	AccessMask
1	Administrators	1	63
1	Everyone	1	1
1	WebAdmins	1	63
2	Administrators	1	63
2	Everyone	1	1



dbo.tblContentLanguage [Data]

Max Rows: 1000

fkContentID	Content...	Name	URLSegment	LinkURL
1	43f936c9...	Root	NULL	~/link/43F936C99B...
2	2f40ba4...	Recycle Bin	Recycle-Bin	~/link/2F40BA47F4F...

Module A – Getting Started with Episerver CMS

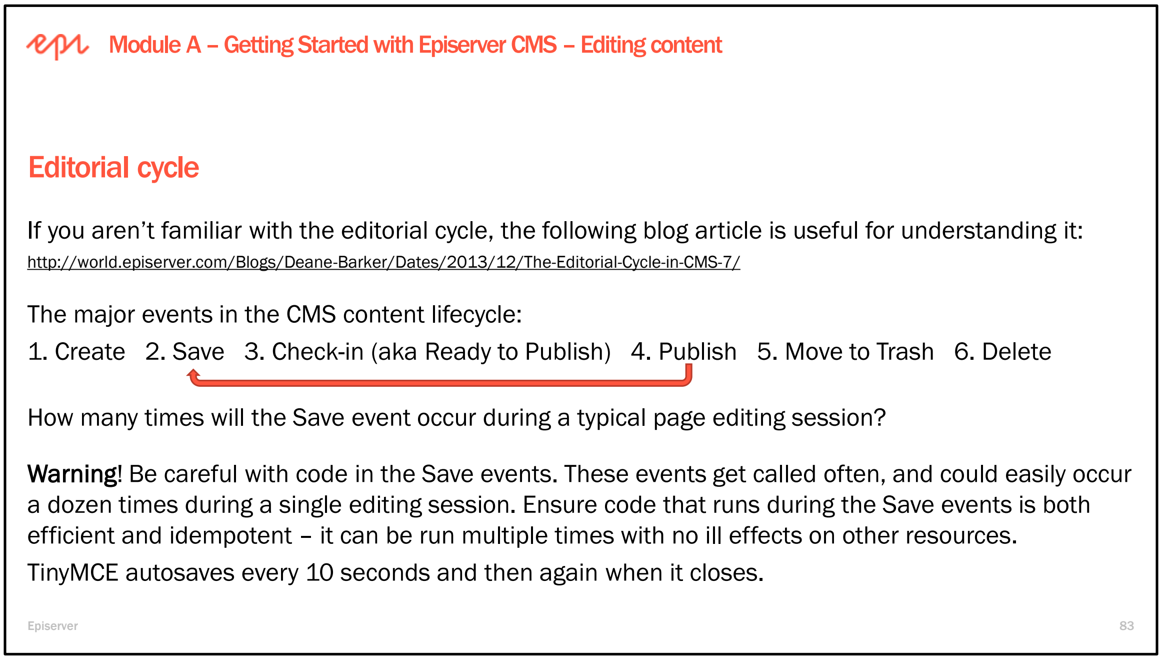
Exercise A2 – Reviewing and creating groups and users

Estimated time: 30 minutes

Prerequisites: Exercise A1.

In this exercise, you will follow good practice for setting up authentication and authorization.

Episerver 81



Module A – Getting Started with Episerver CMS – Editing content

Editorial cycle

If you aren't familiar with the editorial cycle, the following blog article is useful for understanding it:
http://world.episerver.com/Blogs/Deane-Barker/Dates/2013/12/The-Editorial-Cycle-in-CMS-7/

The major events in the CMS content lifecycle:

1. Create 2. Save 3. Check-in (aka Ready to Publish) 4. Publish 5. Move to Trash 6. Delete

How many times will the Save event occur during a typical page editing session?

Warning! Be careful with code in the Save events. These events get called often, and could easily occur a dozen times during a single editing session. Ensure code that runs during the Save events is both efficient and idempotent – it can be run multiple times with no ill effects on other resources.

TinyMCE autosaves every 10 seconds and then again when it closes.

Episerver 83

The editorial cycle in detail
http://world.episerver.com/Blogs/Deane-Barker/Dates/2013/12/The-Editorial-Cycle-in-CMS-7/
http://world.episerver.com/Blogs/Mattias-Lovstrom/Dates/2010/7/Version-state-graph-of-a-PageData-object/

Module A – Getting Started with Episerver CMS – Editing content

Deleting content

All CMS content has the ability to be moved to the **Trash**.

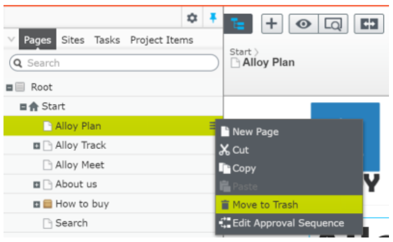
To view the items in the Trash, select **View Trash.** A user must have Administer rights to work with items in the Trash.

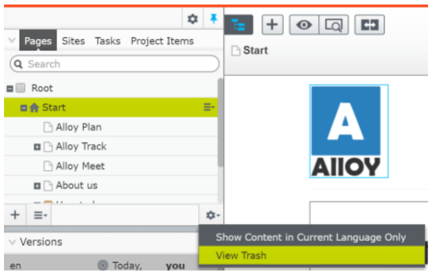
After 30 days in the Trash, the next time the scheduled job named **Automatic Emptying of Trash** executes, the item will be deleted. By default it runs once per week.

Admins can select **Empty Trash** to delete all items currently in the trash permanently. This cannot be undone.

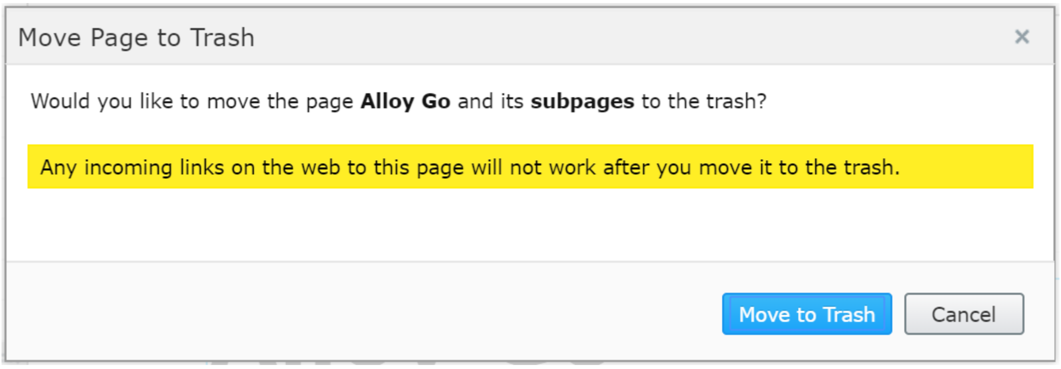
Admins can select **Restore** for an item in the Trash to restore the item to its original place in the content tree.

In **Set Access Rights**, it is named **Recycle Bin**. In scheduled jobs, it is called **Trash**. In code, **Trash** is called **Wastebasket**!





Episerver 84

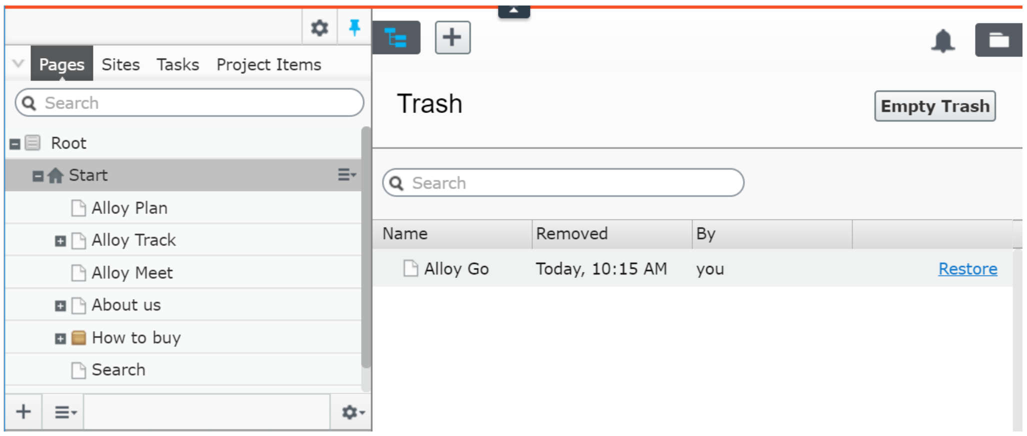


Move Page to Trash

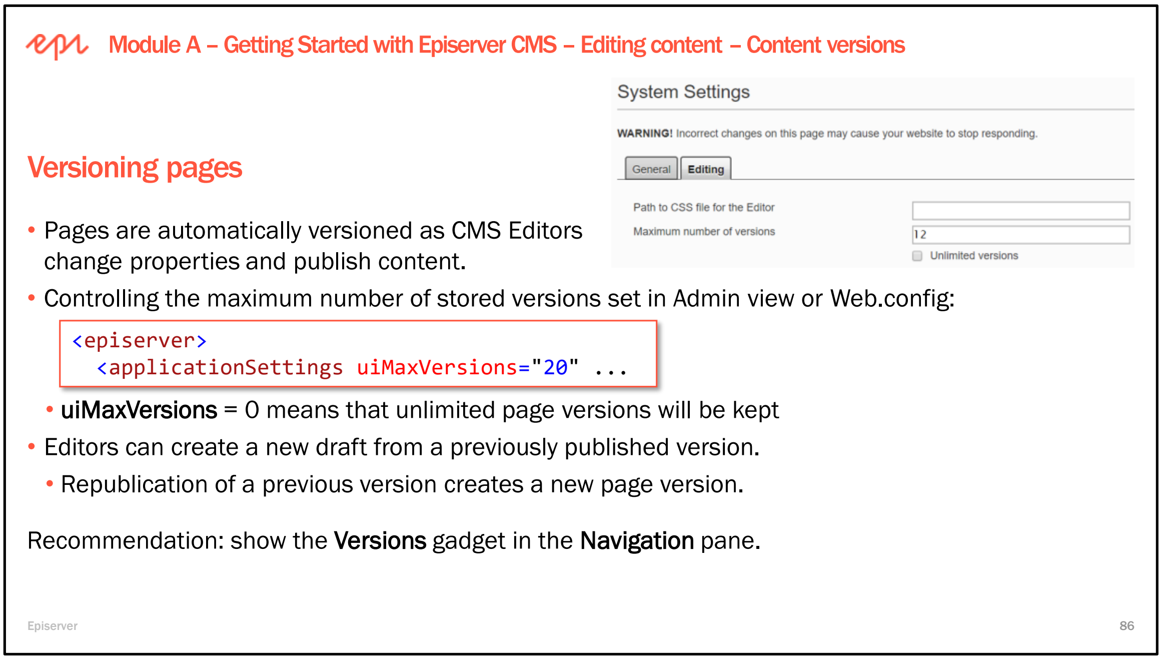
Would you like to move the page **Alloy Go** and its **subpages** to the trash?

Any incoming links on the web to this page will not work after you move it to the trash.

WARNING! By default, only members of the Windows group **Administrators** can use the Trash/Recycle Bin. Therefore in an Alloy (MVC) website you can't see content that has been moved to the Trash or empty the Trash. You must either give WebAdmins access rights, or better, give CmsAdmins access rights to Recycle Bin.

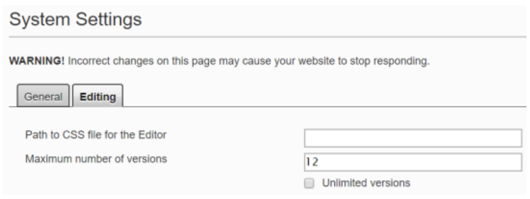


Name	Removed	By	
Alloy Go	Today, 10:15 AM	you	Restore



Module A – Getting Started with Episerver CMS – Editing content – Content versions

Versioning pages



System Settings

WARNING! Incorrect changes on this page may cause your website to stop responding.

Path to CSS file for the Editor	
Maximum number of versions	12
	☐ Unlimited versions

- Pages are automatically versioned as CMS Editors change properties and publish content.
- Controlling the maximum number of stored versions set in Admin view or Web.config:

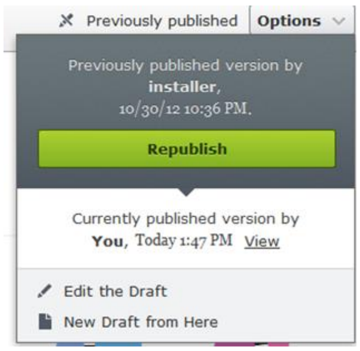
```
<episerver>
  <applicationSettings uiMaxVersions="20" ...
```

  - **uiMaxVersions** = 0 means that unlimited page versions will be kept
- Editors can create a new draft from a previously published version.
  - Republication of a previous version creates a new page version.

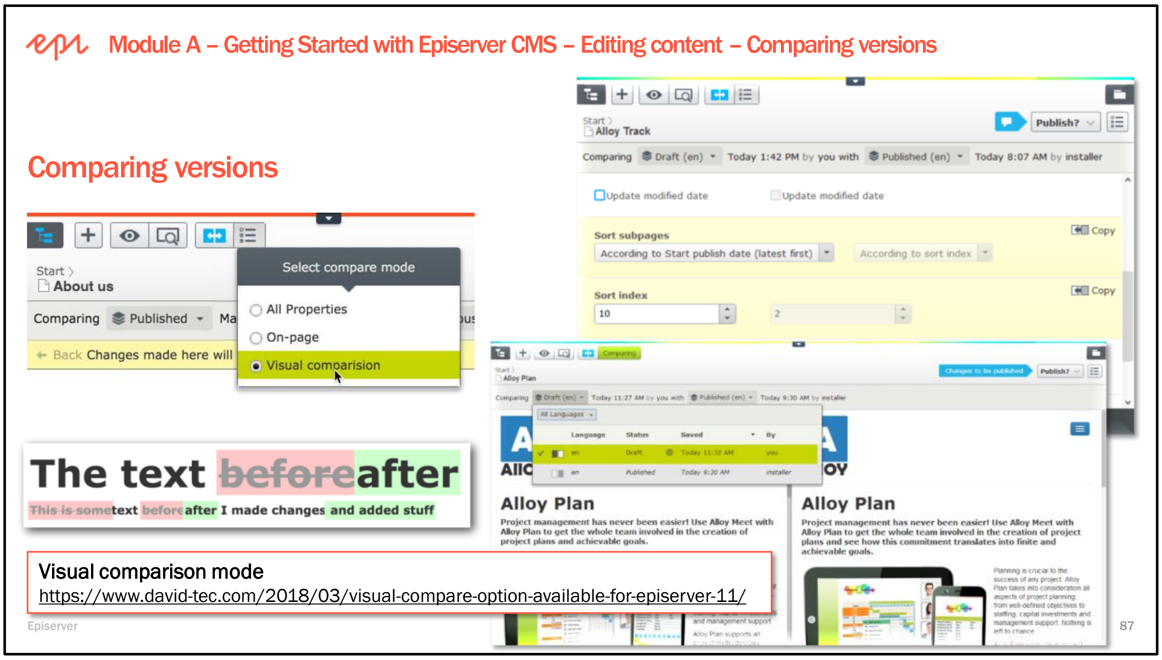
Recommendation: show the **Versions** gadget in the **Navigation** pane.

Episerver 86

You can select one of the previously published versions (add the **Versions** gadget to the Edit View to see them all) and under Options for a previously published page there is a choice **New Draft from Here**, that when selected will create a new **Not Ready** (i.e. Draft) version of the page .



Previously published
Options
Previously published version by installer, 10/30/12 10:36 PM.
Republish
Currently published version by You, Today 1:47 PM View
Edit the Draft
New Draft from Here



Module A – Getting Started with Episerver CMS – Editing content – Comparing versions

Comparing versions

Visual comparison mode
https://www.david-tec.com/2018/03/visual-compare-option-available-for-episerver-11/

Episerver

87

Compare modes

Toggle the Compare button on the toolbar to activate the compare view, two modes will be available by default: **All Properties** and **On-page**:

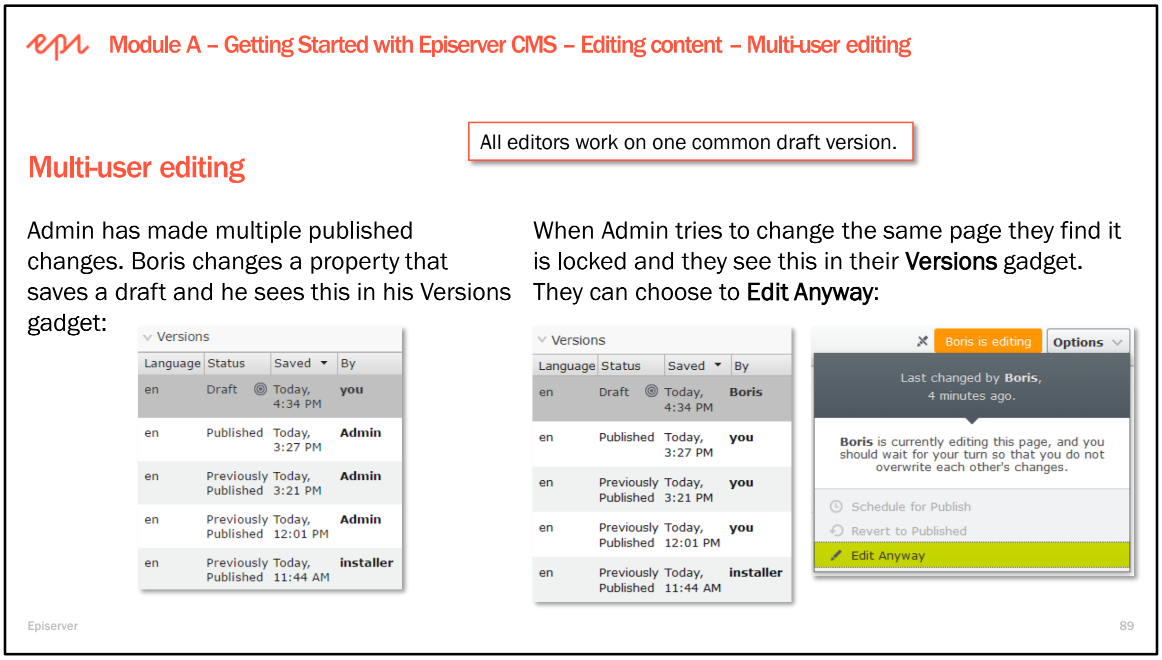
- **On-page** mode allows comparison of content for two different versions side by side.
- **All Properties** mode allows comparison of properties between versions by highlighting properties that changed with a yellow background.

Getting the visual comparison tool for Episerver

The package is open source and requires Episerver 11. It is available as a Nuget package. The source code is available on Github: https://github.com/davidknipe/VisualCompare

Word of caution

This add-on overrides some core Episerver UI components involved in showing compare options to editors. Every effort will be made to ensure this package stays compatible with the latest versions of the Episerver UI. But it's something to bear in mind (and test) when upgrading to the latest versions of Episerver.



Module A – Getting Started with Episerver CMS – Editing content – Multi-user editing

All editors work on one common draft version.

Multi-user editing

Admin has made multiple published changes. Boris changes a property that saves a draft and he sees this in his Versions gadget:

Versions

Language	Status	Saved	By
en	Draft	Today, 4:34 PM	you
en	Published	Today, 3:27 PM	Admin
en	Previously Published	Today, 3:21 PM	Admin
en	Previously Published	Today, 12:01 PM	Admin
en	Previously Published	Today, 11:44 AM	installer

When Admin tries to change the same page they find it is locked and they see this in their **Versions** gadget. They can choose to **Edit Anyway**:

Versions

Language	Status	Saved	By
en	Draft	Today, 4:34 PM	Boris
en	Published	Today, 3:27 PM	you
en	Previously Published	Today, 3:21 PM	you
en	Previously Published	Today, 12:01 PM	you
en	Previously Published	Today, 11:44 AM	installer

Boris is editing Options

Last changed by **Boris**, 4 minutes ago.

Boris is currently editing this page, and you should wait for your turn so that you do not overwrite each other's changes.

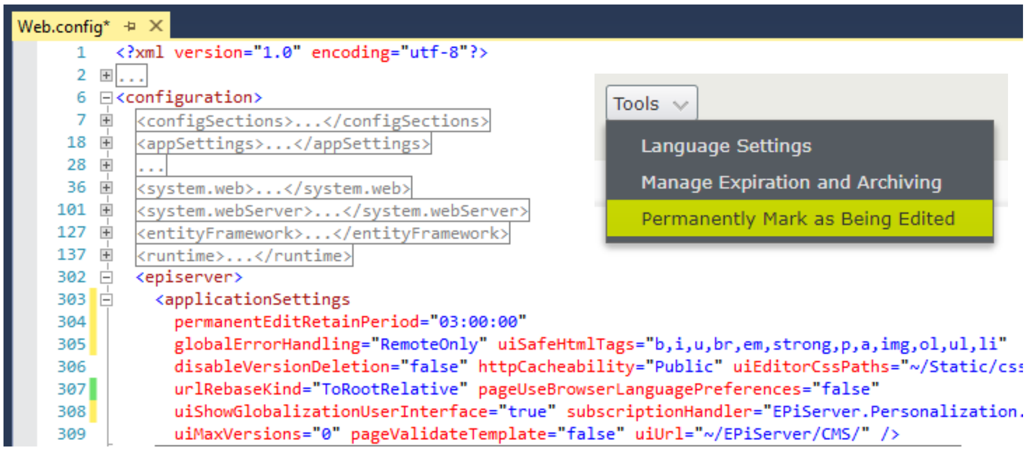
Schedule for Publish

Revert to Published

Edit Anyway

Episerver 89

Add the **permanentEditRetainPeriod** attribute with a time span value to control how long you must wait before permanent editing can be removed by the **Remove Permanent Editing** scheduled job. The default is 30 days. In the following example it has been changed to three hours:



Web.config*

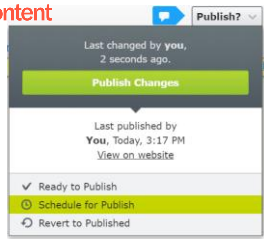
```
<?xml version="1.0" encoding="utf-8"?>
...
<configuration>
  <configSections>...</configSections>
  <appSettings>...</appSettings>
  ...
  <system.web>...</system.web>
  <system.webServer>...</system.webServer>
  <entityFramework>...</entityFramework>
  <runtime>...</runtime>
  <episerver>
    <applicationSettings
      permanentEditRetainPeriod="03:00:00"
      globalErrorHandling="RemoteOnly" uiSafeHtmlTags="b,i,u,br,em,strong,p,a,img,ol,ul,li"
      disableVersionDeletion="false" httpCacheability="Public" uiEditorCssPaths="~/Static/cs
      urlRebaseKind="ToRootRelative" pageUseBrowserLanguagePreferences="false"
      uiShowGlobalizationUserInterface="true" subscriptionHandler="EPiServer.Personalization.
      uiMaxVersions="0" pageValidateTemplate="false" uiUrl="~/EPiServer/CMS/" />
```

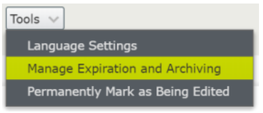
Tools
Language Settings
Manage Expiration and Archiving
Permanently Mark as Being Edited

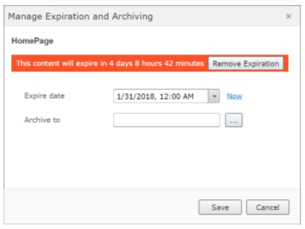
Module A – Getting Started with Episerver CMS – Editing content – Publishing content

Controlling when content is published

- **StartPublish** property and **StopPublish** property are used by the system to control when content should be published (i.e. visible to visitors).
- Content can be scheduled to be published at a later date and time.
- **Publish Delayed Content Versions** scheduled job checks for scheduled content and publishes them at the specified time, setting the user who scheduled it as the publisher in change log. Default interval is one hour.
- **Tools** | **Manage Expiration and Archiving** allows setting **StopPublish**. When a page passes the expire date, the content will no longer be considered to be published, and it returns a 404 status code.
- You can retain expired content without cluttering the content tree by moving it to an archive parent. **Archive Function** scheduled job checks for expiring content with an archive parent and moves it.




Tools
Language Settings
Manage Expiration and Archiving
Permanently Mark as Being Edited



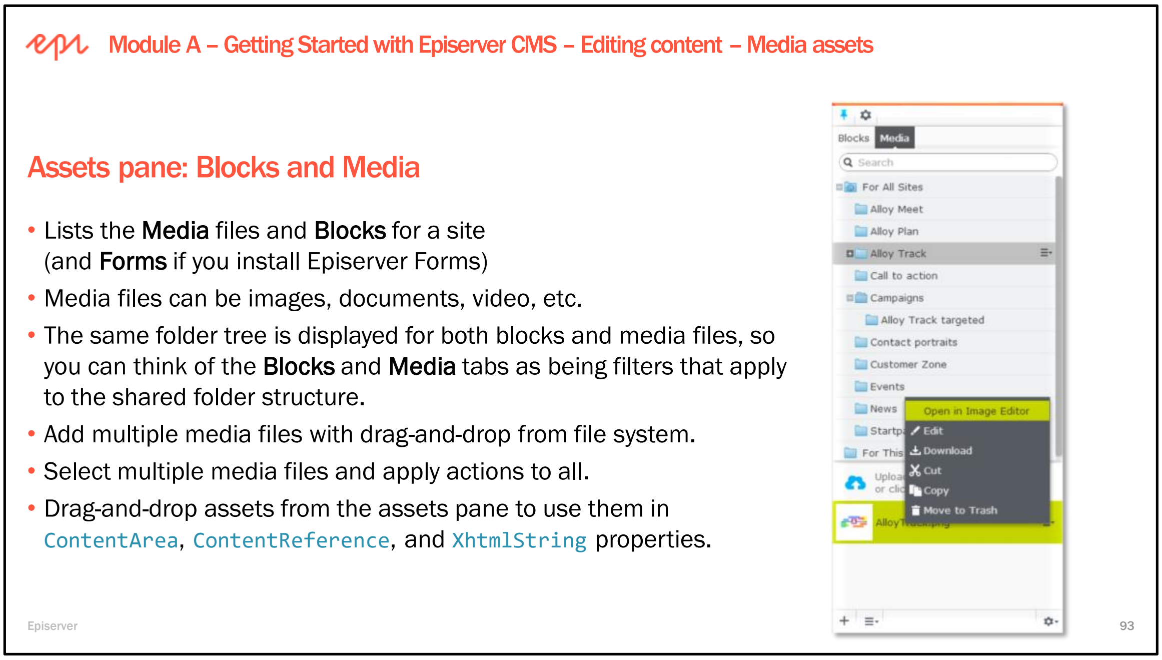
This content will expire in 4 days 8 hours 42 minutes

Expire date 1/31/2018, 12:00 AM

Episerver 91

When content has a stop publish date the behavior of the edit UI has changed. Previously, a warning in the notification field would always be visible if stop publish had been set. This has been changed so that the warning is hidden until the stop publish date is in the near future.

How long before the stop publish date the warning becomes visible is controlled by the **expirationNotificationPeriod** setting in the **applicationSettings** element in the **<episerver>** section of the site configuration files. The default is 60 days.



Module A – Getting Started with Episerver CMS – Editing content – Media assets

Assets pane: Blocks and Media

- Lists the **Media** files and **Blocks** for a site (and **Forms** if you install Episerver Forms)
- Media files can be images, documents, video, etc.
- The same folder tree is displayed for both blocks and media files, so you can think of the **Blocks** and **Media** tabs as being filters that apply to the shared folder structure.
- Add multiple media files with drag-and-drop from file system.
- Select multiple media files and apply actions to all.
- Drag-and-drop assets from the assets pane to use them in `ContentArea`, `ContentReference`, and `XhtmlString` properties.

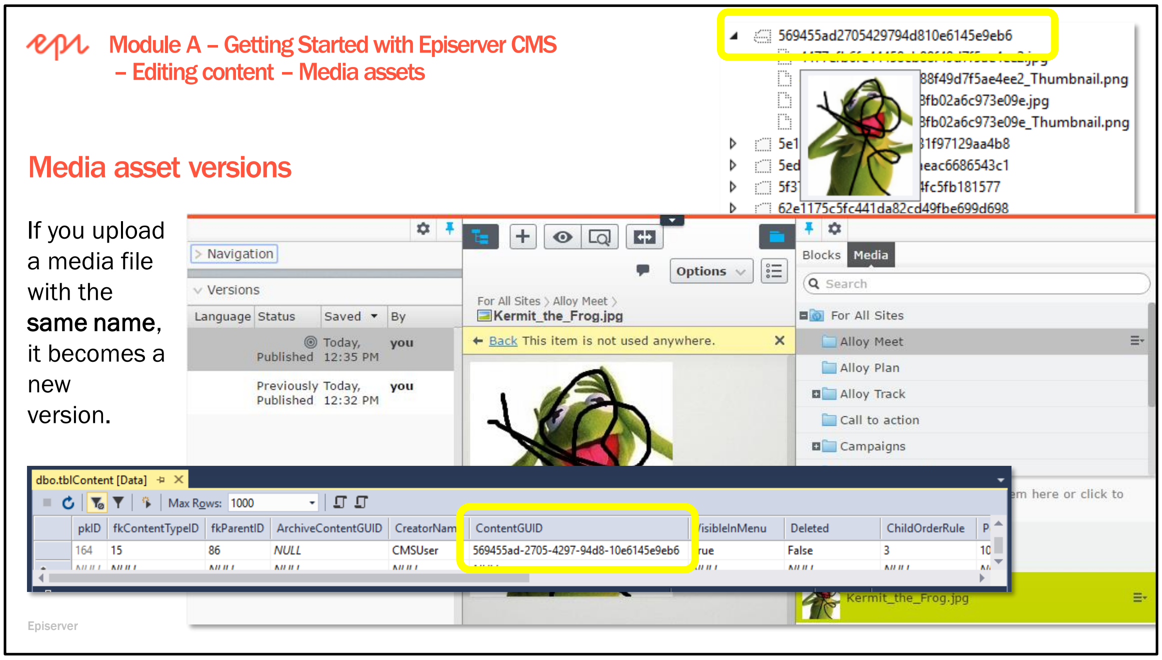
Episerver 93

For This Page/For This Block

- Files located in For This Page/Block belong to only that one specific page or block.
- Unused files, i.e. files that are not linked in pages, can be deleted automatically. This functionality only applies to files located in For This Page.
- Use the scheduled jobs “Remove Unrelated Content Assets” and “Remove Abandoned BLOBs” in Admin to remove the unused files and clean up binary stored data.
- Access rights, start-publish and stop-publish on page files are the same as on the page itself.

Files that are not being used only take space on disk!

Best Practice Tip to keep the assets archive neat and tidy: Use “For This Page”!
Tip #1: If working with a file that is to be included on more than one page, it is still beneficial to use “For This Page” and upload it twice instead of putting the file in Global Assets. Why? Imagine the following common scenario: A new file is added to Global Assets and then included in one or more pages. A few months later the reference to the file is removed from all the pages, but the editor forgets to remove it from Global Assets. Nobody else will dare to remove the file in case it is still being referenced and it will remain in the archive taking up space and making clutter.
Tip #2: While developing in a shared project; have a ”master” file share configured that everyone uses.



Module A – Getting Started with Episerver CMS – Editing content – Media assets

Media asset versions

If you upload a media file with the **same name**, it becomes a new version.

569455ad2705429794d810e6145e9eb6

62e1175c5fc441da82cd49fbe699d698

Versions

Language	Status	Saved	By
	Published	Today, 12:35 PM	you
	Previously Published	Today, 12:32 PM	you

For All Sites > Alloy Meet > Kermit_the_Frog.jpg

Back This item is not used anywhere.

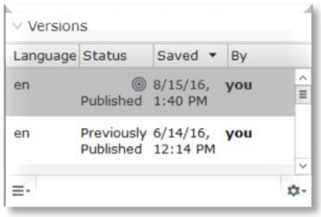
dbo.tblContent [Data]

pkID	fkContentTypeID	fkParentID	ArchiveContentGUID	CreatorNam	ContentGUID	VisibleInMenu	Deleted	ChildOrderRule	P
164	15	86	NULL	CMSUser	569455ad-2705-4297-94d8-10e6145e9eb6	rue	False	3	10

Episerver

File Versions

- It is possible to edit files and upload newer versions
- Versions gadget shows every version for the current file
- Maximum number of older versions can be set from admin or in web.config thru the “uiMaxVersions”

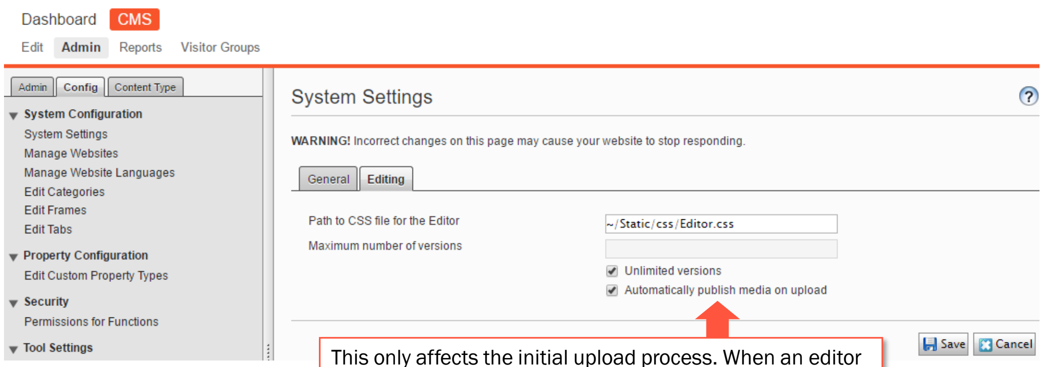


Versions

Language	Status	Saved	By
en	Published	8/15/16, 1:40 PM	you
en	Previously Published	6/14/16, 12:14 PM	you

Module A – Getting Started with Episerver CMS – Editing content – Media assets

Uploading media assets


Dashboard CMS
Edit Admin Reports Visitor Groups

Admin Config Content Type

System Configuration
System Settings
Manage Websites
Manage Website Languages
Edit Categories
Edit Frames
Edit Tabs
Property Configuration
Edit Custom Property Types
Security
Permissions for Functions
Tool Settings

System Settings

WARNING! Incorrect changes on this page may cause your website to stop responding.

General | Editing

Path to CSS file for the Editor: ~/Static/css/Editor.css
Maximum number of versions:
☑ Unlimited versions
☑ Automatically publish media on upload

Save Cancel

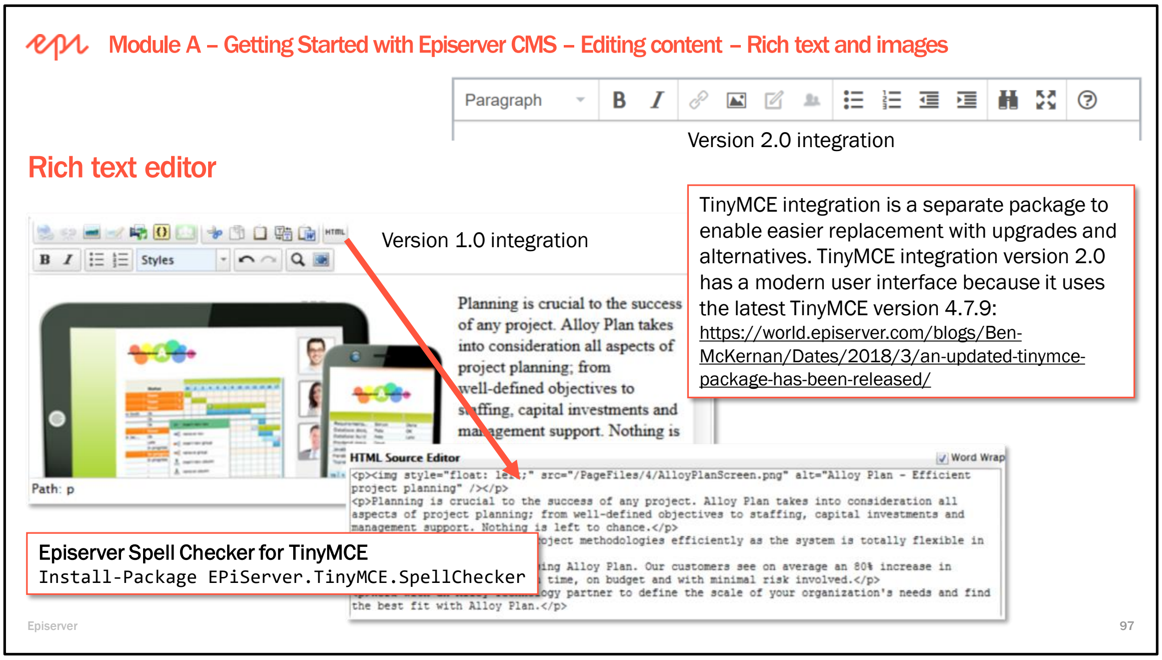
This only affects the initial upload process. When an editor changes properties of an image like Name and Description they must remember to publish those changes!

Episerver 95

Ways to Upload Files

- In assets pane
    - Drag-and-drop files to the currently selected folder
    - Using the Upload Files option in the context menu
- Programmatically by using the Episerver CMS API
- Create a custom provider

Automatically publish media on upload
When you upload media, it is by default published (and indexed) even if it is not linked to any content. You might not want this to happen with sensitive documents.
A system setting is available in Admin to stop media from being auto-published when uploaded. Another alternative is to upload sensitive documents as media “for this block” or “for this page” and set the required access rights on the content (i.e. the page/block). The access level on the media will then be the same as for the content it belongs to. Or upload media as part of a project.



Module A – Getting Started with Episerver CMS – Editing content – Rich text and images

Rich text editor

TinyMCE integration is a separate package to enable easier replacement with upgrades and alternatives. TinyMCE integration version 2.0 has a modern user interface because it uses the latest TinyMCE version 4.7.9: https://world.episerver.com/blogs/Ben-McKernan/Dates/2018/3/an-updated-tinymce-package-has-been-released/

Planning is crucial to the success of any project. Alloy Plan takes into consideration all aspects of project planning; from well-defined objectives to staffing, capital investments and management support. Nothing is

```
<p><img style="float: left;" src="/PageFiles/4/AlloyPlanScreen.png" alt="Alloy Plan - Efficient
project planning" /></p>
<p>Planning is crucial to the success of any project. Alloy Plan takes into consideration all
aspects of project planning; from well-defined objectives to staffing, capital investments and
management support. Nothing is left to chance.</p>
                       oject methodologies efficiently as the system is totally flexible in

                       ing Alloy Plan. Our customers see on average an 80% increase in
                       time, on budget and with minimal risk involved.</p>
                       ogy partner to define the scale of your organization's needs and find
the best fit with Alloy Plan.</p>
```

Episerver Spell Checker for TinyMCE
`Install-Package EPiServer.TinyMCE.SpellChecker`

Episerver 97

- 3rd party Javascript HTML WYSIWYG editor integrated with Episerver CMS with cross-browser support
- Creates clean validated XHTML markup
- Drag and drop files and images to the Editor from the Assets Pane
- Easy for developers to plug into and extend

Episerver only bundles the standard plugins with the editor. So if you have a customer that wants to use a premium plugin they will need to license that from TinyMCE directly.

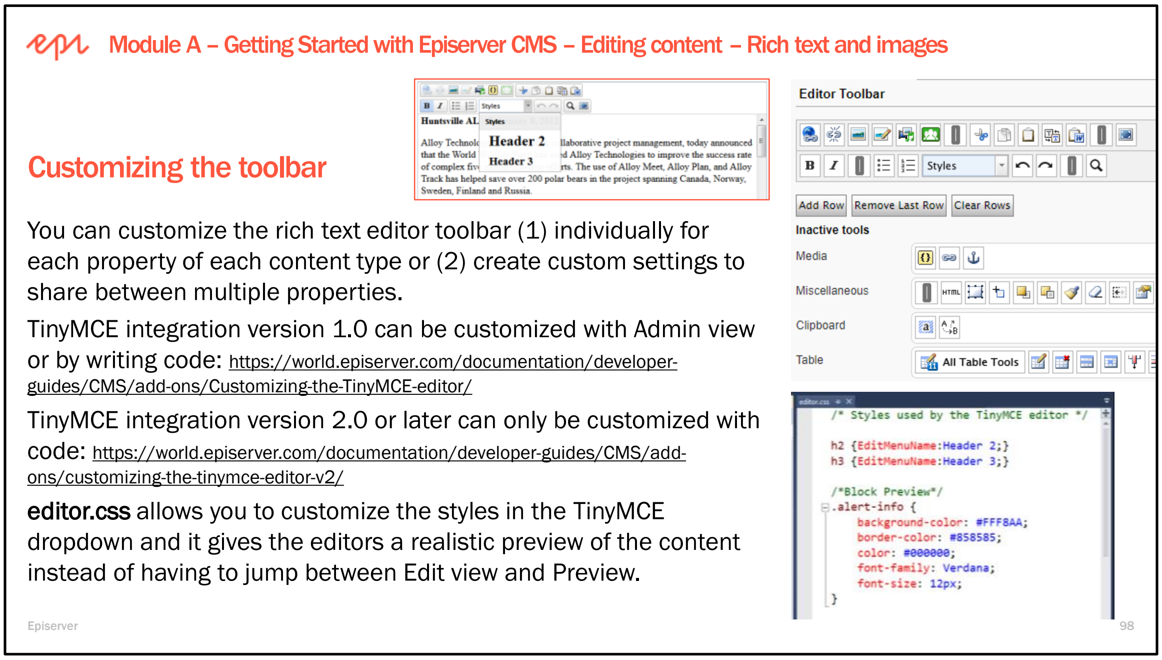
First look at Episerver updated TinyMCE editor
https://swapcode.wordpress.com/2018/03/20/first-look-at-episerver-updated-tinymce-editor/

Configuring the editor with version 2.0

We have introduced a new **TinyMceSettings** class, which can be configured with any setting supported by TinyMCE, and also a **TinyMceConfiguration** class, which is responsible for mapping settings to properties on page types. Here is an example of how to extend the default settings and also how to configure custom settings for a particular property:

```
context.Services.Configure<TinyMceConfiguration>(config =>
{
    // Add content CSS to the default settings.
    config.Default()
        .ContentCss("/static/css/editor.css");

    // Limit the block formats for the MainBody property of an ArticlePage.
    config.For<ArticlePage>(t => t.MainBody)
        .BlockFormats("Paragraph=p;Header 1=h1;Header 2=h2;Header 3=h3");
});
```

Full documentation on TinyMCE is available on http://www.tinymce.com/



Module A – Getting Started with Episerver CMS – Editing content – Rich text and images

Customizing the toolbar

You can customize the rich text editor toolbar (1) individually for each property of each content type or (2) create custom settings to share between multiple properties.

TinyMCE integration version 1.0 can be customized with Admin view or by writing code: https://world.episerver.com/documentation/developer-guides/CMS/add-ons/Customizing-the-TinyMCE-editor/

TinyMCE integration version 2.0 or later can only be customized with code: https://world.episerver.com/documentation/developer-guides/CMS/add-ons/customizing-the-tinymce-editor-v2/

editor.css allows you to customize the styles in the TinyMCE dropdown and it gives the editors a realistic preview of the content instead of having to jump between Edit view and Preview.

```
/* Styles used by the TinyMCE editor */

h2 {EditMenuName:Header 2;}
h3 {EditMenuName:Header 3;}

/*Block Preview*/
.alert-info {
    background-color: #FFF8AA;
    border-color: #858585;
    color: #000000;
    font-family: Verdana;
    font-size: 12px;
}
```

Episerver 98

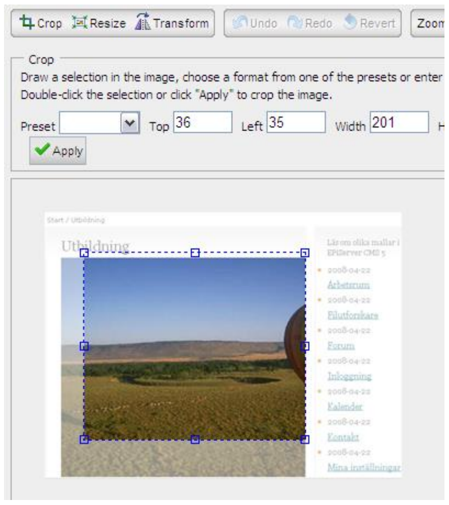
TinyMce control identifier constants

To add toolbar and menu buttons we need to know the string constants defined for these which are listed on the TinyMCE website: https://swapcode.wordpress.com/2018/04/02/tinymce-control-identifier-constants/

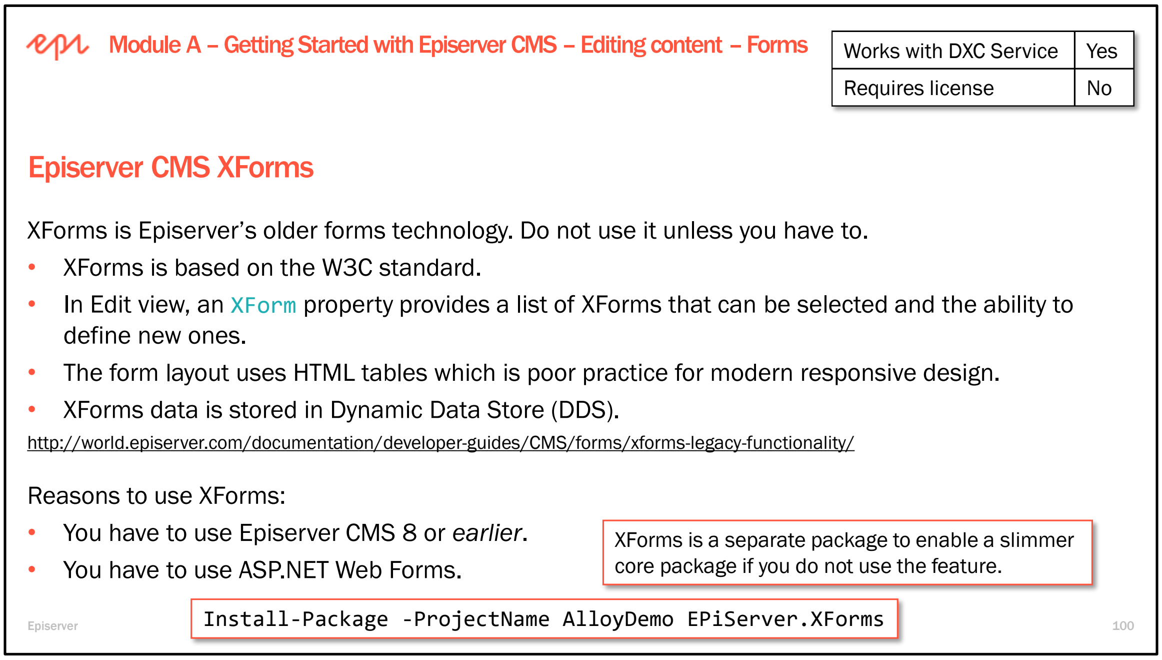
Customizing the image editor

With the image editor there are functions to crop and resize images. By using preset formats you can make it easier for editors to resize and crop to most commonly used sizes on the website. To customize the preset sizes, modify the imageEditor section in Web.config:

```
<episerver>
  <imageEditor>
    <sizePresets>
      <preset width="250" height="150" />
      <preset width="150" height="250" />
```



Crop
Draw a selection in the image, choose a format from one of the presets or enter
Double-click the selection or click "Apply" to crop the image.
Preset Top 36 Left 35 Width 201
Apply



Module A – Getting Started with Episerver CMS – Editing content – Forms

Works with DXC Service	Yes
Requires license	No

Episerver CMS XForms

XForms is Episerver's older forms technology. Do not use it unless you have to.

- XForms is based on the W3C standard.
- In Edit view, an `XForm` property provides a list of XForms that can be selected and the ability to define new ones.
- The form layout uses HTML tables which is poor practice for modern responsive design.
- XForms data is stored in Dynamic Data Store (DDS).

http://world.episerver.com/documentation/developer-guides/CMS/forms/xforms-legacy-functionality/

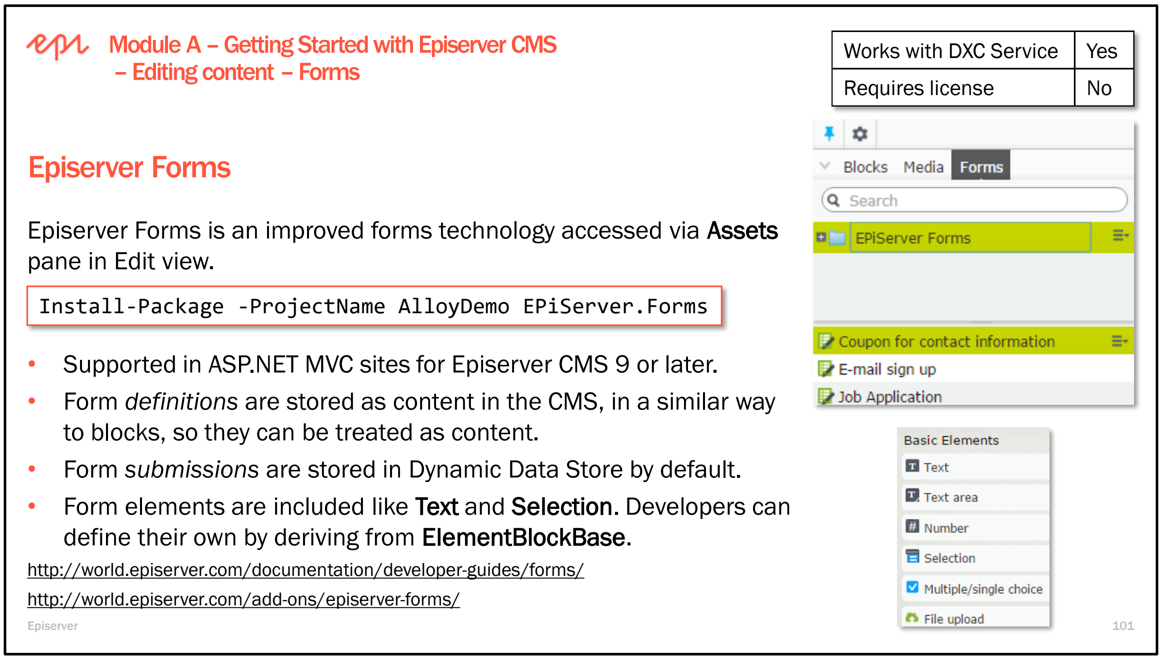
Reasons to use XForms:

- You have to use Episerver CMS 8 or *earlier*.
- You have to use ASP.NET Web Forms.

XForms is a separate package to enable a slimmer core package if you do not use the feature.

```
Install-Package -ProjectName AlloyDemo EPiServer.XForms
```

Episerver 100



Module A – Getting Started with Episerver CMS
– Editing content – Forms

Works with DXC Service	Yes
Requires license	No

Episerver Forms

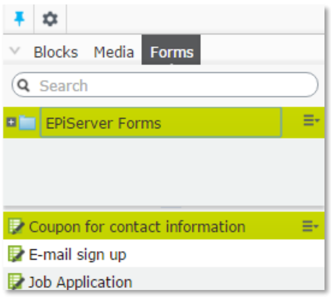
Episerver Forms is an improved forms technology accessed via **Assets** pane in Edit view.

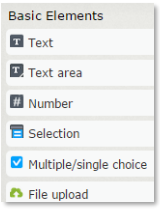
```
Install-Package -ProjectName AlloyDemo EPiServer.Forms
```

- Supported in ASP.NET MVC sites for Episerver CMS 9 or later.
- Form *definitions* are stored as content in the CMS, in a similar way to blocks, so they can be treated as content.
- Form *submissions* are stored in Dynamic Data Store by default.
- Form elements are included like **Text** and **Selection.** Developers can define their own by deriving from **ElementBlockBase.**

http://world.episerver.com/documentation/developer-guides/forms/

http://world.episerver.com/add-ons/episerver-forms/

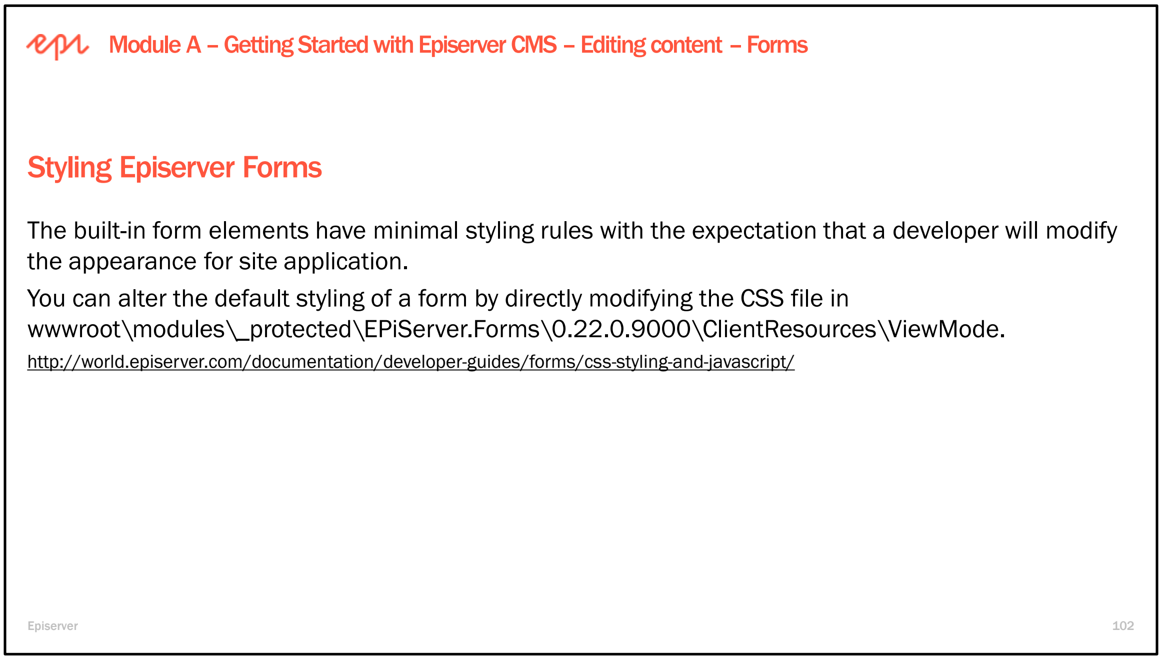




Episerver 101

Episerver Forms videos
Use the following link and scroll down to the Episerver Forms section:
http://webhelp.episerver.com/latest/_online-only-topics/videos.htm

Add-ons
Episerver Forms
Email form page
Creating an email sign-up form
Creating a multi-step form
Post-processing with form metadata



Module A – Getting Started with Episerver CMS – Editing content – Forms

Styling Episerver Forms

The built-in form elements have minimal styling rules with the expectation that a developer will modify the appearance for site application.

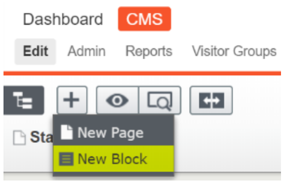
You can alter the default styling of a form by directly modifying the CSS file in wwwroot\modules_protected\EPiServer.Forms\0.22.0.9000\ClientResources\ViewMode.

http://world.episerver.com/documentation/developer-guides/forms/css-styling-and-javascript/

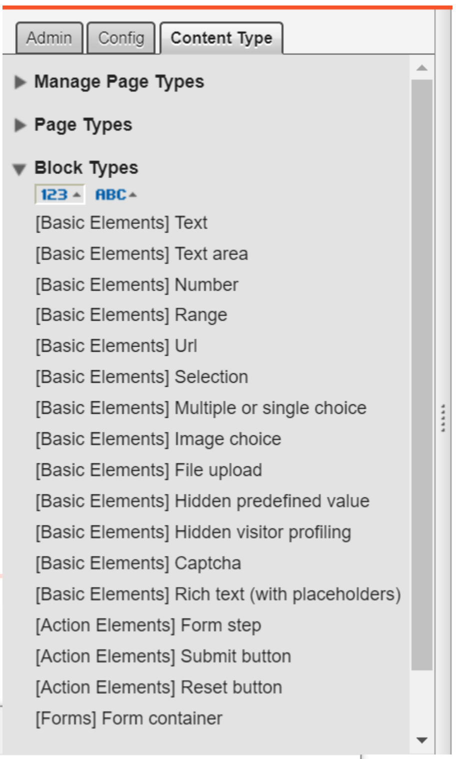
Episerver 102

Forms and form elements are built on blocks, as you can see in this screenshot of Admin view's **Content Type** tab.

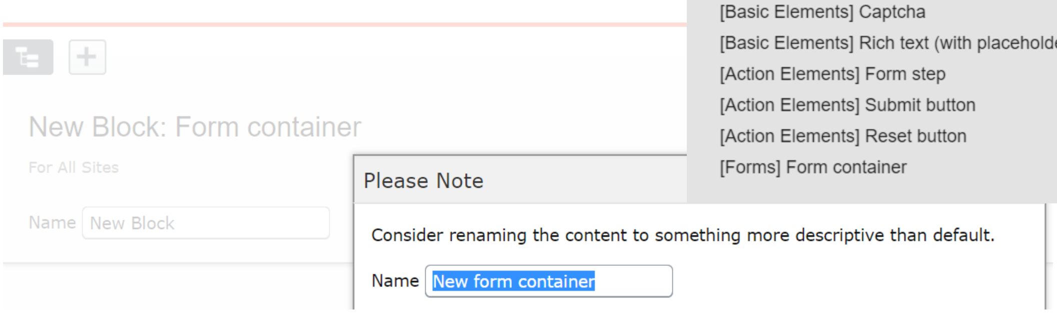
Basic and **Action** form elements can only be created inside a **Form container**, but **Form container** is treated as a normal block type. This can cause confusion if you do not have any of your own block types, and an Editor tries to create a **New Block**:



Since **Form container** is the only block type registered, a new instance of it is created without allowing the Editor to choose a different type of block!



[Basic Elements] Text
[Basic Elements] Text area
[Basic Elements] Number
[Basic Elements] Range
[Basic Elements] Url
[Basic Elements] Selection
[Basic Elements] Multiple or single choice
[Basic Elements] Image choice
[Basic Elements] File upload
[Basic Elements] Hidden predefined value
[Basic Elements] Hidden visitor profiling
[Basic Elements] Captcha
[Basic Elements] Rich text (with placeholders)
[Action Elements] Form step
[Action Elements] Submit button
[Action Elements] Reset button
[Forms] Form container



New Block: Form container

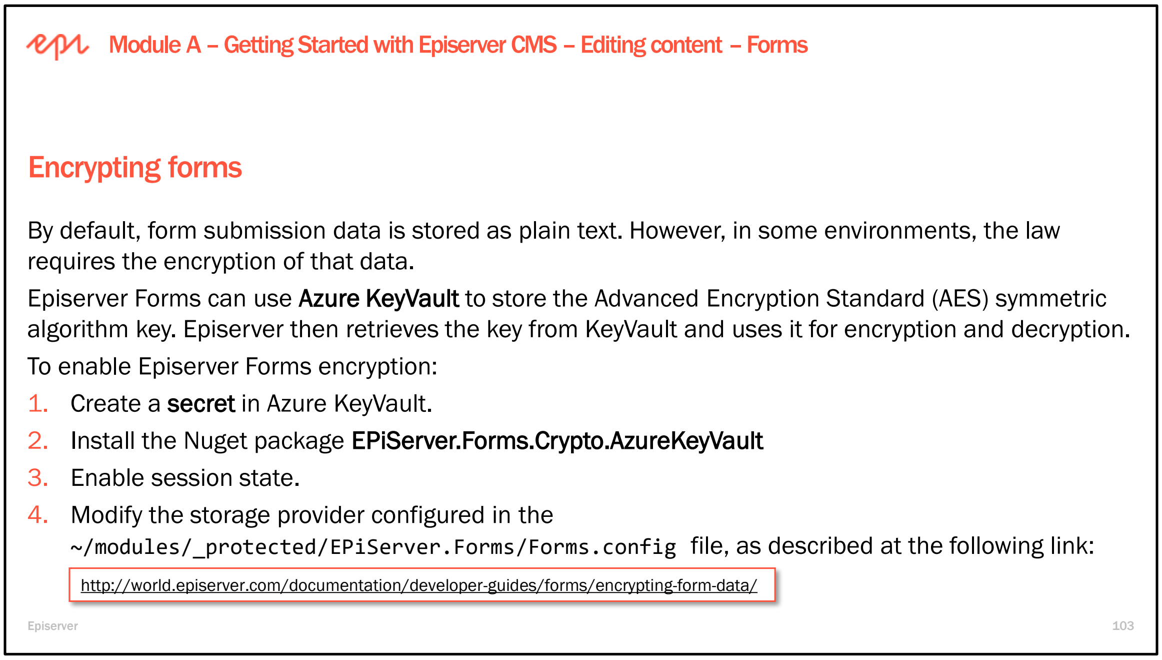
For All Sites

Name: New Block

Please Note

Consider renaming the content to something more descriptive than default.

Name: New form container



Module A – Getting Started with Episerver CMS – Editing content – Forms

Encrypting forms

By default, form submission data is stored as plain text. However, in some environments, the law requires the encryption of that data.

Episerver Forms can use **Azure KeyVault** to store the Advanced Encryption Standard (AES) symmetric algorithm key. Episerver then retrieves the key from KeyVault and uses it for encryption and decryption.

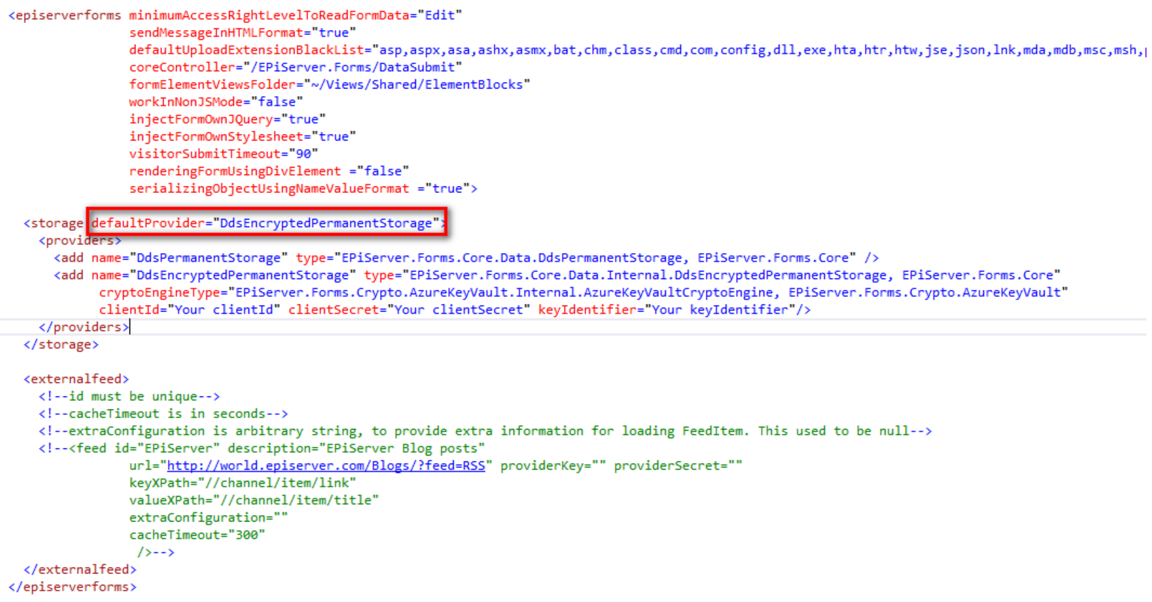
To enable Episerver Forms encryption:

1. Create a **secret** in Azure KeyVault.
2. Install the Nuget package **EPiServer.Forms.Crypto.AzureKeyVault**
3. Enable session state.
4. Modify the storage provider configured in the `~/modules/_protected/EPiServer.Forms/Forms.config` file, as described at the following link:

http://world.episerver.com/documentation/developer-guides/forms/encrypting-form-data/

Episerver 103

Before the <providers> section, add a <storage defaultProvider="DdsEncryptedPermanentStorage"> element, and within the <providers> element, specify three Azure KeyVault–related parameters for the cryptographic engine:



```
<episerverforms minimumAccessRightLevelToReadFormData="Edit"
                sendMessageInHTMLFormat="true"
                defaultUploadExtensionBlackList="asp,aspx,asa,ashx,asmx,bat,chm,class,cmd,com,config,dll,exe,hta,htr,htw,jse,json,lnk,mda,mdb,msc,msh,
                coreController="/EPiServer.Forms/DataSubmit"
                formElementViewsFolder="~/Views/Shared/ElementBlocks"
                workInNonJSMode="false"
                injectFormOwnJQuery="true"
                injectFormOwnStylesheet="true"
                visitorSubmitTimeout="90"
                renderingFormUsingDivElement ="false"
                serializingObjectUsingNameValueFormat ="true">

  <storage defaultProvider="DdsEncryptedPermanentStorage">
    <providers>
      <add name="DdsPermanentStorage" type="EPiServer.Forms.Core.Data.DdsPermanentStorage, EPiServer.Forms.Core" />
      <add name="DdsEncryptedPermanentStorage" type="EPiServer.Forms.Core.Data.Internal.DdsEncryptedPermanentStorage, EPiServer.Forms.Core"
          cryptoEngineType="EPiServer.Forms.Crypto.AzureKeyVault.Internal.AzureKeyVaultCryptoEngine, EPiServer.Forms.Crypto.AzureKeyVault"
          clientId="Your clientId" clientSecret="Your clientSecret" keyIdentifier="Your keyIdentifier"/>
    </providers>
  </storage>

  <externalfeed>
    <!--id must be unique-->
    <!--cacheTimeout is in seconds-->
    <!--extraConfiguration is arbitrary string, to provide extra information for loading FeedItem. This used to be null-->
    <!--<feed id="EPiServer" description="EPiServer Blog posts"
                url="http://world.episerver.com/Blogs/?feed=RSS" providerKey="" providerSecret=""
                keyXPath="//channel/item/link"
                valueXPath="//channel/item/title"
                extraConfiguration=""
                cacheTimeout="300"
                 />-->
  </externalfeed>
</episerverforms>
```

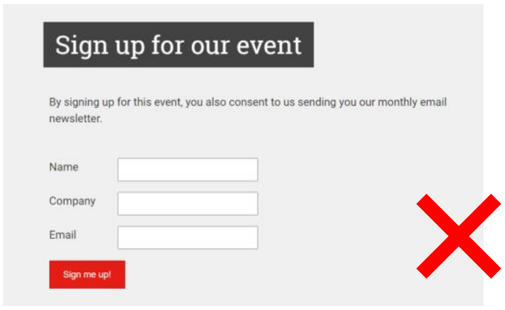
Setup guide #Azure #KeyVault with Azure Active Directory Authentication
https://devblog.gosso.se/2017/11/setup-guide-azure-keyvault-with-azure-active-directory-authentication/

Module A – Getting Started with Episerver CMS – Editing content – Forms

GDPR and Episerver: Unbundled consent in signup forms
https://www.epinova.no/en/blog/gdpr-and-episerver-unbundled-consent-in-signup-forms/

GDPR

It's common to get the user subscribed to newsletters or promotional offers at the same time as signing up for events and other reasons. To be GDPR-compliant you must unbundle consent.

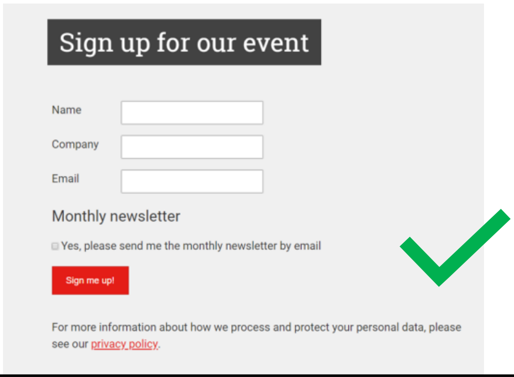


Sign up for our event

By signing up for this event, you also consent to us sending you our monthly email newsletter.

Name
Company
Email

Sign me up!



Sign up for our event

Name
Company
Email

Monthly newsletter

☐ Yes, please send me the monthly newsletter by email

Sign me up!

For more information about how we process and protect your personal data, please see our privacy policy.

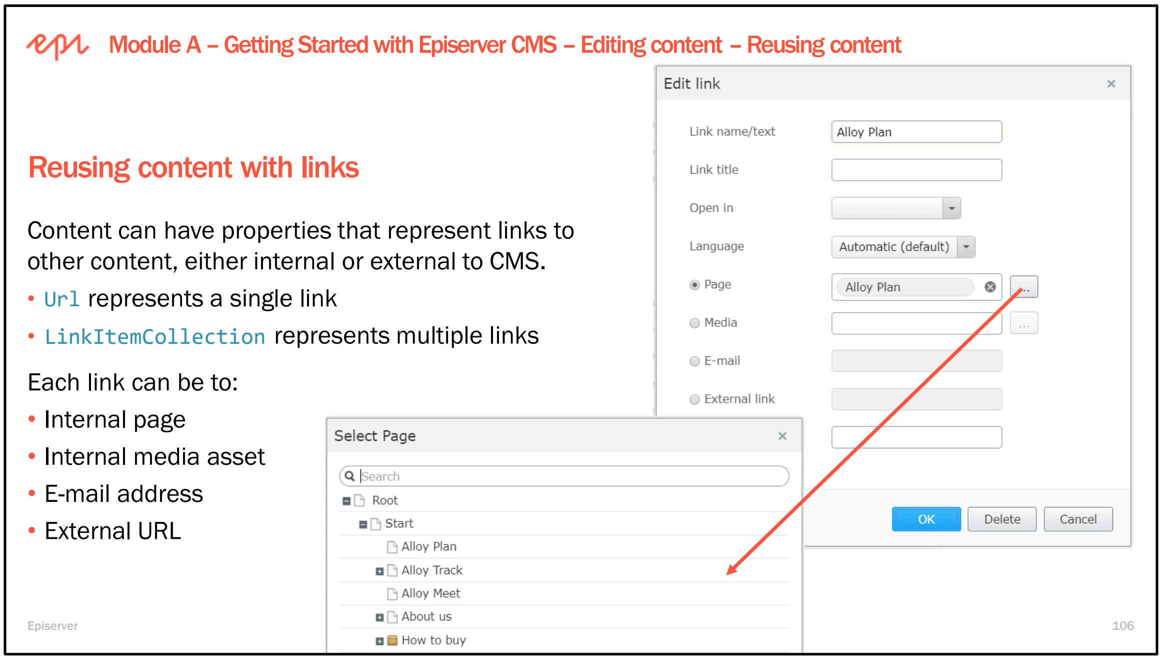
Episerver 104

General Data Protection Regulation and Episerver
Learn how to leverage your organization's data to support GDPR compliance. Learn about the impacts, opportunities and key considerations to prepare for the new data protection law.
https://www.episerver.com/products/features/gdpr/

GDPR compliance audit of the Episerver "QJet" demo site
https://www.epinova.no/en/blog/gdpr-compliance-audit-of-the-episerver-qjet-demo-site/

GDPR and Episerver: Storing consent context in submitted form data
https://www.epinova.no/en/blog/gdpr-and-episerver-storing-consent-context-in-submitted-form-data/

10 Considerations for GDPR
https://www.episerver.com/learn/resources/blog/peter-yeung/10-considerations-for-gdpr-part-1/
https://www.episerver.com/learn/resources/blog/peter-yeung/10-considerations-for-gdpr-part-2/



Module A – Getting Started with Episerver CMS – Editing content – Reusing content

Reusing content with links

Content can have properties that represent links to other content, either internal or external to CMS.

- `Url` represents a single link
- `LinkItemCollection` represents multiple links

Each link can be to:

- Internal page
- Internal media asset
- E-mail address
- External URL

Edit link

Link name/text	Alloy Plan
Link title	
Open in	
Language	Automatic (default)
● Page	Alloy Plan
○ Media	
○ E-mail	
○ External link	

OK Delete Cancel

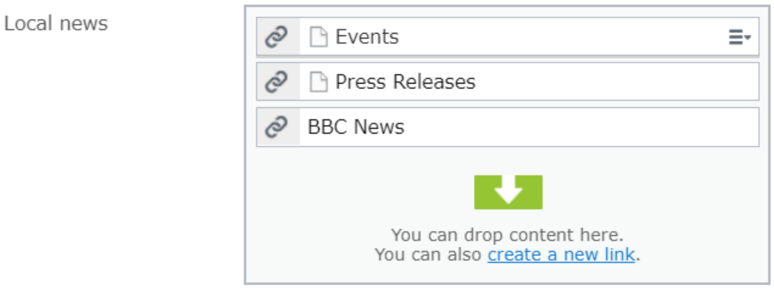
Select Page

Search

- Root
  - Start
    - Alloy Plan
    - Alloy Track
    - Alloy Meet
    - About us
    - How to buy

Episerver 106

All Properties view of LinkItemCollection



Local news

- Events
- Press Releases
- BBC News

You can drop content here.
You can also create a new link.

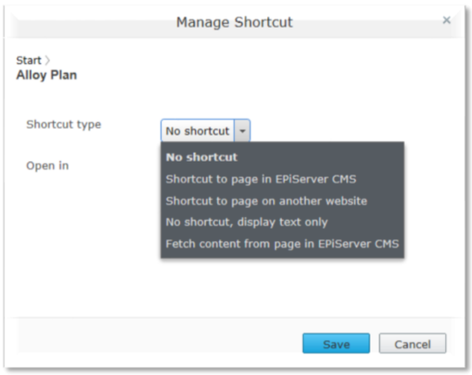
Module A – Getting Started with Episerver CMS – Editing content – Reusing content

Reusing content with page shortcuts

Determines how the system should respond to a click on the link rendered for the current page.

Links can be in a rich text property or in an automatically generated menu.

- No shortcut (default)
- Shortcut to page in Episerver CMS: allows a page to appear in multiple parts of multiple sites.
- Shortcut to page on another website
- No shortcut, display text only
- Fetch content from page in Episerver CMS: allows reuse of property values in multiple pages


Manage Shortcut
Start
Alloy Plan
Shortcut type
No shortcut
Open in
No shortcut
Shortcut to page in EPiServer CMS
Shortcut to page on another website
No shortcut, display text only
Fetch content from page in EPiServer CMS
Save
Cancel

Episerver 107

The following shortcut types are available:

- **No shortcut**. Creates a link that displays the content you have created. By selecting this, you can also reset the page after using other types of links.
- **Shortcut to page in Episerver CMS**. Links to another page on the same website. A visitor who clicks this link will be transferred to the page you have linked to, and kept within the same navigation menu structure.
- **Shortcut to page on another website.** Creates a link to an external page or to a document on the server. Remember to include the entire URL address, including "http://".
- **No shortcut, display text only.** Creates a heading with no link in the menu, without displaying any information or link to another page.
- **Fetch content from page in Episerver CMS**. Creates a link to another page from which content will be retrieved into the original page within the same navigation structure. Useful when re-using content on the website, in which case you only need to maintain it in one place.

Null values

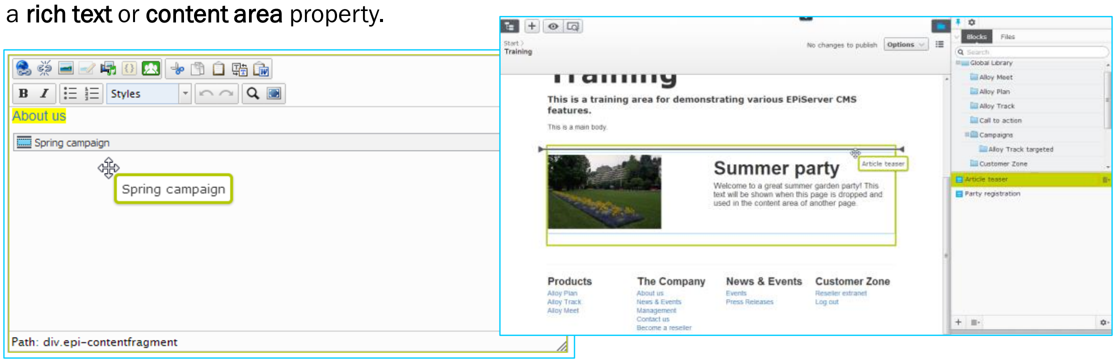
Episerver properties with an empty value are never stored in the database. If you access it from code, it will always be null – not an empty string, 0 or false as you maybe expected. Why null? It is by design and is very convenient if you want to check if something is not set by an editor or does not exist on this page. You just have to compare with null regardless of data type.

If a CMS Editor enables **Fetch content from page in Episerver CMS** then any properties that have not been set will be loaded from the page that the CMS Editor selected to fetch data from.

Module A – Getting Started with Episerver CMS – Editing content – Reusing content

Reusing content with shared blocks

Shared blocks allow the reuse of small pieces of content on the website. Drag-and-drop the content to a **rich text** or **content area** property.



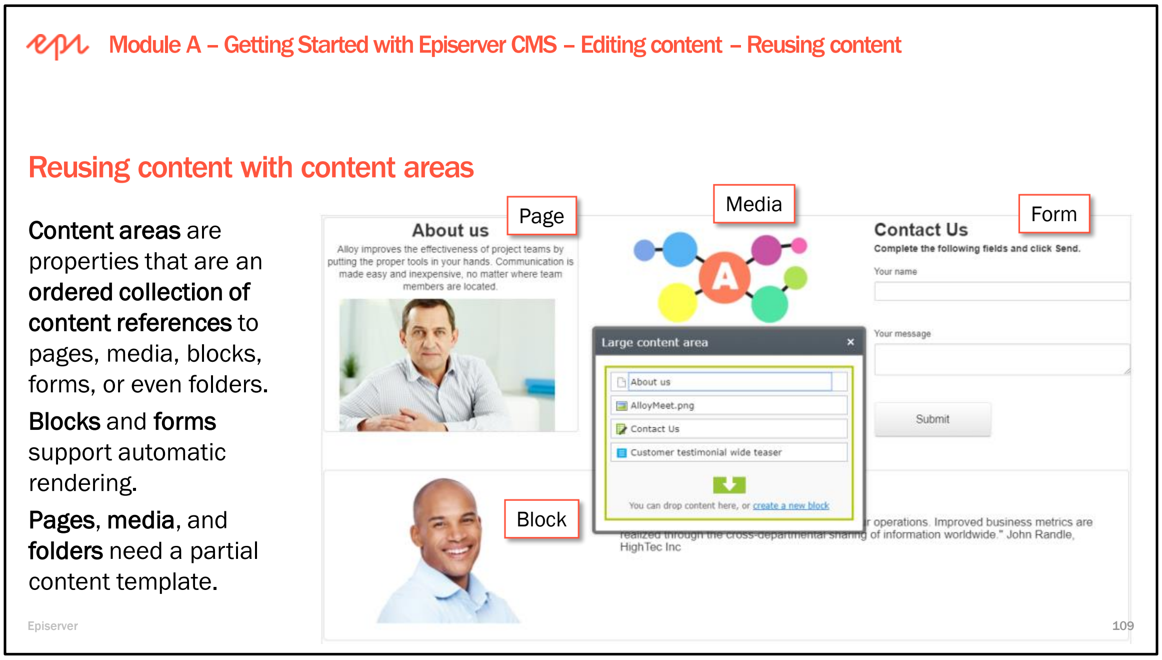
Episerver 108

Shared blocks are pieces of content that can be reused between sites, while a single original is maintained in Assets pane folder.

Editors drag and drop shared blocks from the Assets pane onto the content, into either content areas or rich text areas.

Developers are able to:

- Render blocks differently depending on where they are used, or what display options the editor has applied.
- Specify which blocks can be used in particular content areas.



Module A – Getting Started with Episerver CMS – Editing content – Reusing content

Reusing content with content areas

Content areas are properties that are an **ordered collection of content references** to pages, media, blocks, forms, or even folders.

Blocks and **forms** support automatic rendering.

Pages, **media**, and **folders** need a partial content template.

Page
Media
Form
Block

About us

Alloy improves the effectiveness of project teams by putting the proper tools in your hands. Communication is made easy and inexpensive, no matter where team members are located.

Contact Us

Complete the following fields and click Send.

Your name

Your message

Submit

Large content area

- About us
- AlloyMeet.png
- Contact Us
- Customer testimonial wide teaser

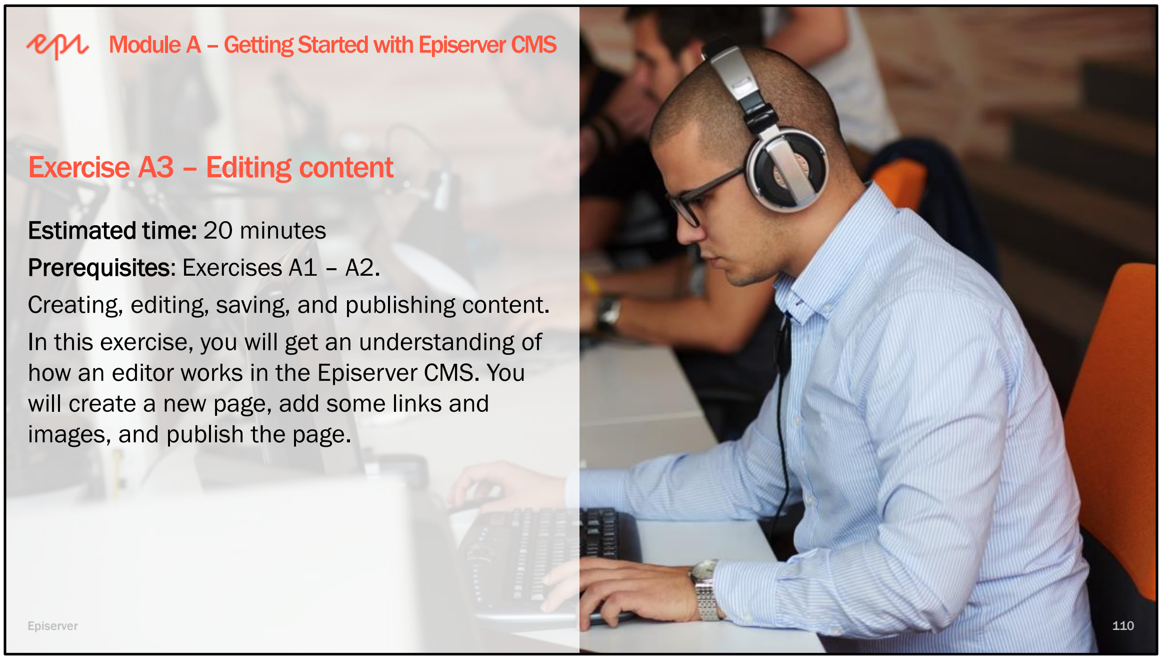
You can drop content here, or create a new block

r operations. Improved business metrics are realized through the cross-departmental sharing of information worldwide." John Randle, HighTec Inc

Episerver

109

You will learn how to create partial content templates for pages, media, and folders in **Module C: Rendering Content**.



Module A – Getting Started with Episerver CMS

Exercise A3 – Editing content

Estimated time: 20 minutes

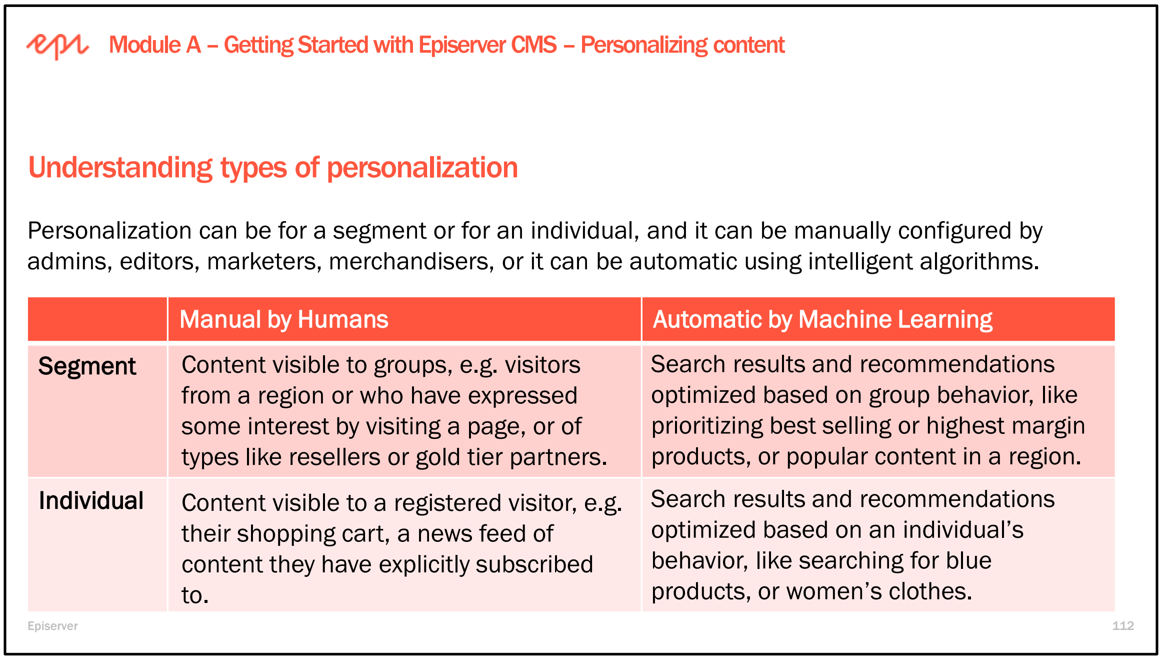
Prerequisites: Exercises A1 – A2.

Creating, editing, saving, and publishing content.

In this exercise, you will get an understanding of how an editor works in the Episerver CMS. You will create a new page, add some links and images, and publish the page.

Episerver

110



Module A – Getting Started with Episerver CMS – Personalizing content

Understanding types of personalization

Personalization can be for a segment or for an individual, and it can be manually configured by admins, editors, marketers, merchandisers, or it can be automatic using intelligent algorithms.

	Manual by Humans	Automatic by Machine Learning
Segment	Content visible to groups, e.g. visitors from a region or who have expressed some interest by visiting a page, or of types like resellers or gold tier partners.	Search results and recommendations optimized based on group behavior, like prioritizing best selling or highest margin products, or popular content in a region.
Individual	Content visible to a registered visitor, e.g. their shopping cart, a news feed of content they have explicitly subscribed to.	Search results and recommendations optimized based on an individual's behavior, like searching for blue products, or women's clothes.

Episerver 112

Personalization helps you strategically target the right products and information to the right visitors, instead of showing the same content and products to everyone.

Related content from the CMS can be shown to visitors connected to what they have previously viewed, or what they are buying on a commerce site. Related content includes news, how-to articles, technical support documents, videos, and any of the content related to the products they have purchased or expressed an interest in by visiting a marketing or information page.

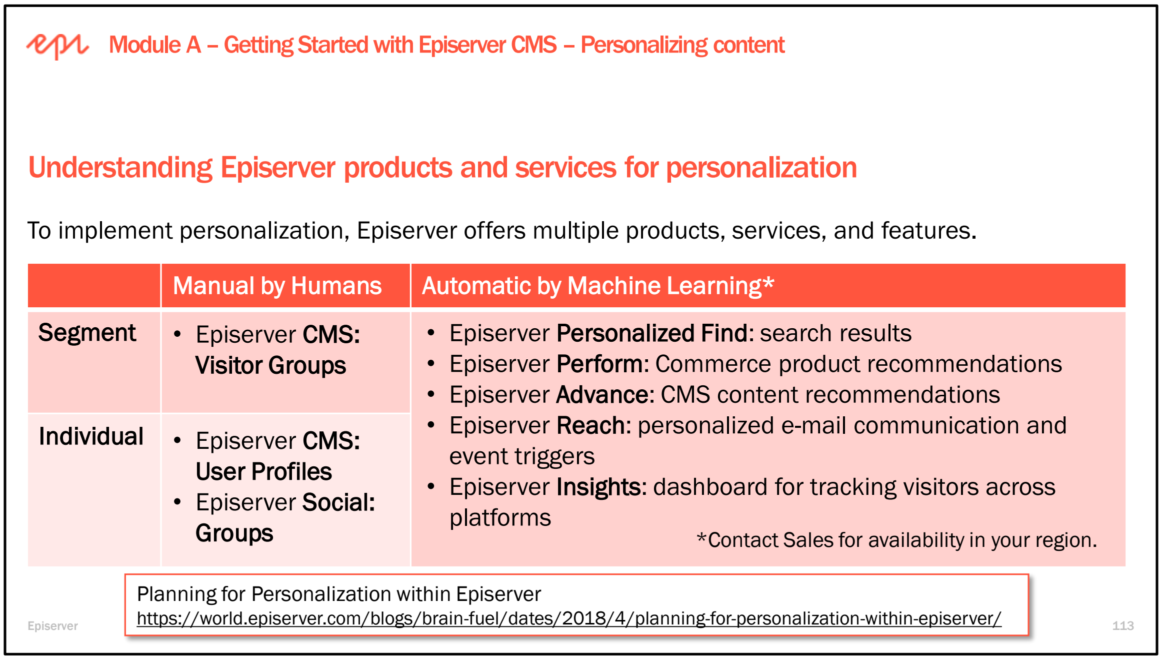
The idea of personalizing content for a visitor is to show products or information that he might want or need. The best results are gained when we combine what the visitor wants with what the company's vision of what it wants to show or sell. This might be based on best margin, the strategic importance of the brand, products that are overstocked, and so on.

In B2B scenarios, automated re-occurring messages are very common. For example, a company that sells food to restaurants. Their restaurant customers might always buy milk before 15:00, so a personalized reminder can be shown when they log in, with direct links to the most likely dairy products that they regularly order.

A wholesaler where the website customers are mainly returning customers, who order several times a week, it may not be beneficial to try to get better conversions by recommending best sellers. Instead, recommend products that they did not come to the website to buy regularly.

To make the most of personalization, you need to have a clear idea what your goal is for your website and what types of customers you have.

Getting B2B on board with online personalization
https://www.digitalcommerce360.com/2017/03/30/getting-b2b-board-online-personalization/



Module A – Getting Started with Episerver CMS – Personalizing content

Understanding Episerver products and services for personalization

To implement personalization, Episerver offers multiple products, services, and features.

	Manual by Humans	Automatic by Machine Learning*
Segment	• Episerver **CMS: Visitor Groups**	• Episerver **Personalized Find**: search results • Episerver **Perform**: Commerce product recommendations • Episerver **Advance**: CMS content recommendations • Episerver **Reach**: personalized e-mail communication and event triggers • Episerver **Insights**: dashboard for tracking visitors across platforms *Contact Sales for availability in your region.
Individual	• Episerver **CMS: User Profiles** • Episerver **Social: Groups**	

Planning for Personalization within Episerver
https://world.episerver.com/blogs/brain-fuel/dates/2018/4/planning-for-personalization-within-episerver/

Episerver 113

Episerver CMS feature: Visitor Groups

Selecting which content to show using Visitor Groups is used a lot. Various customer segments are presented Start pages tailored for them. To implement this, the first step is to define how, or if, you want to segment your customers, and do you have different policies or practices for these segments.

Visitor Groups are easy to customize for an organization's data model, and you can use this custom data as criteria in setting up Visitor Groups. If you already have a CRM system, and there are segments already defined in it defined by business practices, this segmentation can be integrated into Episerver Visitor Groups because custom Visitor Group criteria can be created by developers.

You might decide to divide your visitors into vertical segments: service stations, car repair shops, spare parts dealers, and so on. These would all have their respective Visitor Groups defined in Episerver. Different customers could see different product descriptions. For example, a retail customer (service station) could see primarily commercial data about the product while an end-user (car repair shop) would see technical details about the same product.

This means that the user experience is being optimized to fit the needs of the customer.

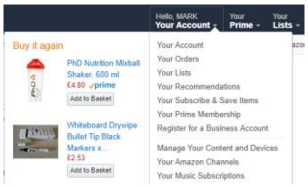
Episerver Personalization Portal

The user guide to Recommendations, Mail, and Triggers.
http://webhelp.episerver.com/Personalization/

Episerver personalization developer guides

https://world.episerver.com/documentation/developer-guides/personalization/

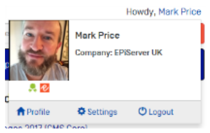
Module A – Getting Started with Episerver CMS – Personalizing content – User profiles



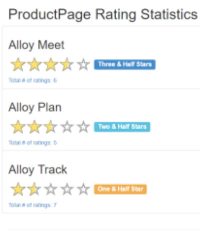
Understanding user profiles

Some websites allow visitors to register and manage their own user profile, to remember a wish list or shopping cart, or to subscribe to content notifications of updates.

Episerver CMS includes a user profile feature that can be implemented by using the `EPiServerProfile` class and ASP.NET Profile configuration.



Episerver Social includes group, user, and ratings micro-services that can be used to implement personalization for individuals and segments.

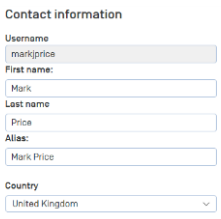


ProductPage Rating Statistics

Alloy Meet
★★★★☆ Three & Half Stars
Total # of ratings: 6

Alloy Plan
★★★☆☆ Two & Half Stars
Total # of ratings: 5

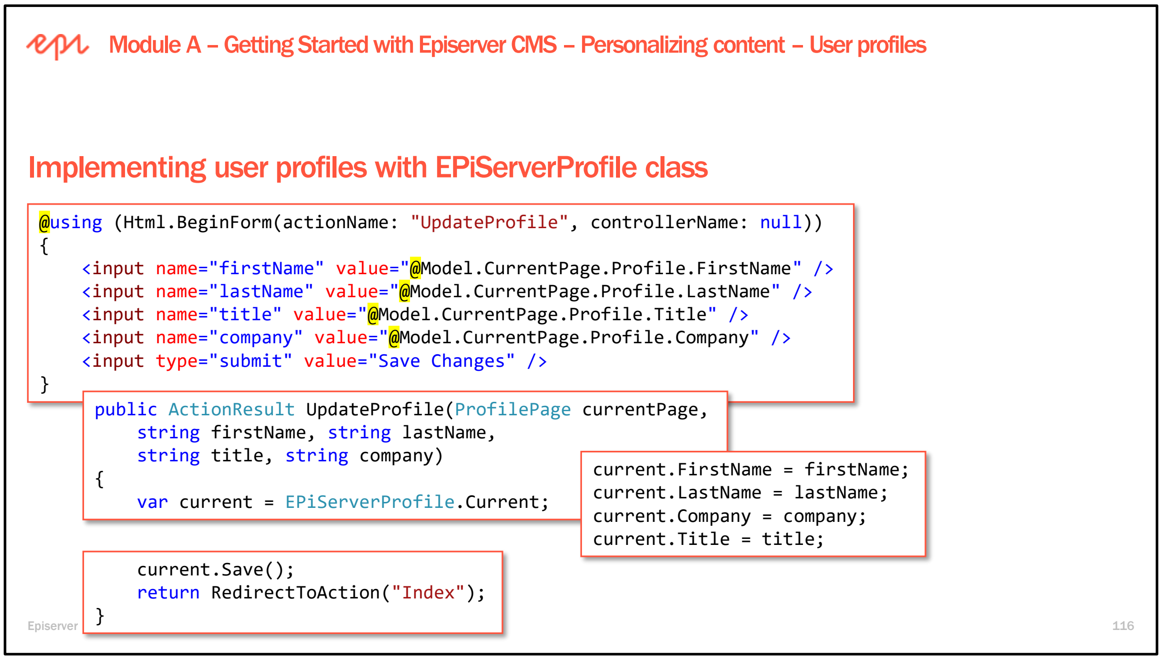
Alloy Track
★★☆☆☆ One & Half Star
Total # of ratings: 7



Contact information

Username	markjprice
First name:	Mark
Last name	Price
Alias:	Mark Price
Country	United Kingdom

Episerver



Module A – Getting Started with Episerver CMS – Personalizing content – User profiles

Implementing user profiles with EPiServerProfile class

```
@using (Html.BeginForm(actionName: "UpdateProfile", controllerName: null))
{
    <input name="firstName" value="@Model.CurrentPage.Profile.FirstName" />
    <input name="lastName" value="@Model.CurrentPage.Profile.LastName" />
    <input name="title" value="@Model.CurrentPage.Profile.Title" />
    <input name="company" value="@Model.CurrentPage.Profile.Company" />
    <input type="submit" value="Save Changes" />
}
```

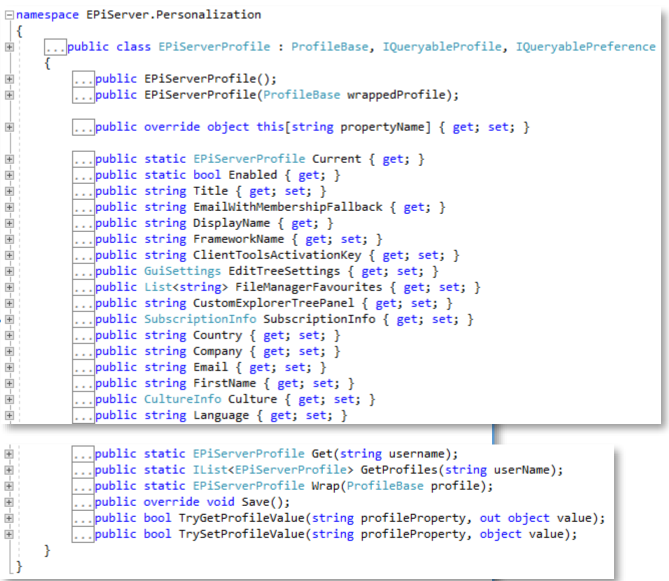
```
public ActionResult UpdateProfile(ProfilePage currentPage,
    string firstName, string lastName,
    string title, string company)
{
    var current = EPiServerProfile.Current;
```

```
current.FirstName = firstName;
current.LastName = lastName;
current.Company = company;
current.Title = title;
```

```
    current.Save();
    return RedirectToAction("Index");
}
```

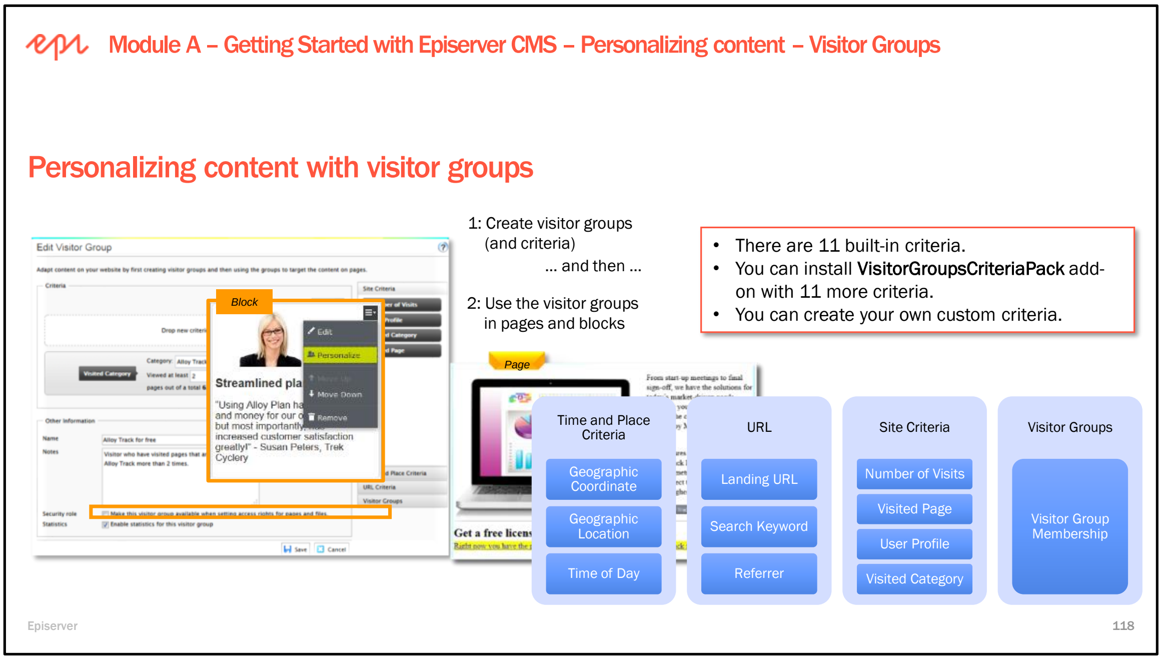
Episerver 116

Add custom properties to the ASP.NET profile configuration, and then get and set through the TryGetProfileValue and TrySetProfileValue methods:



```
namespace EPiServer.Personalization
{
    ...public class EPiServerProfile : ProfileBase, IQueryableProfile, IQueryablePreference
    {
        ...public EPiServerProfile();
        ...public EPiServerProfile(ProfileBase wrappedProfile);

        ...public override object this[string propertyName] { get; set; }

        ...public static EPiServerProfile Current { get; }
        ...public static bool Enabled { get; }
        ...public string Title { get; set; }
        ...public string EmailWithMembershipFallback { get; }
        ...public string DisplayName { get; }
        ...public string FrameworkName { get; set; }
        ...public string ClientToolsActivationKey { get; set; }
        ...public GuiSettings EditTreeSettings { get; set; }
        ...public List<string> FileManagerFavourites { get; set; }
        ...public string CustomExplorerTreePanel { get; set; }
        ...public SubscriptionInfo SubscriptionInfo { get; set; }
        ...public string Country { get; set; }
        ...public string Company { get; set; }
        ...public string Email { get; set; }
        ...public string FirstName { get; set; }
        ...public CultureInfo Culture { get; set; }
        ...public string Language { get; set; }

        ...public static EPiServerProfile Get(string username);
        ...public static IList<EPiServerProfile> GetProfiles(string userName);
        ...public static EPiServerProfile Wrap(ProfileBase profile);
        ...public override void Save();
        ...public bool TryGetProfileValue(string profileProperty, out object value);
        ...public bool TrySetProfileValue(string profileProperty, object value);
    }
}
```

```
<profile defaultProvider="DefaultProfileProvider">
  <properties>
    <add name="Address" type="System.String" />
    <add name="ZipCode" type="System.String" />
    <add name="Locality" type="System.String" />
    <add name="Email" type="System.String" />
    <add name="FirstName" type="System.String" />
    <add name="LastName" type="System.String" />
    <add name="Language" type="System.String" />
    <add name="Country" type="System.String" />
    <add name="Company" type="System.String" />
    <add name="Title" type="System.String" />
    <add name="CustomExplorerTreePanel" type="System.String" />
    <add name="FileManagerFavourites" type="System.Collections.Generic.List`1[System.String]" />
    <add name="EditTreeSettings" type="EPiServer.Personalization.GuiSettings, EPiServer" />
    <add name="ClientToolsActivationKey" type="System.String" />
    <add name="FrameworkName" type="System.String" />
  </properties>
  <providers>
    <add name="DefaultProfileProvider" type="System.Web.Providers.DefaultProfileProvider, ..."
         connectionStringName="EPiServerDB" applicationName="/" />
  </providers>
</profile>
```



Module A – Getting Started with Episerver CMS – Personalizing content – Visitor Groups

Personalizing content with visitor groups

1: Create visitor groups (and criteria)

... and then ...

2: Use the visitor groups in pages and blocks

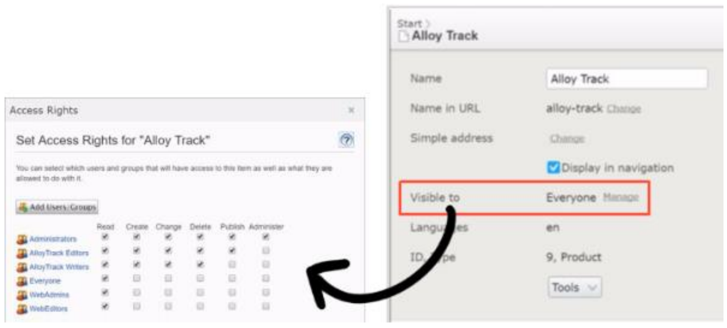
- There are 11 built-in criteria.
- You can install **VisitorGroupsCriteriaPack** add-on with 11 more criteria.
- You can create your own custom criteria.

Episerver 118

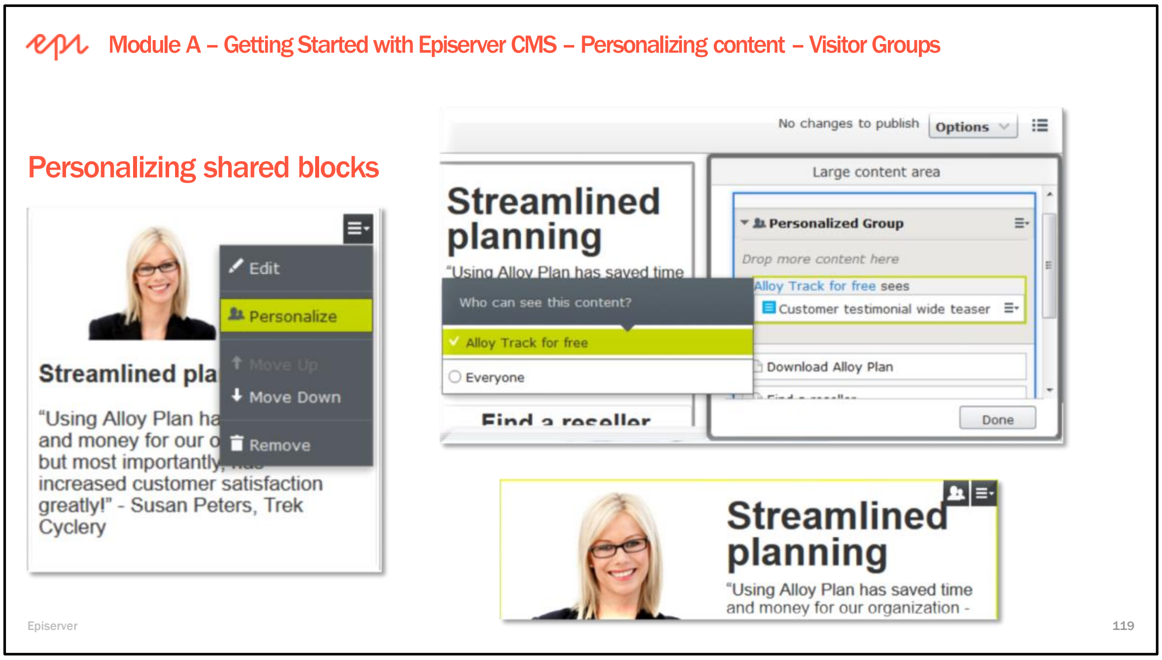
- Personalization can be done for partial pages and shared blocks in a content area, and for any content in a rich-text editor (property of type XHTML string)
- Personalization is based on Visitor Groups, which are defined as a number of criteria
- Visitor Group Criteria can be defined with logical rules or with fuzzy scoring
- You can develop your own criteria and plug it into the existing criteria collection
- You can preview any page or block as any visitor group by using the view toggle setting
- Built-in reports on which groups visit your site: Visitor Groups Statistics gadget
- Visitor groups can be used as Security roles when setting access rights

Personalization of Pages

Like Blocks, Pages can be personalized. An Editor can edit a page, and click Visible To, and then remove Everyone's Read access rights, and add a Visitor Group and give it Read access rights.



Name: Alloy Track
Name in URL: alloy-track Change
Simple address: Change
Display in navigation
Visible to: Everyone Manage
Languages: en
ID, Type: 9, Product



Module A – Getting Started with Episerver CMS – Personalizing content – Visitor Groups

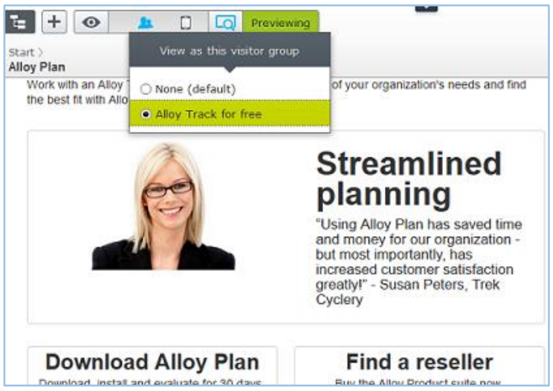
Personalizing shared blocks

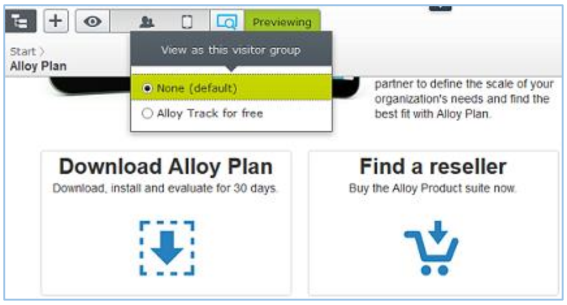
Edit
Personalize
Move Up
Move Down
Remove
Streamlined planning
"Using Alloy Plan has saved time and money for our organization - but most importantly, has increased customer satisfaction greatly!" - Susan Peters, Trek Cyclery
No changes to publish
Options
Large content area
Personalized Group
Drop more content here
Alloy Track for free sees
Customer testimonial wide teaser
Who can see this content?
Alloy Track for free
Everyone
Download Alloy Plan
Find a reseller
Done

Episerver 119

Previewing personalized content

Given the above scenario, this would be the result of previewing the page that the personalized block is used on and toggle the view setting to see it as the different visitor groups:





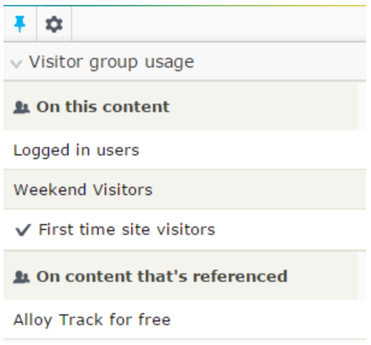
Previewing
View as this visitor group
None (default)
Alloy Track for free
Start
Alloy Plan
Download Alloy Plan
Download, install and evaluate for 30 days.
Find a reseller
Buy the Alloy Product suite now.

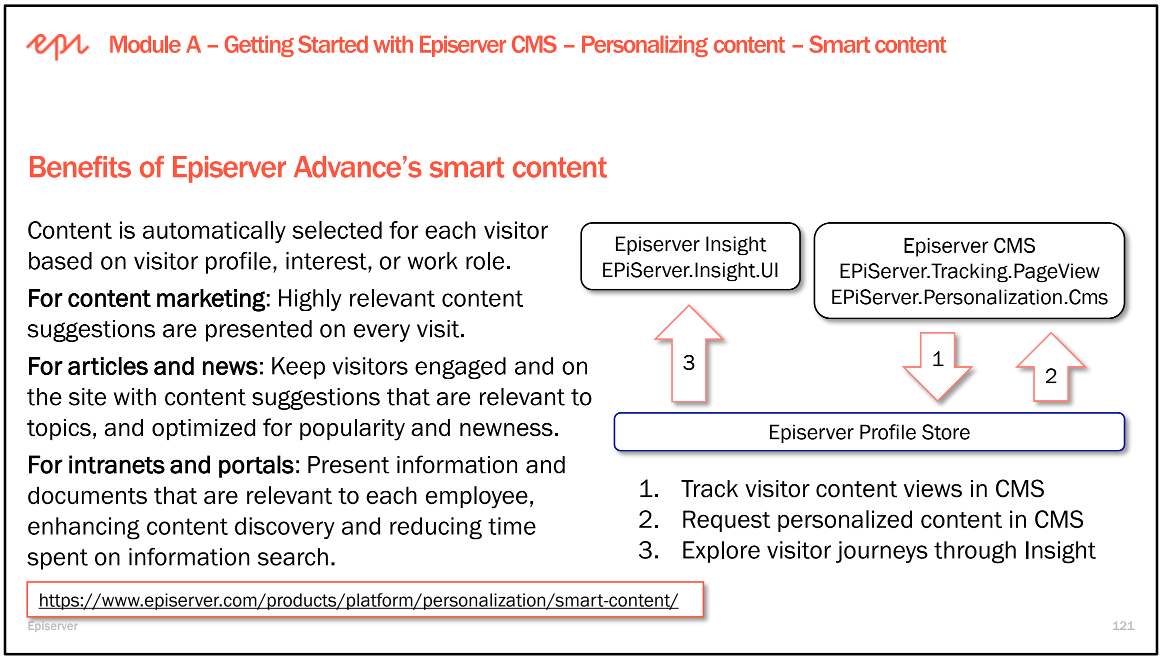
Visitor Group Usage Viewer by David Knipe

https://www.david-tec.com/2017/12/visitor-group-usage-viewer-for-episerver-11/

The visitor group usage viewer adds a new component that shows the visitor groups that are used on the current content item when in Edit view.

```
Install-Package VisitorGroupUsage
```





Module A – Getting Started with Episerver CMS – Personalizing content – Smart content

Benefits of Episerver Advance's smart content

Content is automatically selected for each visitor based on visitor profile, interest, or work role.

For content marketing: Highly relevant content suggestions are presented on every visit.

For articles and news: Keep visitors engaged and on the site with content suggestions that are relevant to topics, and optimized for popularity and newness.

For intranets and portals: Present information and documents that are relevant to each employee, enhancing content discovery and reducing time spent on information search.

1. Track visitor content views in CMS
2. Request personalized content in CMS
3. Explore visitor journeys through Insight

https://www.episerver.com/products/platform/personalization/smart-content/

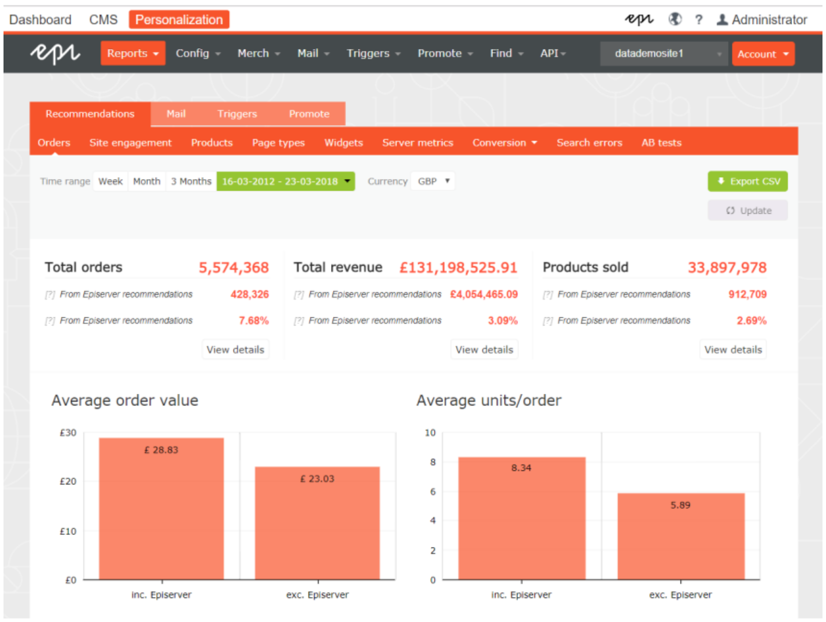
Episerver 121

Instead of time-consuming and error-prone rules-based personalization, Episerver Advance makes use of autonomous personalization – where machine learning based algorithms are used to inject the right content at the right time for every single visitor.

Personalized content on first page view: Episerver uses contextual data, such as ad clicks, geolocation, and organization affiliation to present relevant content on start pages, content listings and landing pages.

Automatically surfaced information: For many websites and portals, it is deep content that is most likely to be relevant to a visitor. Episerver uses tagging and content filtering to surface content that is more likely to serve the visitor's needs.

Interactive content drill-down: From the first content selection, visitors are quickly able to drill down into large content repositories thanks to Episerver's intuitive guided navigation.



Time range: Week | Month | 3 Months | 16-03-2012 - 23-03-2018 — Currency: GBP

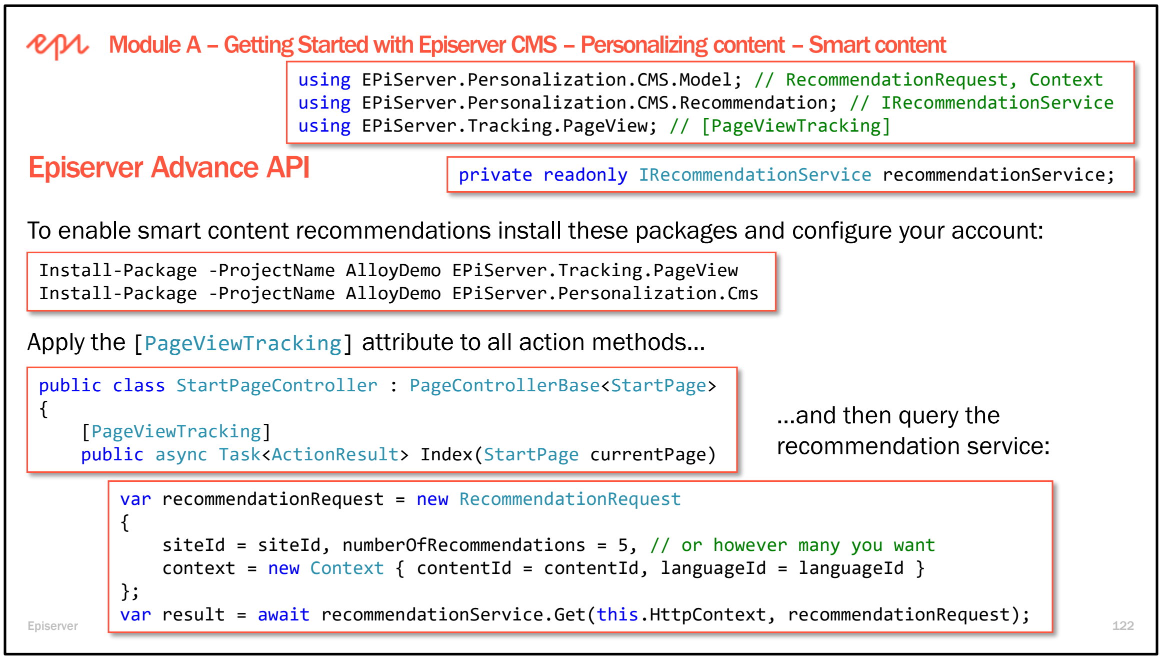
Total orders	5,574,368	Total revenue	£131,198,525.91	Products sold	33,897,978
[?] From Episerver recommendations	428,326	[?] From Episerver recommendations	£4,054,465.09	[?] From Episerver recommendations	912,709
[?] From Episerver recommendations	7.68%	[?] From Episerver recommendations	3.09%	[?] From Episerver recommendations	2.69%

Average order value

	inc. Episerver	exc. Episerver
Average order value	£ 28.83	£ 23.03

Average units/order

	inc. Episerver	exc. Episerver
Average units/order	8.34	5.89



Module A – Getting Started with Episerver CMS – Personalizing content – Smart content

```
using EPiServer.Personalization.CMS.Model; // RecommendationRequest, Context
using EPiServer.Personalization.CMS.Recommendation; // IRecommendationService
using EPiServer.Tracking.PageView; // [PageViewTracking]
```

Episerver Advance API

```
private readonly IRecommendationService recommendationService;
```

To enable smart content recommendations install these packages and configure your account:

```
Install-Package -ProjectName AlloyDemo EPiServer.Tracking.PageView
Install-Package -ProjectName AlloyDemo EPiServer.Personalization.Cms
```

Apply the [PageViewTracking] attribute to all action methods…

```
public class StartPageController : PageControllerBase<StartPage>
{
    [PageViewTracking]
    public async Task<ActionResult> Index(StartPage currentPage)
```

…and then query the recommendation service:

```
var recommendationRequest = new RecommendationRequest
{
    siteId = siteId, numberOfRecommendations = 5, // or however many you want
    context = new Context { contentId = contentId, languageId = languageId }
};
var result = await recommendationService.Get(this.HttpContext, recommendationRequest);
```

Episerver 122

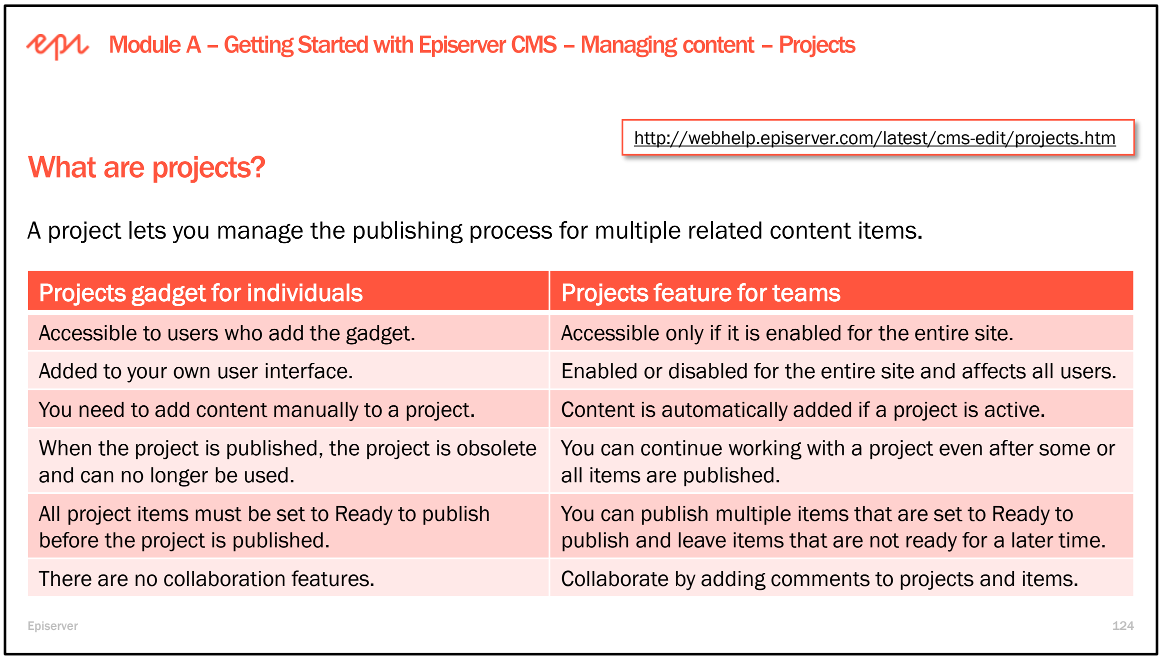
https://world.episerver.com/documentation/developer-guides/personalization/advance-api/

```
public class StartPageController : PageControllerBase<StartPage>
{
    [PageViewTracking]
    public async Task<ActionResult> Index(StartPage currentPage)
    {
        var siteId = SiteDefinition.Current.Id.ToString();
        var contentId = currentPage.ContentGuid.ToString();
        var languageId = currentPage.Language.Name;

        var recommendationRequest = new RecommendationRequest
        {
            siteId = siteId,
            context = new Context { contentId = contentId, languageId = languageId },
            numberOfRecommendations = 5 //or another number as needed
        };

        var result = await recommendationService.Get(this.HttpContext, recommendationRequest);
```

Example block for displaying recommended content on a page:
https://github.com/episerver/ContentRecommendationBlock

Episerver Insight and the Profile Store – Basic Implementation
https://blog.nicolaayan.com/2018/04/episerver-insight-profile-store-implementation/

A developers guide to Machine Learning - Tess Ferrandez-Norlander
https://www.youtube.com/watch?v=hjpUHZY5-18



Module A – Getting Started with Episerver CMS – Managing content – Projects

http://webhelp.episerver.com/latest/cms-edit/projects.htm

What are projects?

A project lets you manage the publishing process for multiple related content items.

Projects gadget for individuals	Projects feature for teams
Accessible to users who add the gadget.	Accessible only if it is enabled for the entire site.
Added to your own user interface.	Enabled or disabled for the entire site and affects all users.
You need to add content manually to a project.	Content is automatically added if a project is active.
When the project is published, the project is obsolete and can no longer be used.	You can continue working with a project even after some or all items are published.
All project items must be set to Ready to publish before the project is published.	You can publish multiple items that are set to Ready to publish and leave items that are not ready for a later time.
There are no collaboration features.	Collaborate by adding comments to projects and items.

Episerver 124

Project feature – media assets
Uploaded media that is associated with a project is not published until it is manually published or published via scheduling, even if the automatic publish for media assets setting is turned on.

Module A – Getting Started with Episerver CMS – Managing content – Projects

Projects feature for teams

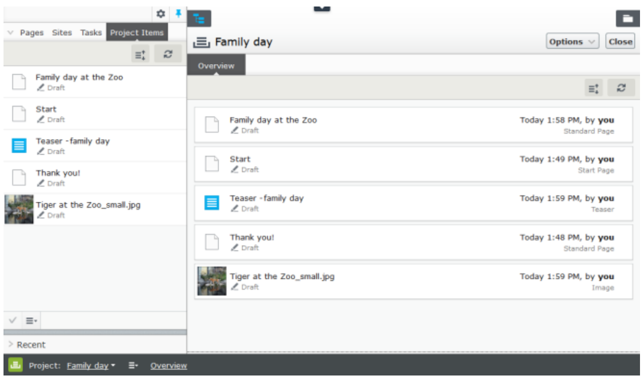
Projects feature is either enabled or disabled for the entire site and affects all users, it is enabled by default.

The project interface consists of: the project bar, the Overview page, and the Project Items tab in the Navigation pane.

Editing actions automatically associate a content item with an active project.

To disable Projects feature, you need to add an entry to appSettings in Web.config:

http://webhelp.episerver.com/latest/cms-edit/projects-feature.htm

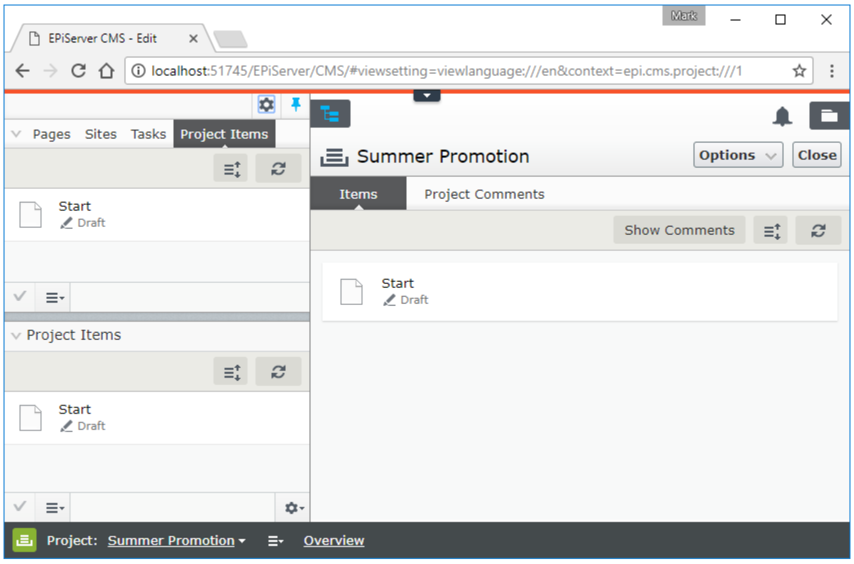


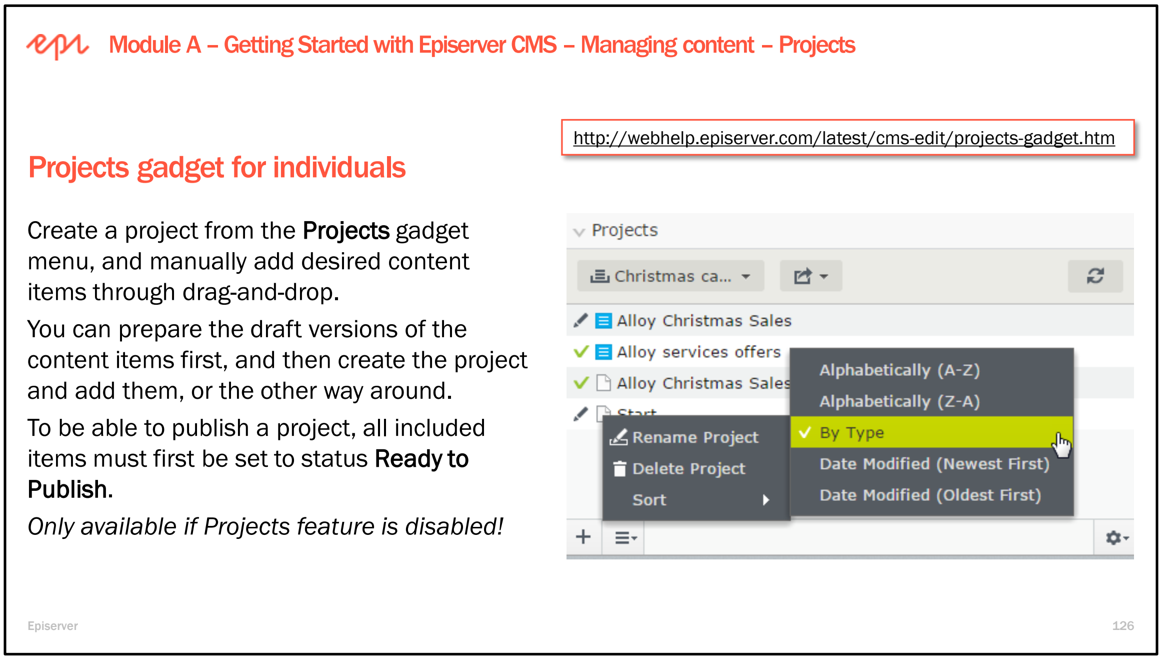
```
<appSettings>
  <add key="episerver:ui:IsProjectModeEnabled" value="false" />
```

Episerver 125

http://world.episerver.com/documentation/developer-guides/CMS/projects/

Project Items tab and **Project Items** gadget show a simplified view of the Projects feature **Overview**, as shown in the following screenshot:





Module A – Getting Started with Episerver CMS – Managing content – Projects

http://webhelp.episerver.com/latest/cms-edit/projects-gadget.htm

Projects gadget for individuals

Create a project from the **Projects** gadget menu, and manually add desired content items through drag-and-drop.

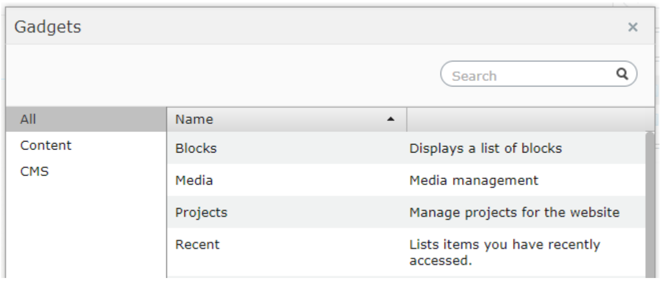
You can prepare the draft versions of the content items first, and then create the project and add them, or the other way around.

To be able to publish a project, all included items must first be set to status **Ready to Publish**.

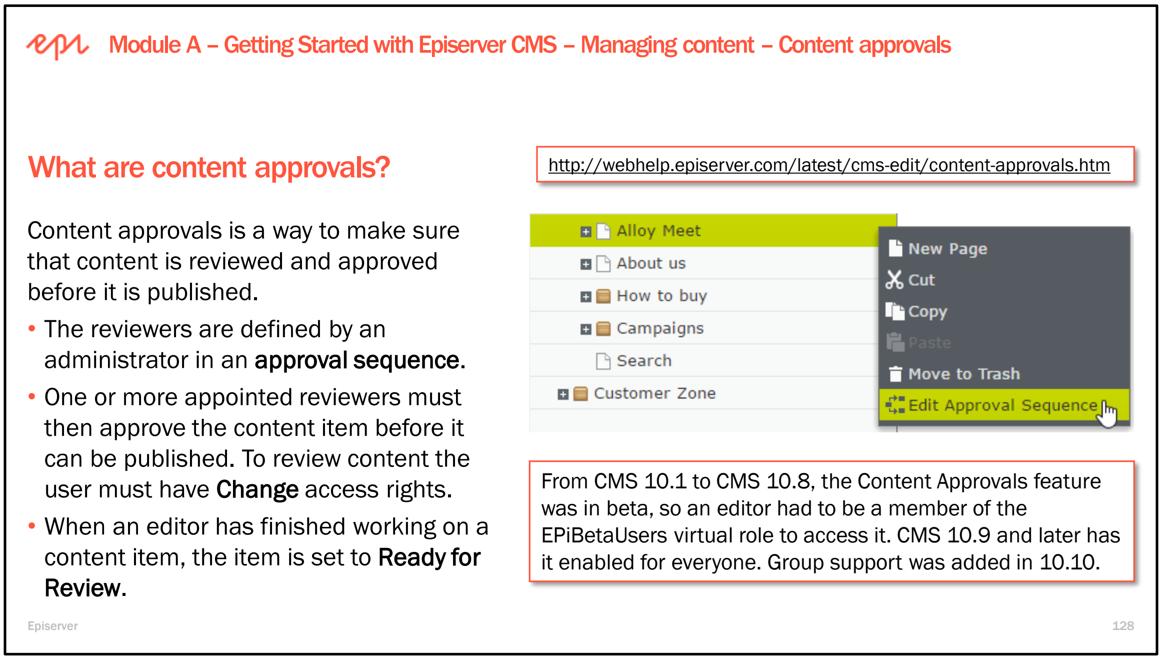
Only available if Projects feature is disabled!

Episerver 126

The option to add the **Projects** gadget, as shown in the following screenshot, will only be visible if Projects feature is disabled!



Name	
Blocks	Displays a list of blocks
Media	Media management
Projects	Manage projects for the website
Recent	Lists items you have recently accessed.



Module A – Getting Started with Episerver CMS – Managing content – Content approvals

What are content approvals?

http://webhelp.episerver.com/latest/cms-edit/content-approvals.htm

Content approvals is a way to make sure that content is reviewed and approved before it is published.

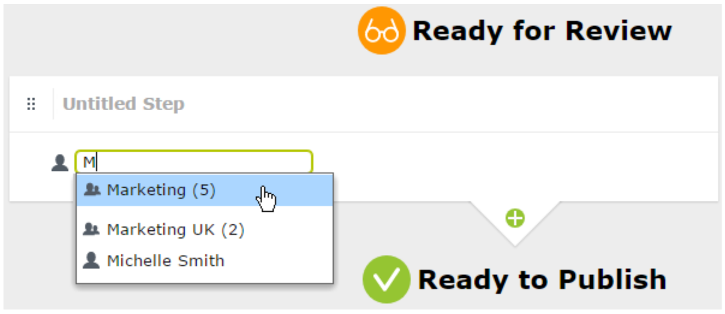
- The reviewers are defined by an administrator in an **approval sequence**.
- One or more appointed reviewers must then approve the content item before it can be published. To review content the user must have **Change** access rights.
- When an editor has finished working on a content item, the item is set to **Ready for Review**.

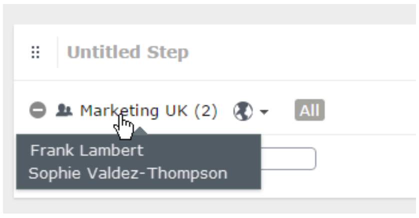
From CMS 10.1 to CMS 10.8, the Content Approvals feature was in beta, so an editor had to be a member of the EPiBetaUsers virtual role to access it. CMS 10.9 and later has it enabled for everyone. Group support was added in 10.10.

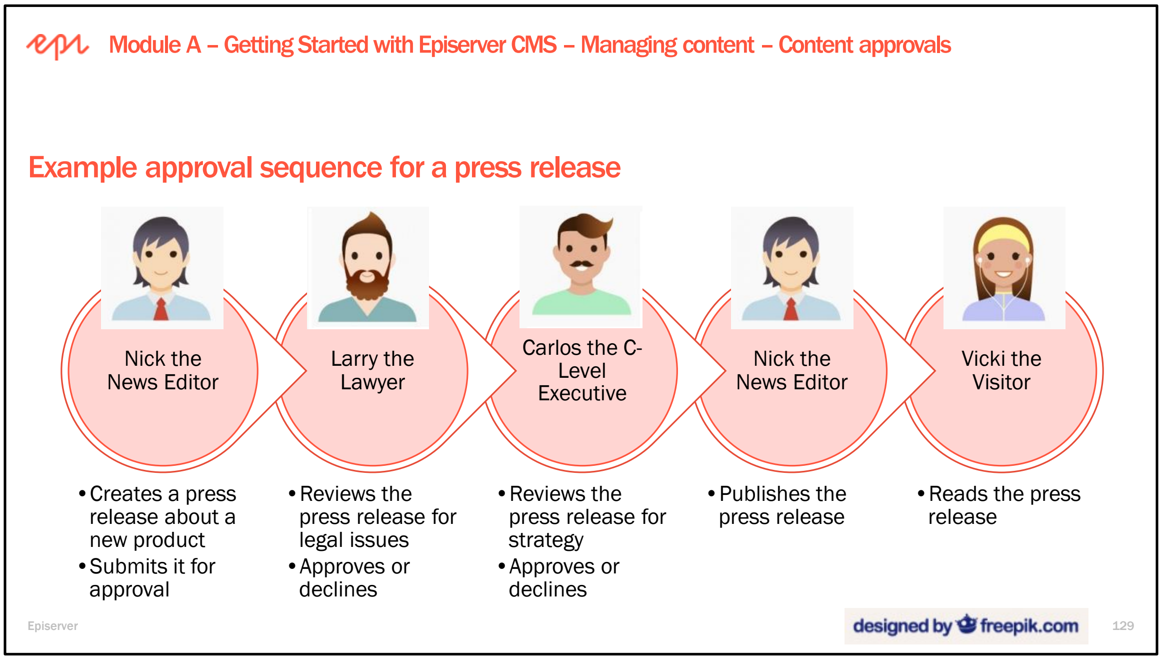
Episerver 128

Sequences and reviewers
An approval sequence can be set up with **any number of approval steps** and **any number of reviewers** in each step. The sequence is set up by an administrator, who also defines, for each step individually, who can approve a content item.
It is possible to have only one person as reviewer in a step, but it is recommended to have at least two (per language) in case one of them is unavailable.
As soon as one of the reviewers in a step approves the content, that step is considered completed and the item moves to the next step in the approval sequence.
When a content item enters an approval step, the reviewers in that step are **notified by email and in the user interface** that they have an item to approve.
When the content has been approved in all steps, it is automatically set as **Ready to Publish**, and anyone with publishing rights can publish it.

Group/role support was added in CMS 10.10 and later
We recommend that you use small groups because when you assign a group with lots of members, there is a tendency for everyone in that group to assume that someone else will approve the content. It will also get annoying for all those group members if you have email notifications enabled, so use common sense.
http://world.episerver.com/blogs/john-philip-johansson/dates/2017/5/introducing-grouprole-support-in-content-approvals/







Module A – Getting Started with Episerver CMS – Managing content – Content approvals

Example approval sequence for a press release

Nick the News Editor
- Creates a press release about a new product
- Submits it for approval

Larry the Lawyer
- Reviews the press release for legal issues
- Approves or declines

Carlos the C-Level Executive
- Reviews the press release for strategy
- Approves or declines

Nick the News Editor
- Publishes the press release

Vicki the Visitor
- Reads the press release

Episerver 129

Images created by Freepik:
http://www.freepik.com/free-vector/nice-people-avatars-in-flat-design_844761.htm

Module A – Getting Started with Episerver CMS – Managing content – Content approvals

http://webhelp.episerver.com/latest/cms-admin/managing-approval-sequences.htm

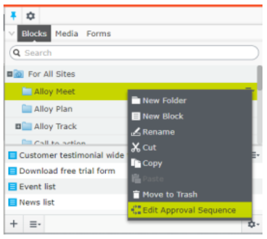
Content approvals on assets

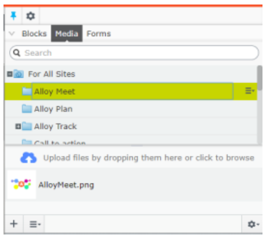
Assets, such as blocks and media (and also forms and catalogues if you have Episerver Forms and Episerver Commerce installed), cannot have individual approval sequences. Instead, the content approval sequence is set on each assets folder, and **all assets in a folder have the same approval sequence set.**

The Blocks and Media folders in the assets pane are actually the same folders in the software and **share the same content approval sequences**; the Blocks and Media tabs in the assets pane are merely a way of filtering out blocks if you are in the Media tab and vice versa.

Forms and Commerce catalogues have their own structures.

Editors can drag and drop an unapproved image into a rich-text property but visitors will not see it because the `<img src="" />` returns a 404.





Episerver 130

Reviewers, roles, languages, and required comments on approve or decline
It is only the role name that is part of the definition, not the users in the role. The validation to see if a user is part of a role is made at the moment it is needed. This means that a user can be added to a role or removed from one and that will affect an already started approval.
To avoid content getting stuck in an approval step if a reviewer is unable to approve, it is recommended that you have at least two reviewers (per language) in a step.

- An administrator can always approve and publish a page.
- Administrators and the editor who started the approval sequence can cancel the approval sequence at any step.
- If you have content in more than one language, each language must have at least one reviewer.
- The administrator decides whether a reviewer can approve content for all languages or for specific languages. Therefore, it is possible to have different reviewers for different languages.
- Administrators can require comments on Approve and/or Decline.

http://world.episerver.com/blogs/Khurram-Hanif/Dates/2017/3/content-approvals---require-comments-for-decline-and-approve/

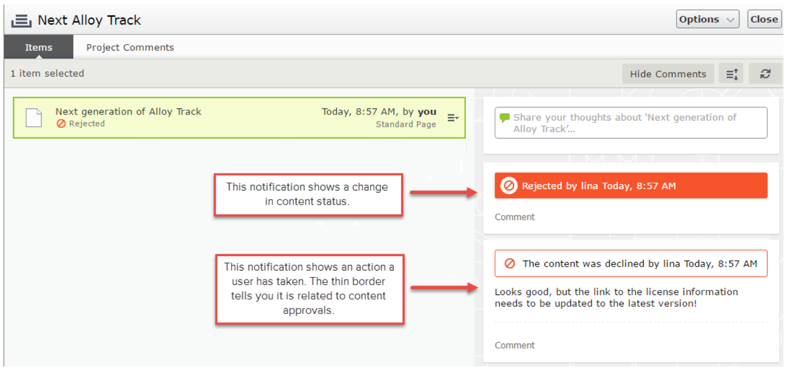
Module A – Getting Started with Episerver CMS – Managing content – Content approvals

http://webhelp.episerver.com/17-3/cms-edit/content-approvals.htm

Content approvals on projects

Content Approvals work with Projects but only on a per-content basis, and not for the whole Project.

When approving or declining content you may have to give a reason for your action. This comment will be visible in the project overview if the content item in review is associated with a project.

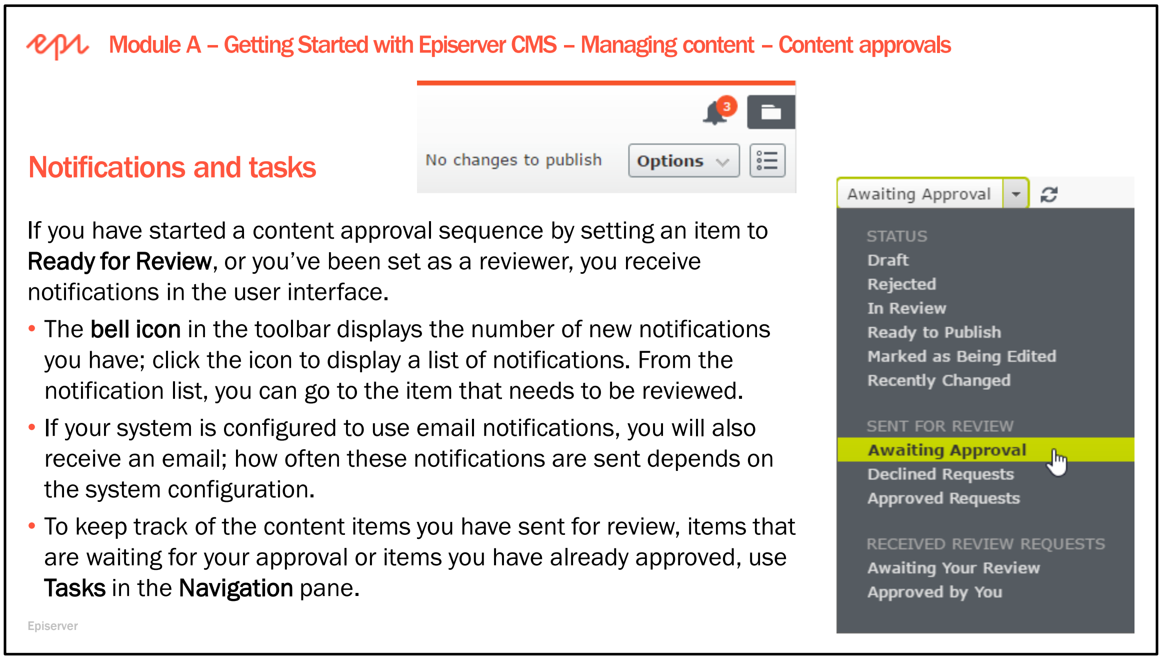

Next Alloy Track
Options
Close
Items
Project Comments
1 item selected
Hide Comments
Next generation of Alloy Track
Rejected
Today, 8:57 AM, by you
Standard Page
Share your thoughts about 'Next generation of Alloy Track'...
Rejected by lina Today, 8:57 AM
Comment
The content was declined by lina Today, 8:57 AM
Comment

Looks good, but the link to the license information needs to be updated to the latest version!

This notification shows a change in content status.

This notification shows an action a user has taken. The thin border tells you it is related to content approvals.

Episerver 131



Module A – Getting Started with Episerver CMS – Managing content – Content approvals

Notifications and tasks

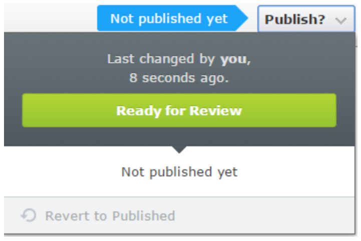
No changes to publish
Options
Awaiting Approval
STATUS
Draft
Rejected
In Review
Ready to Publish
Marked as Being Edited
Recently Changed
SENT FOR REVIEW
Awaiting Approval
Declined Requests
Approved Requests
RECEIVED REVIEW REQUESTS
Awaiting Your Review
Approved by You

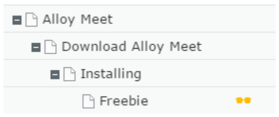
If you have started a content approval sequence by setting an item to **Ready for Review**, or you've been set as a reviewer, you receive notifications in the user interface.

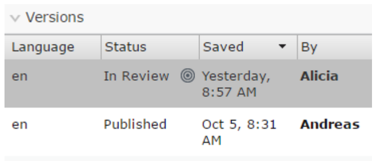
- The **bell icon** in the toolbar displays the number of new notifications you have; click the icon to display a list of notifications. From the notification list, you can go to the item that needs to be reviewed.
- If your system is configured to use email notifications, you will also receive an email; how often these notifications are sent depends on the system configuration.
- To keep track of the content items you have sent for review, items that are waiting for your approval or items you have already approved, use **Tasks** in the **Navigation** pane.

Episerver

Adding a very large group or role to an approval step will result in notifications being sent out to all the current members of the group on every step in a content approval sequence. This can have negative effects on the performance of the server, as well as probably annoy users. Therefore, we have added a new configuration setting, "ApprovalStepRoleUserLimit", that defaults to 100 users per group. Adding a group larger than this setting to a content approval sequence, triggers a validation warning, and it also limits the amount of notifications per group.

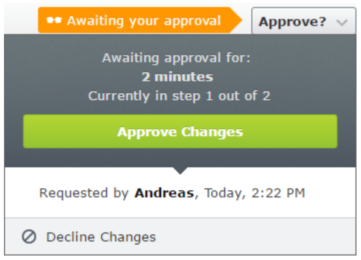


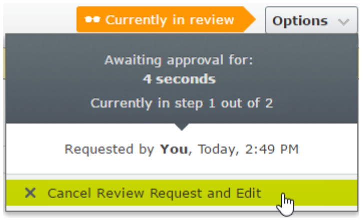




Versions

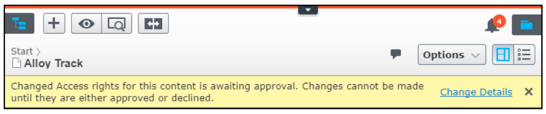
Language	Status	Saved	By
en	In Review	Yesterday, 8:57 AM	Alicia
en	Published	Oct 5, 8:31 AM	Andreas


Awaiting your approval
Approve?
Awaiting approval for: 2 minutes
Currently in step 1 out of 2
Approve Changes
Requested by Andreas, Today, 2:22 PM
Decline Changes


Currently in review
Options
Awaiting approval for: 4 seconds
Currently in step 1 out of 2
Requested by You, Today, 2:49 PM
Cancel Review Request and Edit

Module A – Getting Started with Episerver CMS – Managing content – Content approvals

Change approvals


Start
Alloy Track
Changed Access rights for this content is awaiting approval. Changes cannot be made until they are either approved or declined.

Ensure changes that affect the website are reviewed and approved before they are applied, including:

- changes to access rights,
- language settings for fallback and replacement languages,
- content expiration dates, and moving pages and blocks in the structure.

```
Install-Package -ProjectName AlloyDemo EPiServer.ChangeApproval
```

When all steps in the approval sequence have been approved, the change is immediately applied.

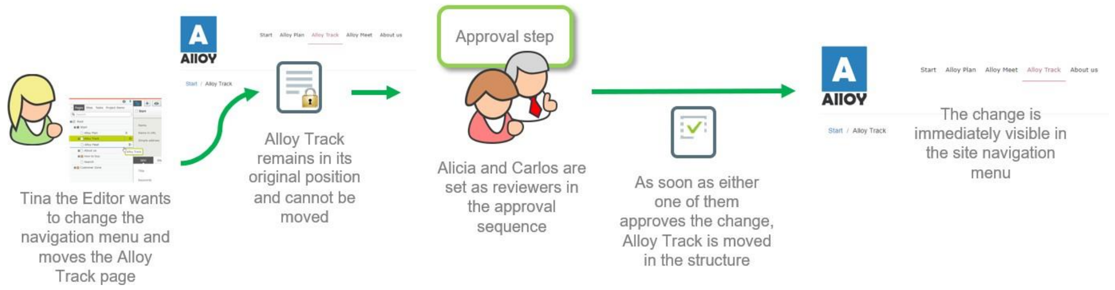
Change approvals use the same approval sequences as content approvals. This means that if you have set a content approval sequence for a content item, the same sequence and reviewers are used when changes are performed on that content item.

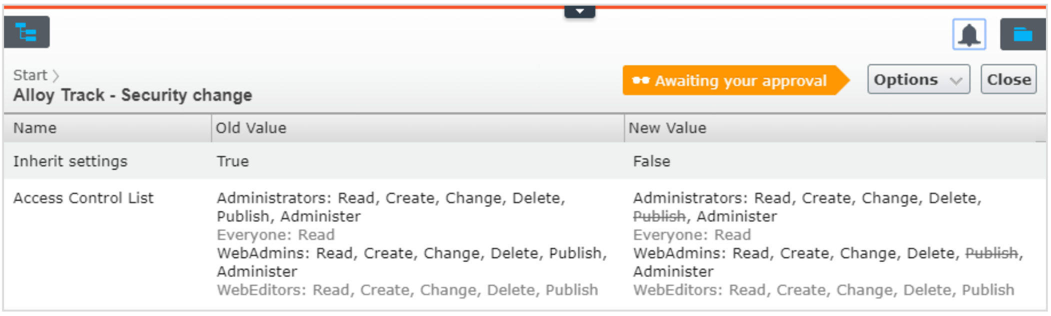
Change approvals affects all versions of the page or block, so while one change is in review, you cannot perform any of the changes that must be approved before being applied.

Episerver 133

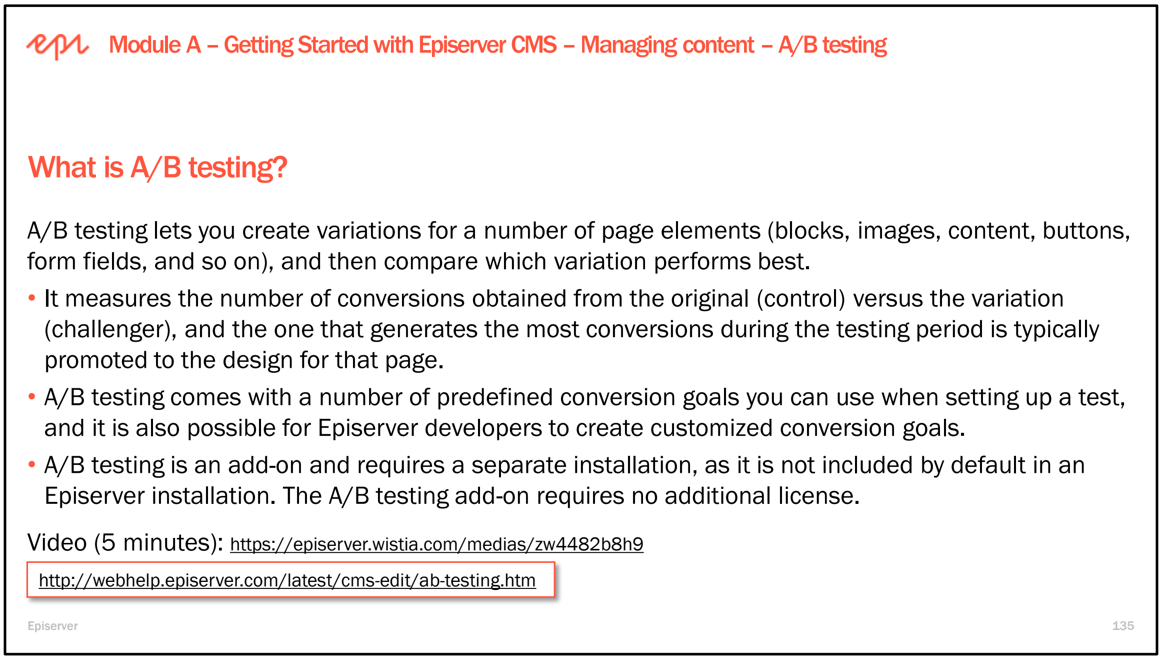
Example change approval

Tina has been asked to change the order of the products in the Alloy top navigation menu. Since the navigation menu order is controlled by the order of the pages in the page tree, she moves the Alloy Track page in the page tree. The Alloy Track page has a content approval sequence defined so the page is not immediately moved, and Tina sees a message that the move of the page is awaiting approval.
The approval sequence is set up with one step, and both reviewers, Alicia and Carlos, are notified in the user interface when they log in that Tina has moved Alloy Track and that they need to approve that move. Carlos now approves the move and the page is moved immediately and the top navigation menu is updated on the website. If Carlos had instead declined, the page would have remained in its original position.


Tina the Editor wants to change the navigation menu and moves the Alloy Track page
Alloy
Alloy Track remains in its original position and cannot be moved
Approval step
Alicia and Carlos are set as reviewers in the approval sequence
As soon as either one of them approves the change, Alloy Track is moved in the structure
Alloy
The change is immediately visible in the site navigation menu


Start
Alloy Track - Security change

Name	Old Value	New Value
Inherit settings	True	False
Access Control List	Administrators: Read, Create, Change, Delete, Publish, Administer Everyone: Read WebAdmins: Read, Create, Change, Delete, Publish, Administer WebEditors: Read, Create, Change, Delete, Publish	Administrators: Read, Create, Change, Delete, ~~Publish~~, Administer Everyone: Read WebAdmins: Read, Create, Change, Delete, ~~Publish~~, Administer WebEditors: Read, Create, Change, Delete, Publish



Module A – Getting Started with Episerver CMS – Managing content – A/B testing

What is A/B testing?

A/B testing lets you create variations for a number of page elements (blocks, images, content, buttons, form fields, and so on), and then compare which variation performs best.

- It measures the number of conversions obtained from the original (control) versus the variation (challenger), and the one that generates the most conversions during the testing period is typically promoted to the design for that page.
- A/B testing comes with a number of predefined conversion goals you can use when setting up a test, and it is also possible for Episerver developers to create customized conversion goals.
- A/B testing is an add-on and requires a separate installation, as it is not included by default in an Episerver installation. The A/B testing add-on requires no additional license.

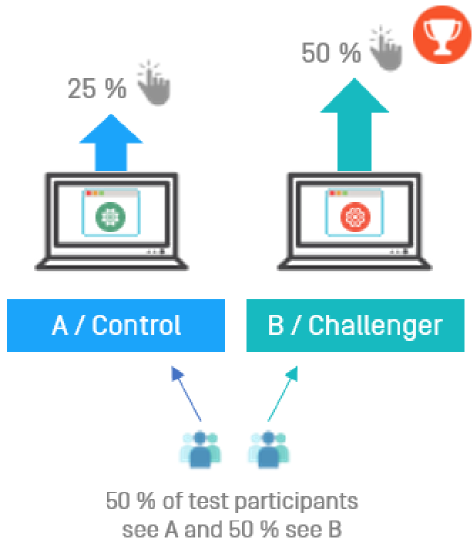
Video (5 minutes): https://episerver.wistia.com/medias/zw4482b8h9

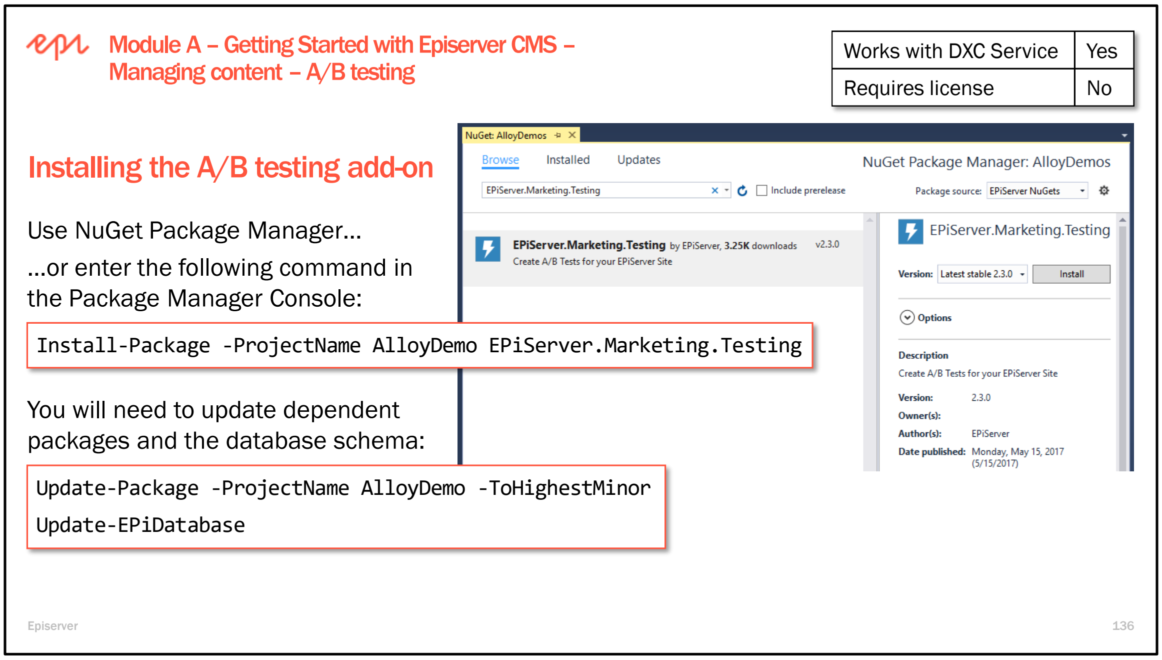
http://webhelp.episerver.com/latest/cms-edit/ab-testing.htm

Episerver 135

How A/B testing works

1. When a visitor views content, the visitor sees either the original (A) or the variation (B). A/B testing logs which version the visitor sees. If they return to the content, the visitor sees the same version. If they clear cookies, and revisit the content, they are considered a new visitor in the test.
2. If a visitor clicks on the advertisement, the target page appears and A/B testing logs the action as a conversion.
3. When the test completes, the version that achieves the best results (the most clicks) is declared the winner of the test.
4. You can manually pick a winner or the winner is automatically published when the test completes. Automatic publishing only happens when statistically significant.





Module A – Getting Started with Episerver CMS – Managing content – A/B testing

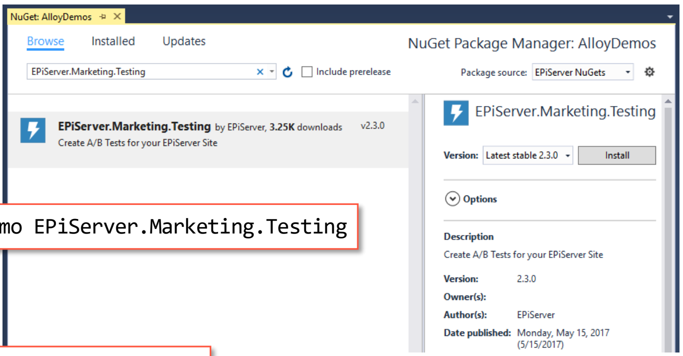
Works with DXC Service	Yes
Requires license	No

Installing the A/B testing add-on

Use NuGet Package Manager…

…or enter the following command in the Package Manager Console:

```
Install-Package -ProjectName AlloyDemo EPiServer.Marketing.Testing
```



NuGet: AlloyDemos
Browse
Installed
Updates
NuGet Package Manager: AlloyDemos
EPiServer.Marketing.Testing
Include prerelease
Package source: EPiServer NuGets
Install
Options

EPiServer.Marketing.Testing by EPiServer, 3.25K downloads v2.3.0
Create A/B Tests for your EPiServer Site

Version: Latest stable 2.3.0

Description
Create A/B Tests for your EPiServer Site

Version: 2.3.0
Owner(s):
Author(s): EPiServer
Date published: Monday, May 15, 2017 (5/15/2017)

You will need to update dependent packages and the database schema:

```
Update-Package -ProjectName AlloyDemo -ToHighestMinor
Update-EPiDatabase
```

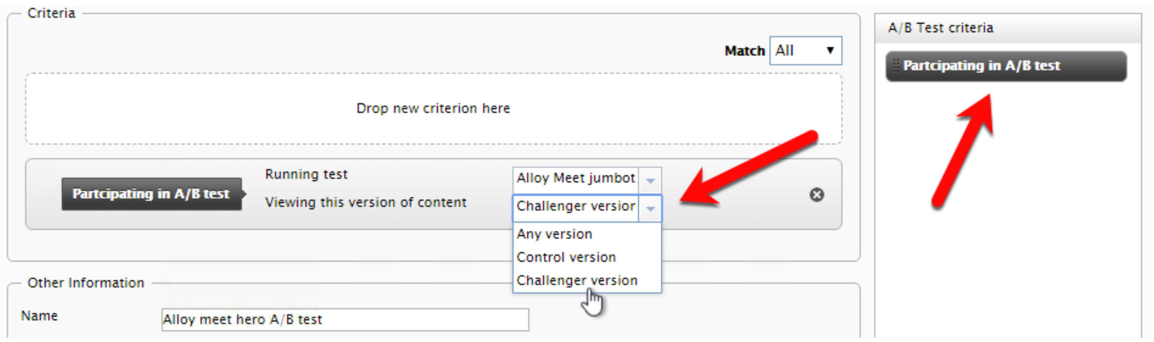
Episerver 136

Visitor group criterion for A/B testing available for Episerver 11

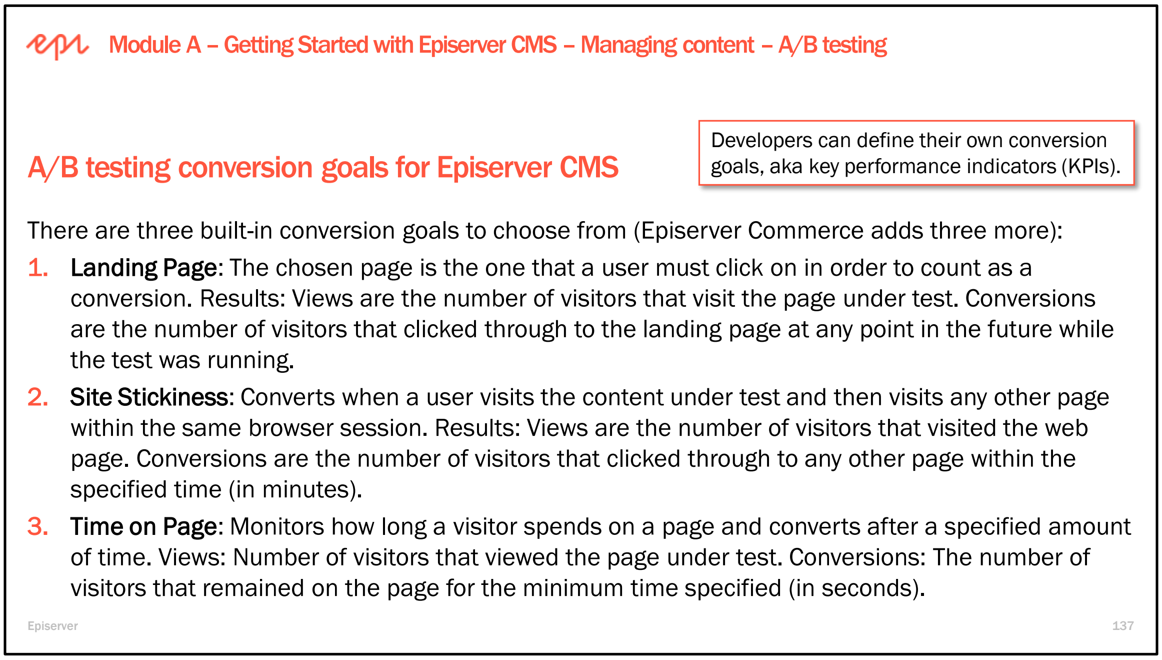
The Episerver visitor group criterion that allows editors to define visitor groups depending on whether an end user is participating in an A/B test is now available for Episerver 11.

```
Install-Package AbTestVisitorGroupCriteria
```

The package adds a new visitor group called “Participating in A/B test” as shown below:



https://www.david-tec.com/2018/01/visitor-group-criterion-for-ab-testing-available-for-episerver-11/



Module A – Getting Started with Episerver CMS – Managing content – A/B testing

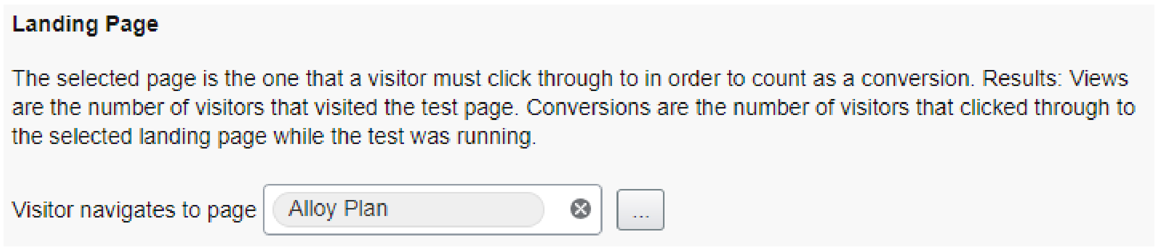
A/B testing conversion goals for Episerver CMS

Developers can define their own conversion goals, aka key performance indicators (KPIs).

There are three built-in conversion goals to choose from (Episerver Commerce adds three more):

1. **Landing Page**: The chosen page is the one that a user must click on in order to count as a conversion. Results: Views are the number of visitors that visit the page under test. Conversions are the number of visitors that clicked through to the landing page at any point in the future while the test was running.
2. **Site Stickiness**: Converts when a user visits the content under test and then visits any other page within the same browser session. Results: Views are the number of visitors that visited the web page. Conversions are the number of visitors that clicked through to any other page within the specified time (in minutes).
3. **Time on Page**: Monitors how long a visitor spends on a page and converts after a specified amount of time. Views: Number of visitors that viewed the page under test. Conversions: The number of visitors that remained on the page for the minimum time specified (in seconds).

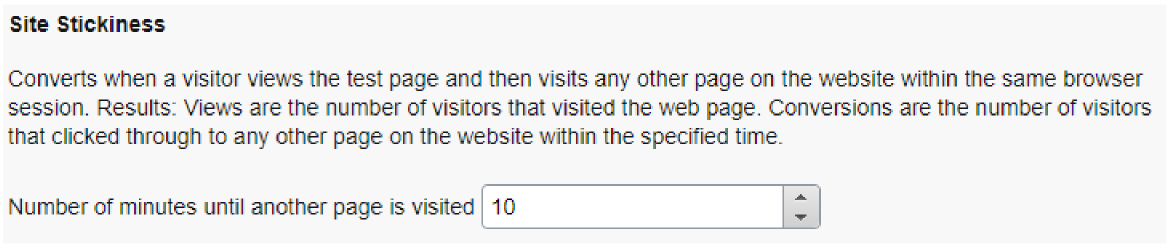
Episerver 137



Landing Page

The selected page is the one that a visitor must click through to in order to count as a conversion. Results: Views are the number of visitors that visited the test page. Conversions are the number of visitors that clicked through to the selected landing page while the test was running.

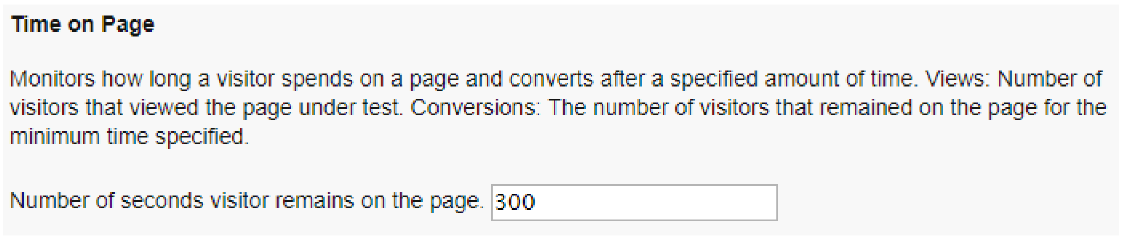
Visitor navigates to page: Alloy Plan



Site Stickiness

Converts when a visitor views the test page and then visits any other page on the website within the same browser session. Results: Views are the number of visitors that visited the web page. Conversions are the number of visitors that clicked through to any other page on the website within the specified time.

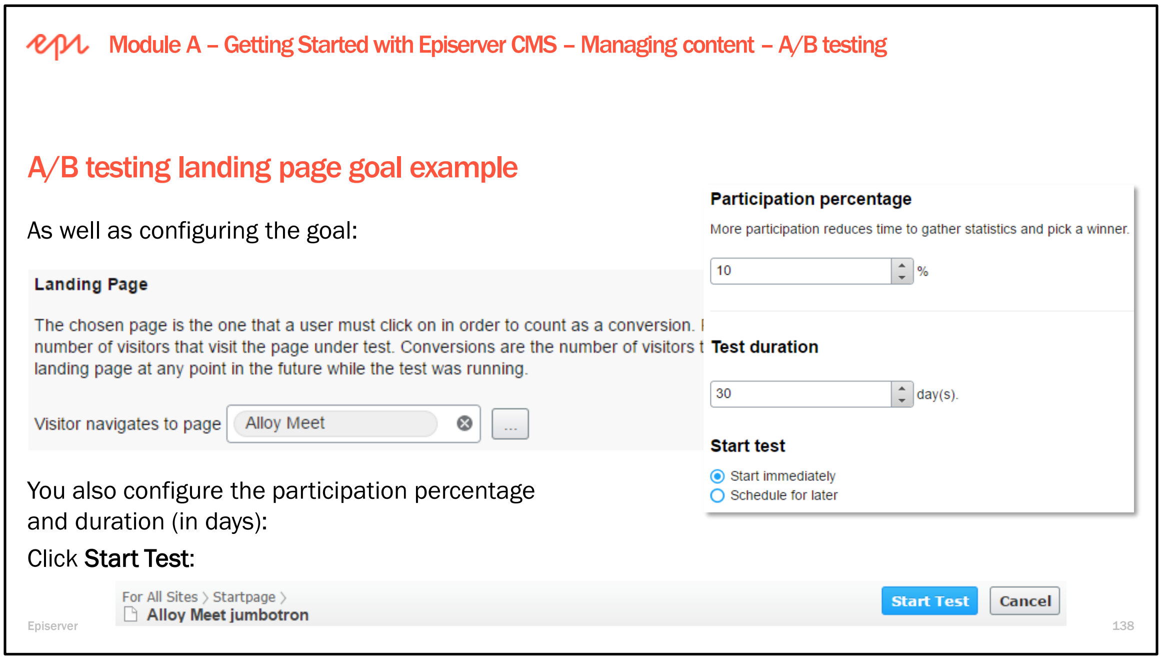
Number of minutes until another page is visited: 10



Time on Page

Monitors how long a visitor spends on a page and converts after a specified amount of time. Views: Number of visitors that viewed the page under test. Conversions: The number of visitors that remained on the page for the minimum time specified.

Number of seconds visitor remains on the page: 300



Module A – Getting Started with Episerver CMS – Managing content – A/B testing

A/B testing landing page goal example

As well as configuring the goal:

Landing Page

The chosen page is the one that a user must click on in order to count as a conversion. I number of visitors that visit the page under test. Conversions are the number of visitors t landing page at any point in the future while the test was running.

Visitor navigates to page: Alloy Meet

Participation percentage

More participation reduces time to gather statistics and pick a winner.

10 %

Test duration

30 day(s).

Start test

[✓ Start immediately]
Schedule for later

You also configure the participation percentage and duration (in days):

Click **Start Test**:

For All Sites > Startpage >
Alloy Meet jumbotron

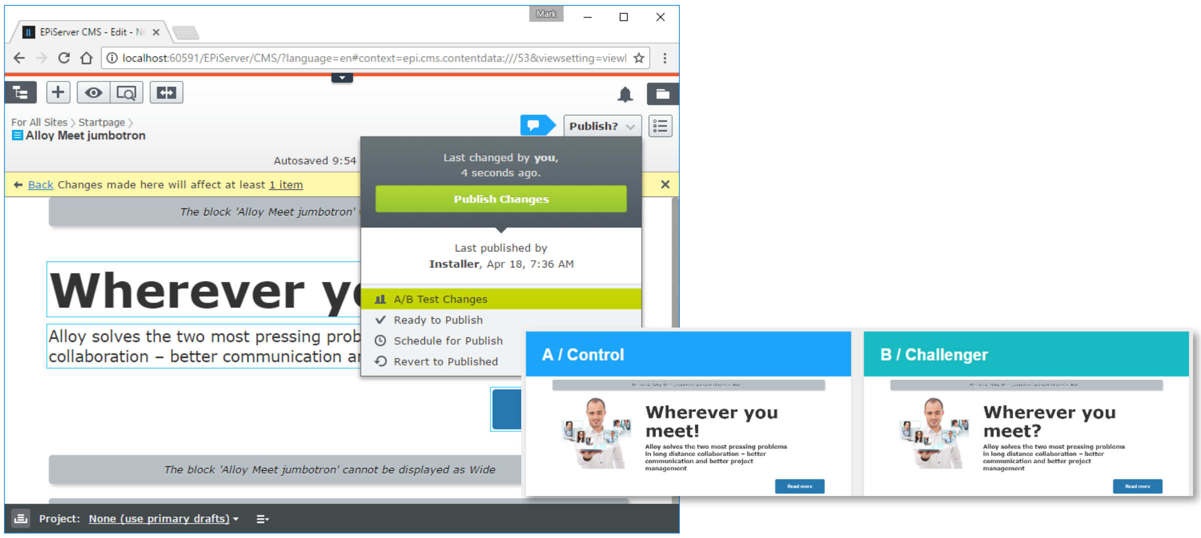
Start Test | Cancel

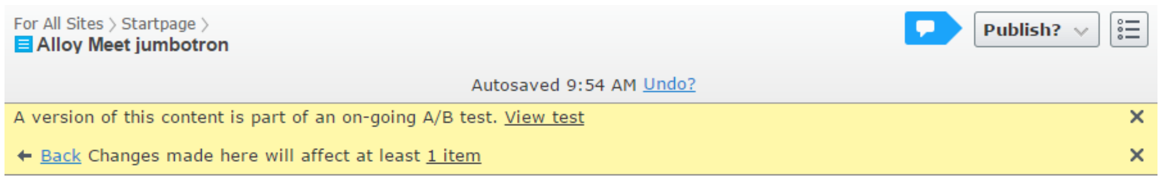
Episerver 138

A/B testing example

In Alloy, edit the Alloy Meet jumbotron block, and changed the "!" to a "?" in the Header property. If you try this demonstration, make sure you apply the A/B test to the block and not the page you drag and drop it onto.

Publish button has a new choice: **A/B Test Changes**. Do NOT publish the change!


EPiServer CMS - Edit
localhost:60591/EPiServer/CMS/?language=en#context=epi.cms.contentdata:///53&viewsetting=viewl
For All Sites > Startpage >
Alloy Meet jumbotron
Publish?
Autosaved 9:54
Back Changes made here will affect at least 1 item
The block 'Alloy Meet jumbotron'
Wherever y
Alloy solves the two most pressing prob
collaboration – better communication ar
Last changed by you, 4 seconds ago.
Publish Changes
Last published by Installer, Apr 18, 7:36 AM
A/B Test Changes
Ready to Publish
Schedule for Publish
Revert to Published
The block 'Alloy Meet jumbotron' cannot be displayed as Wide
Project: None (use primary drafts)
A / Control
Wherever you meet!
B / Challenger
Wherever you meet?



For All Sites > Startpage >
Alloy Meet jumbotron

Publish?

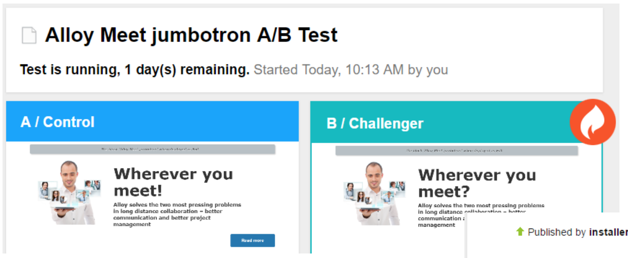
Autosaved 9:54 AM Undo?

A version of this content is part of an on-going A/B test. View test

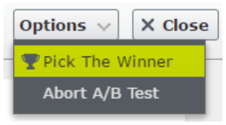
Back Changes made here will affect at least 1 item

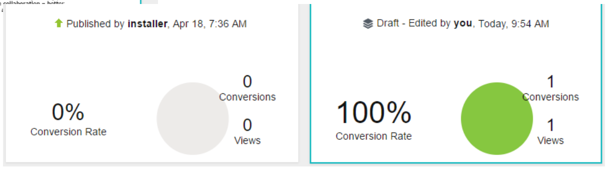
A/B testing viewing results example

The content with the most conversions is highlighted with a flame icon:


Alloy Meet jumbotron A/B Test

Test is running, 1 day(s) remaining. Started Today, 10:13 AM by you



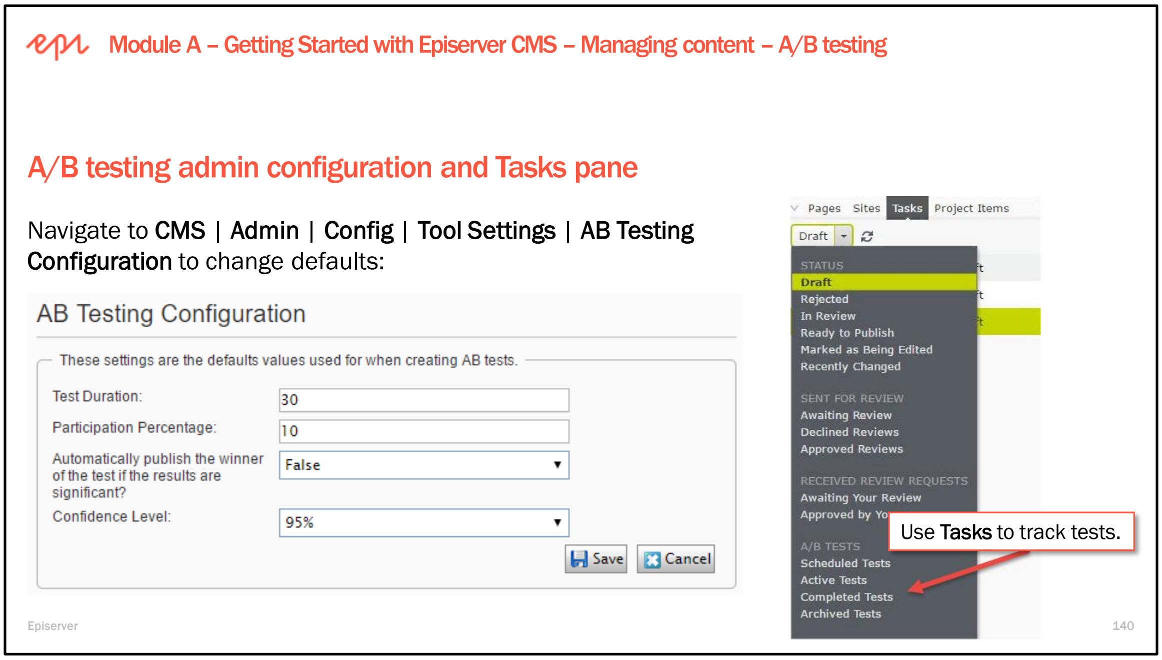

Published by **installer**, Apr 18, 7:36 AM

0% Conversion Rate — 0 Conversions — 0 Views

Draft - Edited by **you**, Today, 9:54 AM

100% Conversion Rate — 1 Conversions — 1 Views

You can: (1) let the test run to completion, (2) pick the winner, or (3) abort the test.



Module A – Getting Started with Episerver CMS – Managing content – A/B testing

A/B testing admin configuration and Tasks pane

Navigate to **CMS** | **Admin** | **Config** | **Tool Settings** | **AB Testing Configuration** to change defaults:

AB Testing Configuration

These settings are the defaults values used for when creating AB tests.

Test Duration:	30
Participation Percentage:	10
Automatically publish the winner of the test if the results are significant?	False
Confidence Level:	95%

Save Cancel

Use **Tasks** to track tests.

Episerver 140

A/B testing and Google SEO

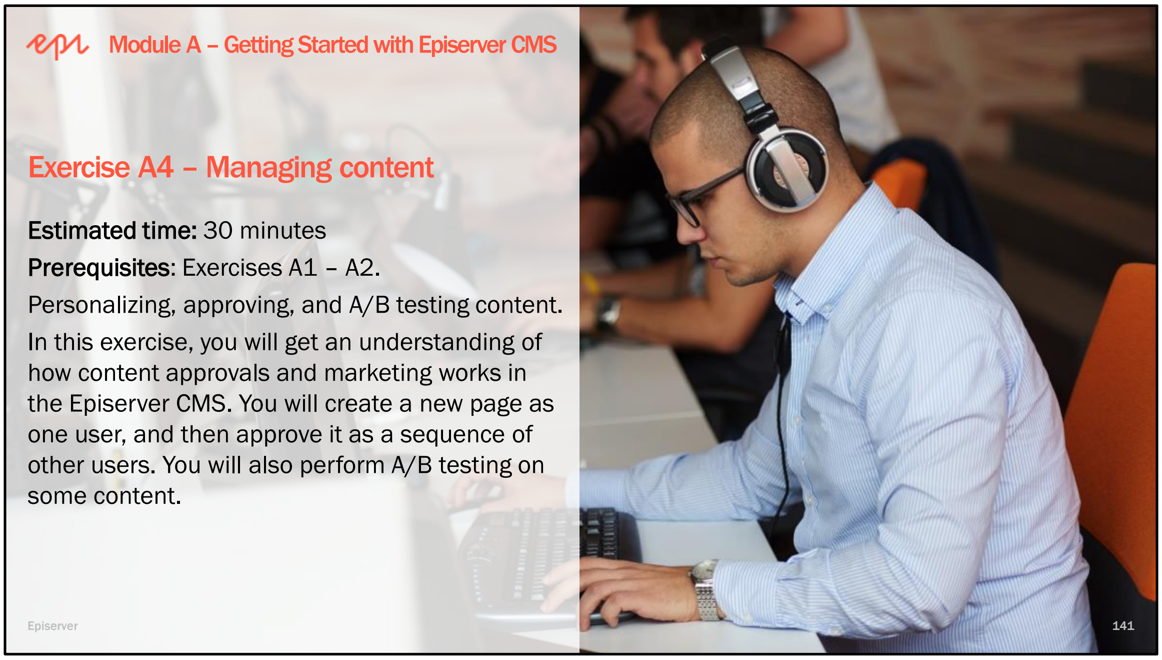
How does A/B testing affect what Googlebot sees on your site? Below are some guidelines for running an effective test with minimal impact on your site's search performance:

No cloaking. Showing one set of content to humans, and a different set to Googlebot is against Google guidelines. Make sure that you're not deciding whether to serve the test, or which content variant to serve, based on user-agent.

Use rel="canonical". If you're running an A/B test with multiple URLs, you can use the rel="canonical" link attribute on all of your alternate URLs to indicate that the original URL is the preferred version. We recommend using rel="canonical" rather than a noindex meta tag because it more closely matches your intent in this situation.

Use 302s, not 301s. If you're running an A/B test that redirects users from the original URL to a variation URL, use a 302 (temporary) redirect, not a 301 (permanent) redirect.

Website testing & Google search https://webmasters.googleblog.com/2012/08/website-testing-google-search.html



Module A – Getting Started with Episerver CMS

Exercise A4 – Managing content

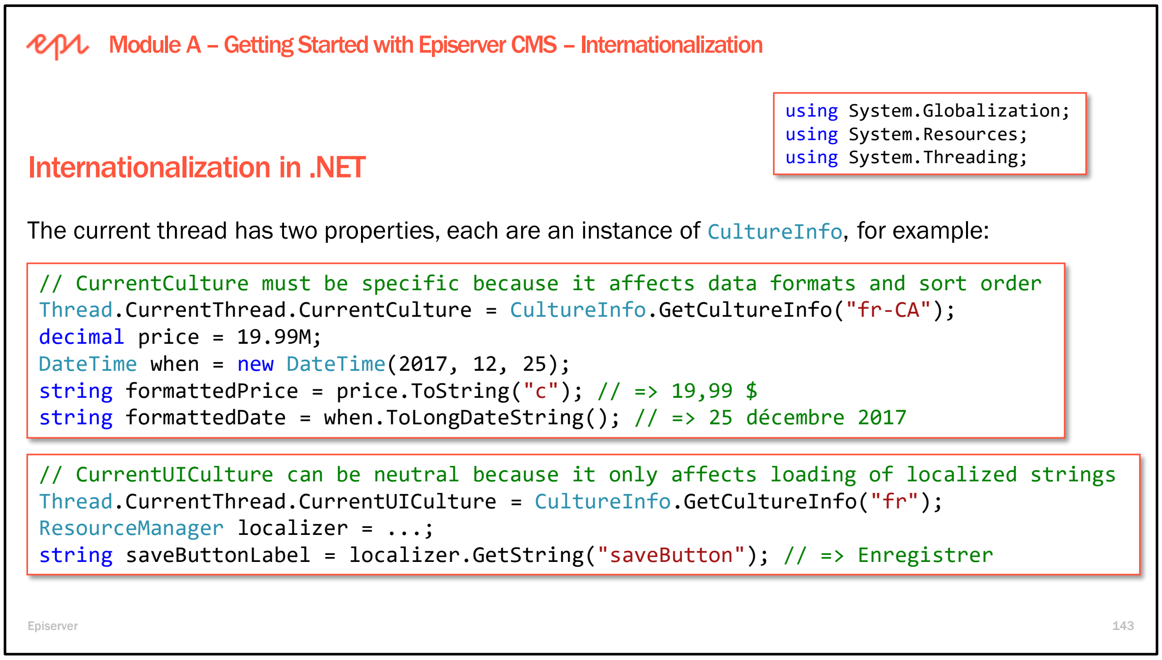
Estimated time: 30 minutes

Prerequisites: Exercises A1 – A2.

Personalizing, approving, and A/B testing content.

In this exercise, you will get an understanding of how content approvals and marketing works in the Episerver CMS. You will create a new page as one user, and then approve it as a sequence of other users. You will also perform A/B testing on some content.

Episerver 141



Module A – Getting Started with Episerver CMS – Internationalization

```
using System.Globalization;
using System.Resources;
using System.Threading;
```

Internationalization in .NET

The current thread has two properties, each are an instance of `CultureInfo`, for example:

```
// CurrentCulture must be specific because it affects data formats and sort order
Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("fr-CA");
decimal price = 19.99M;
DateTime when = new DateTime(2017, 12, 25);
string formattedPrice = price.ToString("c"); // => 19,99 $
string formattedDate = when.ToLongDateString(); // => 25 décembre 2017
```

```
// CurrentUICulture can be neutral because it only affects loading of localized strings
Thread.CurrentThread.CurrentUICulture = CultureInfo.GetCultureInfo("fr");
ResourceManager localizer = ...;
string saveButtonLabel = localizer.GetString("saveButton"); // => Enregistrer
```

Episerver 143

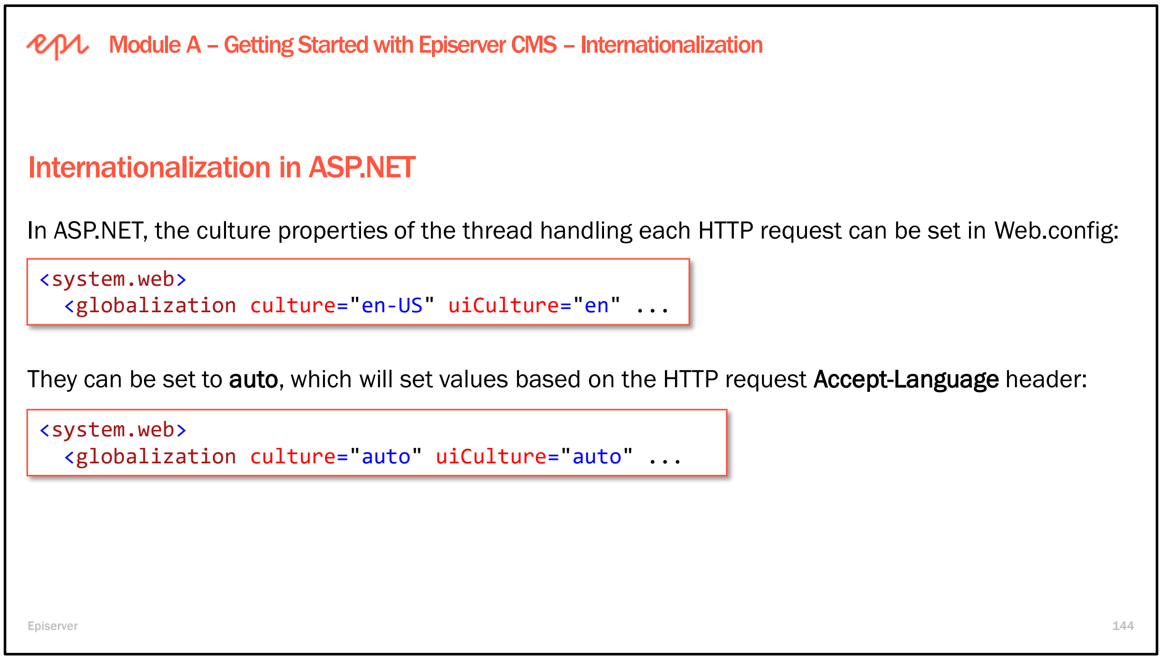
Internationalization terms

Internationalization (I18N): Describes the combination of globalization and localization.

Globalization (G11N): The process of making an app support different languages and regions.

Localization (L10N): The process of customizing an app for a given language and region.

Culture: A language and, optionally, a region. A locale is the same as a culture.
- Neutral culture: A culture that has a language, but not a region. (for example "en", "fr")
- Specific culture: A culture that has a language and region. (for example "en-US", "en-GB", "fr-CA")



Module A – Getting Started with Episerver CMS – Internationalization

Internationalization in ASP.NET

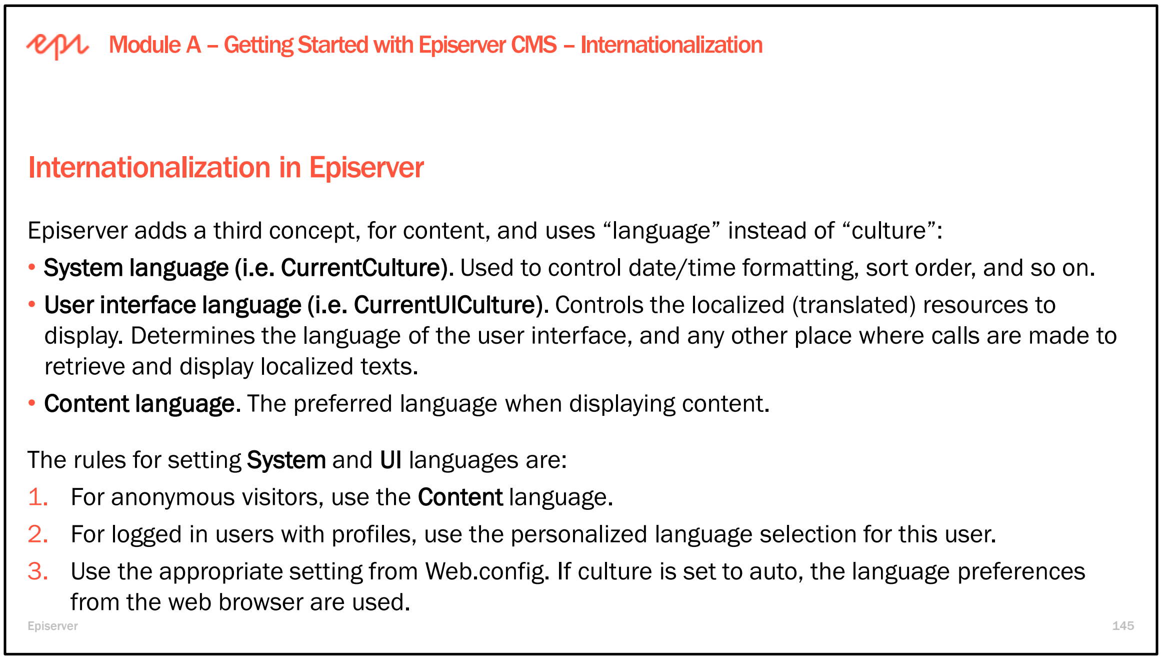
In ASP.NET, the culture properties of the thread handling each HTTP request can be set in Web.config:

```
<system.web>
  <globalization culture="en-US" uiCulture="en" ...
```

They can be set to **auto**, which will set values based on the HTTP request **Accept-Language** header:

```
<system.web>
  <globalization culture="auto" uiCulture="auto" ...
```

Episerver 144



Module A – Getting Started with Episerver CMS – Internationalization

Internationalization in Episerver

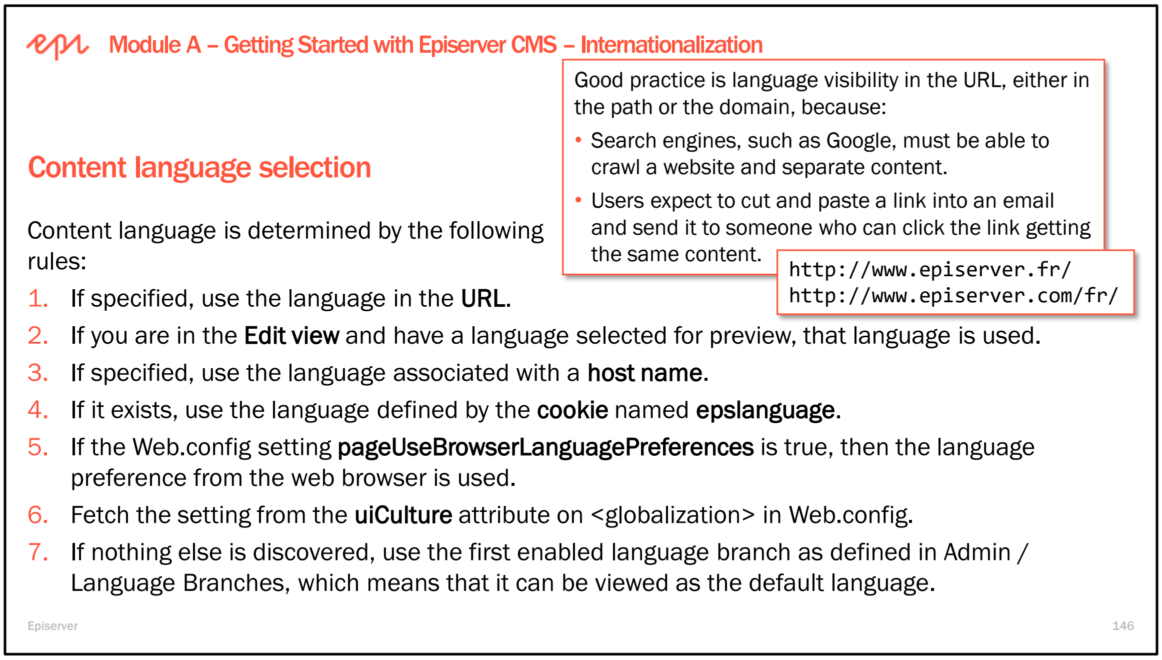
Episerver adds a third concept, for content, and uses “language” instead of “culture”:

- **System language (i.e. CurrentCulture).** Used to control date/time formatting, sort order, and so on.
- **User interface language (i.e. CurrentUICulture).** Controls the localized (translated) resources to display. Determines the language of the user interface, and any other place where calls are made to retrieve and display localized texts.
- **Content language.** The preferred language when displaying content.

The rules for setting **System** and **UI** languages are:

1. For anonymous visitors, use the **Content** language.
2. For logged in users with profiles, use the personalized language selection for this user.
3. Use the appropriate setting from Web.config. If culture is set to auto, the language preferences from the web browser are used.

Episerver 145



Module A – Getting Started with Episerver CMS – Internationalization

Content language selection

Good practice is language visibility in the URL, either in the path or the domain, because:

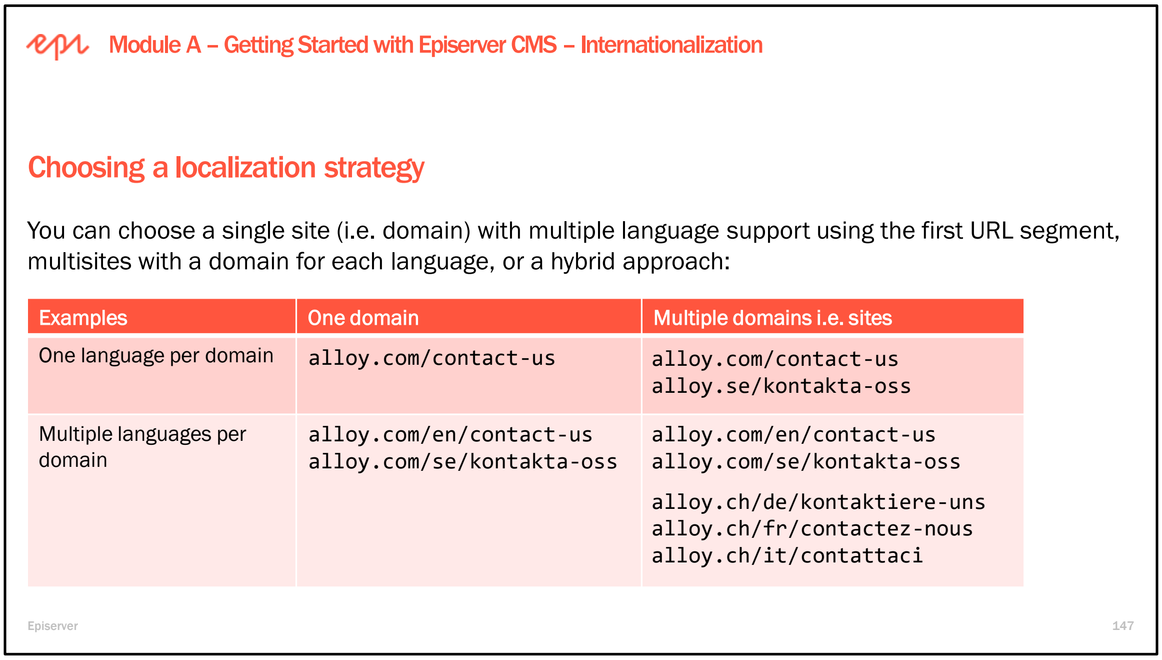
- Search engines, such as Google, must be able to crawl a website and separate content.
- Users expect to cut and paste a link into an email and send it to someone who can click the link getting the same content.

```
http://www.episerver.fr/
http://www.episerver.com/fr/
```

Content language is determined by the following rules:

1. If specified, use the language in the **URL**.
2. If you are in the **Edit view** and have a language selected for preview, that language is used.
3. If specified, use the language associated with a **host name**.
4. If it exists, use the language defined by the **cookie** named **epslanguage**.
5. If the Web.config setting **pageUseBrowserLanguagePreferences** is true, then the language preference from the web browser is used.
6. Fetch the setting from the **uiCulture** attribute on <globalization> in Web.config.
7. If nothing else is discovered, use the first enabled language branch as defined in Admin / Language Branches, which means that it can be viewed as the default language.

Episerver 146



Module A – Getting Started with Episerver CMS – Internationalization

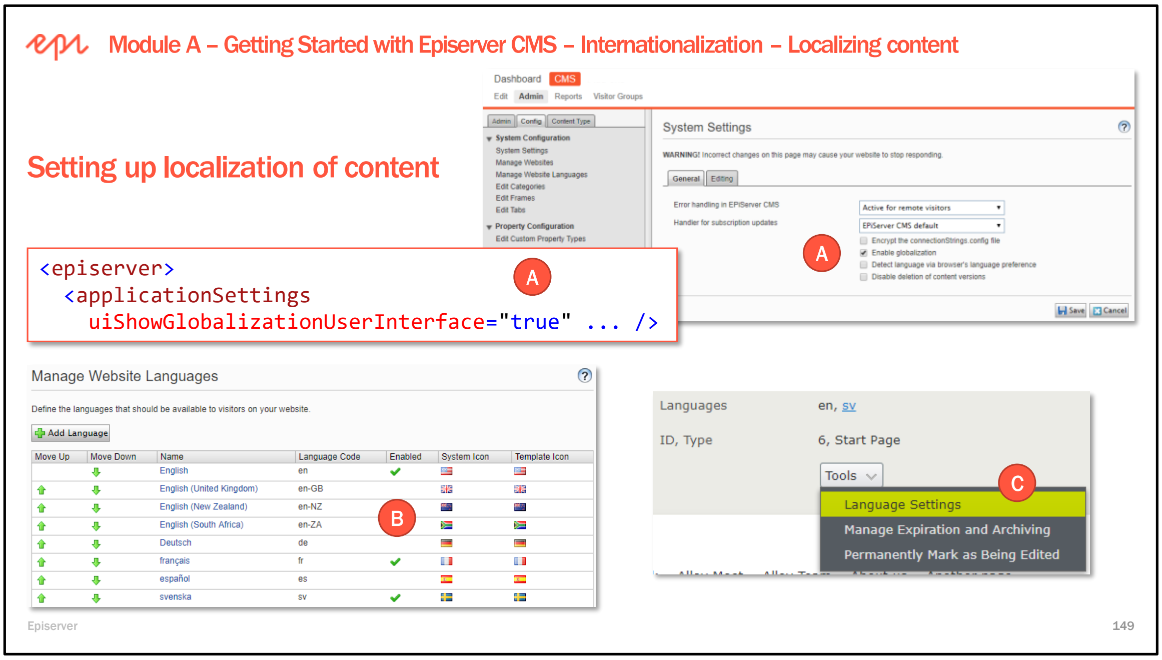
Choosing a localization strategy

You can choose a single site (i.e. domain) with multiple language support using the first URL segment, multisites with a domain for each language, or a hybrid approach:

Examples	One domain	Multiple domains i.e. sites
One language per domain	`alloy.com/contact-us`	`alloy.com/contact-us` `alloy.se/kontakta-oss`

Multiple languages per domain	`alloy.com/en/contact-us`
	`alloy.com/se/kontakta-oss`
		`alloy.com/en/contact-us`
		`alloy.com/se/kontakta-oss`
		
		`alloy.ch/de/kontaktiere-uns`
		`alloy.ch/fr/contactez-nous`
		`alloy.ch/it/contattaci`

Episerver 147



Module A – Getting Started with Episerver CMS – Internationalization – Localizing content

Setting up localization of content

Dashboard | CMS

Edit | Admin | Reports | Visitor Groups

Admin | Config | Content Type

- System Configuration
  - System Settings
  - Manage Websites
  - Manage Website Languages
  - Edit Categories
  - Edit Frames
  - Edit Tabs
- Property Configuration
  - Edit Custom Property Types

System Settings

WARNING! Incorrect changes on this page may cause your website to stop responding.

General | Editing

Error handling in EPiServer CMS	Active for remote visitors
Handler for subscription updates	EPiServer CMS default

- ☐ Encrypt the connectionStrings.config file
- ☑ Enable globalization
- ☐ Detect language via browser's language preference
- ☐ Disable deletion of content versions

Save | Cancel

A

```
<episerver>
  <applicationSettings
    uiShowGlobalizationUserInterface="true" ... />
```

A

Manage Website Languages

Define the languages that should be available to visitors on your website.

Add Language

Move Up	Move Down	Name	Language Code	Enabled	System Icon	Template Icon
	⇩	English	en	✓	🇺🇸	🇺🇸
⇧	⇩	English (United Kingdom)	en-GB		🇬🇧	🇬🇧
⇧	⇩	English (New Zealand)	en-NZ		🇳🇿	🇳🇿
⇧	⇩	English (South Africa)	en-ZA		🇿🇦	🇿🇦
⇧	⇩	Deutsch	de		🇩🇪	🇩🇪
⇧	⇩	français	fr	✓	🇫🇷	🇫🇷
⇧	⇩	español	es		🇪🇸	🇪🇸
⇧	⇩	svenska	sv	✓	🇸🇪	🇸🇪

B

Languages	en, sv
ID, Type	6, Start Page

Tools

- Language Settings
- Manage Expiration and Archiving
- Permanently Mark as Being Edited

C

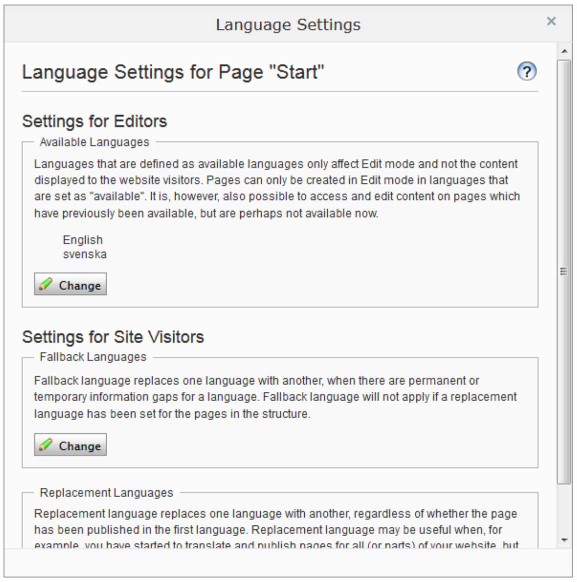
Episerver

149

Localization on the website is enabled in Web.config. The setting is also visible under System Settings in Admin (Config tab, System Configuration section). Changing the system settings (including globalization) from Admin is not recommended since it will (try to) update the Web.config files, which will cause the AppDomain to unload and reload, which is generally bad, especially in a production environment.

Add the languages to be available for the website with "Manage Website Languages" in Admin.
Choose Add Language to add a new language
Select a language and then select the Enabled checkbox to enable a language on the website
Optionally: Set access levels if needed

Make the languages available for web editors with the Language Settings in Edit view.
Fallback and replacement languages can be used to display other languages for visitors on the website.



Language Settings

Language Settings for Page "Start"

Settings for Editors

Available Languages

Languages that are defined as available languages only affect Edit mode and not the content displayed to the website visitors. Pages can only be created in Edit mode in languages that are set as "available". It is, however, also possible to access and edit content on pages which have previously been available, but are perhaps not available now.

English
svenska

Change

Settings for Site Visitors

Fallback Languages

Fallback language replaces one language with another, when there are permanent or temporary information gaps for a language. Fallback language will not apply if a replacement language has been set for the pages in the structure.

Change

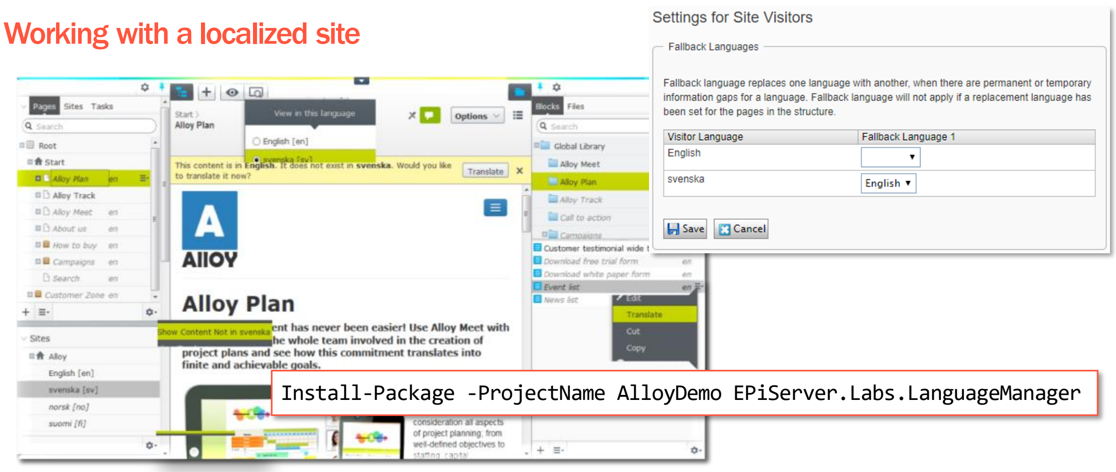
Replacement Languages

Replacement language replaces one language with another, regardless of whether the page has been published in the first language. Replacement language may be useful when, for example, you have started to translate and publish pages for all (or parts) of your website, but

Module A – Getting Started with Episerver CMS – Internationalization – Localizing content

Works with DXC Service	Yes
Requires license	No

Working with a localized site



Settings for Site Visitors

Fallback Languages

Fallback language replaces one language with another, when there are permanent or temporary information gaps for a language. Fallback language will not apply if a replacement language has been set for the pages in the structure.

Visitor Language	Fallback Language 1
English	
svenska	English

This content is in English. It does not exist in svenska. Would you like to translate it now?

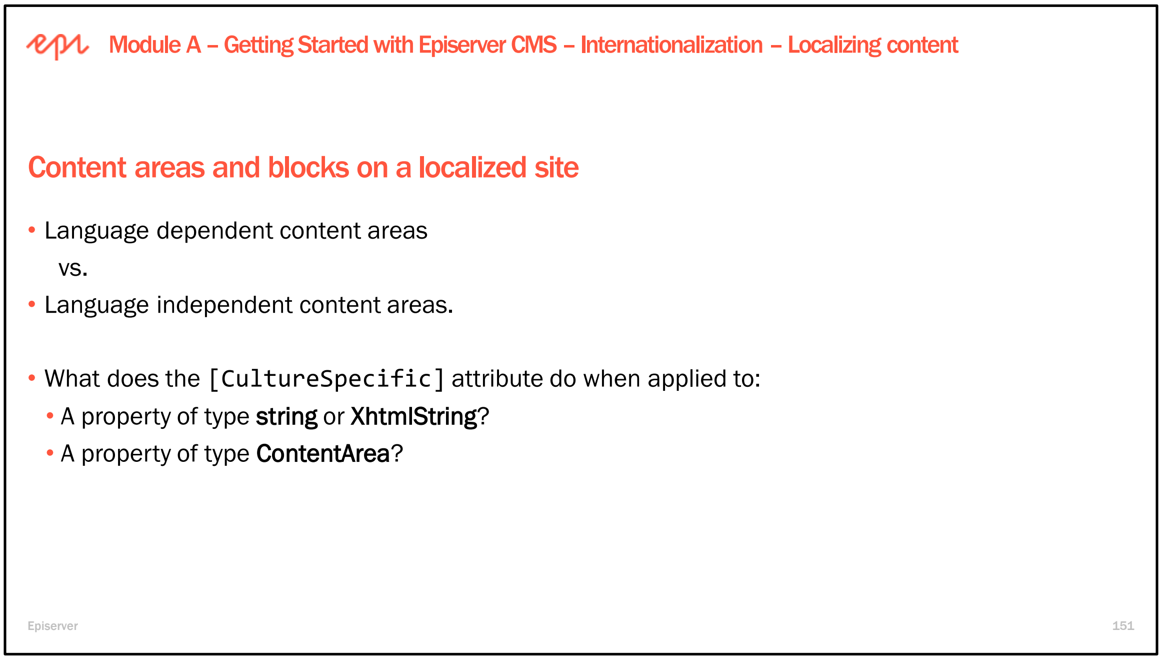
```
Install-Package -ProjectName AlloyDemo EPiServer.Labs.LanguageManager
```

Episerver 150

Functions for working with languages in Edit View

- All languages that are available for a site (set in “Manage Website Languages”) are by default listed in the Sites tab/gadget in Edit View.
- Select “Show All Languages” in the Sites settings menu to see languages that are enabled for the site but not yet available.
- You can toggle between the available languages using the View Settings button.
- All content, including blocks, can be in different languages.

Pages and Blocks are translated in the same way, with the option “Show content not in [currently selected language]” available in the settings menu for the page tree and for the shared blocks gadget. This option shows pages and blocks for all languages. Items that are not translated will have a language code representing the fallback language visible in the list next to the name. When “Show Content Only in...” is turned on, only content that is available for the current language (i.e. the language currently selected in “Sites”) will be shown.
For detailed information on localizing content please see the Globalization section in the SDKs and user guides.



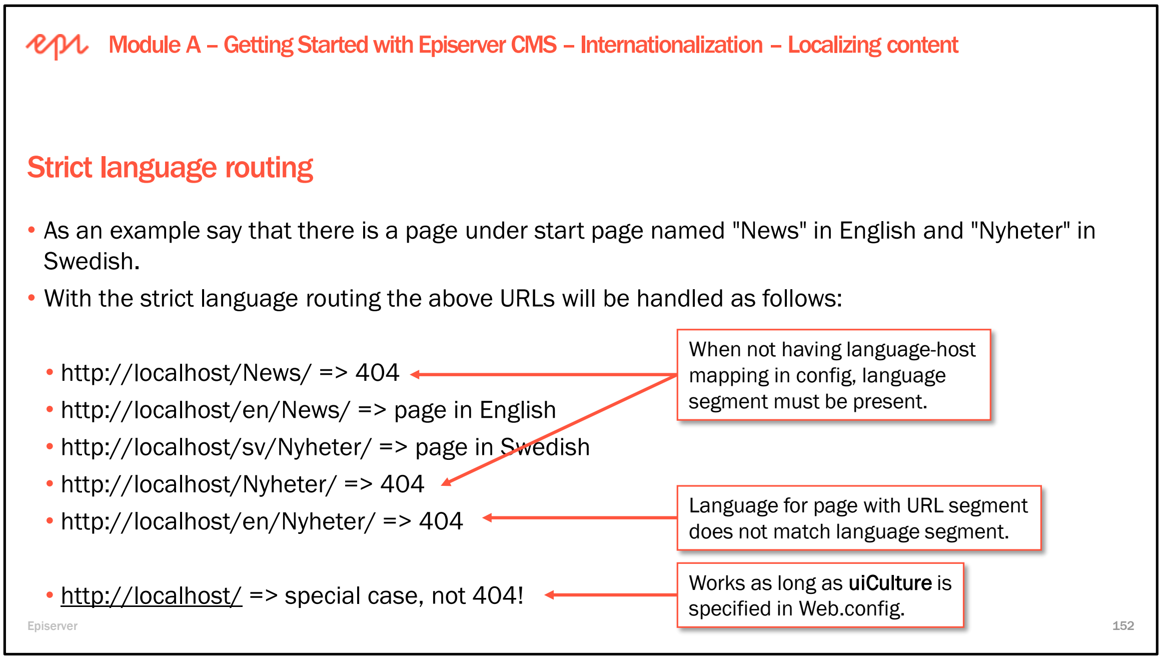
Module A – Getting Started with Episerver CMS – Internationalization – Localizing content

Content areas and blocks on a localized site

- Language dependent content areas
  vs.
- Language independent content areas.

- What does the `[CultureSpecific]` attribute do when applied to:
  - A property of type **string** or **XhtmlString**?
  - A property of type **ContentArea**?

Episerver 151



Module A – Getting Started with Episerver CMS – Internationalization – Localizing content

Strict language routing

- As an example say that there is a page under start page named "News" in English and "Nyheter" in Swedish.
- With the strict language routing the above URLs will be handled as follows:
  - http://localhost/News/ => 404
  - http://localhost/en/News/ => page in English
  - http://localhost/sv/Nyheter/ => page in Swedish
  - http://localhost/Nyheter/ => 404
  - http://localhost/en/Nyheter/ => 404
  - http://localhost/ => special case, not 404!

When not having language-host mapping in config, language segment must be present.

Language for page with URL segment does not match language segment.

Works as long as **uiCulture** is specified in Web.config.

Episerver 152

We now have the possibility to use strict URL handling (for better SEO without duplicated content).

To switch back to use less strict routing

- There is a configuration setting strictLanguageRouting on configuration element applicationSettings that can be set to false to get the more tolerant behavior that was default in previous versions of the CMS.

Full article on routing changes

See http://world.episerver.com/Blogs/Johan-Bjornfot/Dates1/2013/12/Routing-changes-in-75/ for full article on the above example and comparison to earlier versions of CMS.

Why do you not get a 404 when browsing the site root in a multi-language setup?

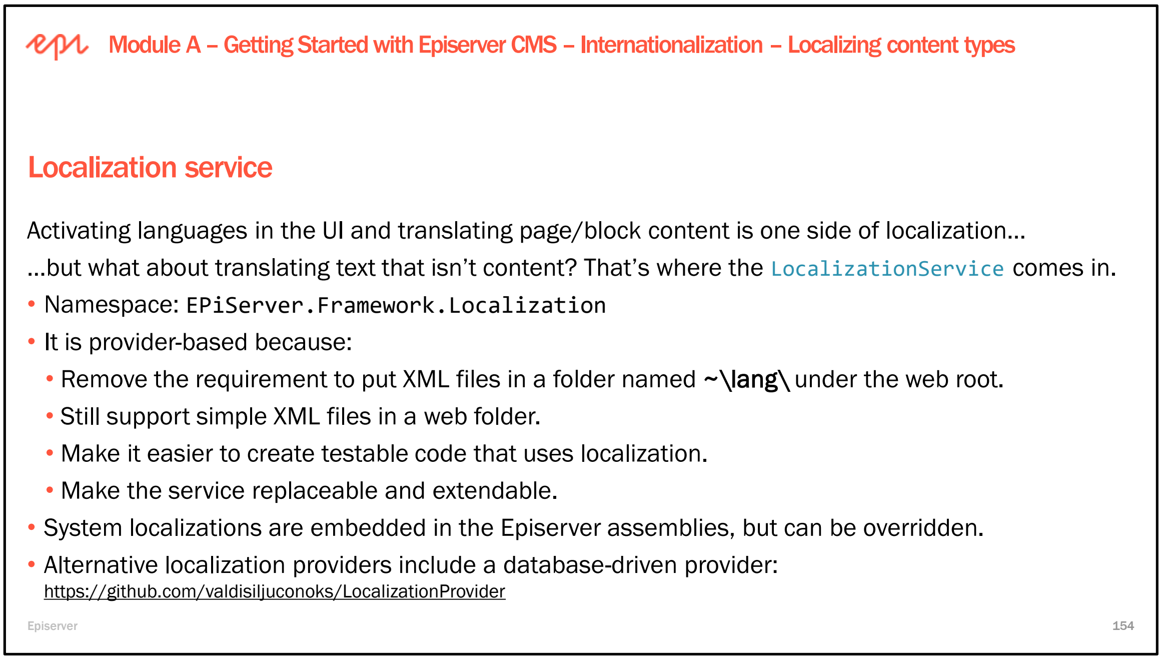
There is an exception for a request to the site root (i.e.path "/"), the reason being that it would be a very unwanted behavior if the home page gave a 404 by default.

The language for the request will be decided from several different parameters:

- First: is there a language mapping for the site?
- Second: Is the attribute pageUseBrowserLanguagePreferences enabled in applicationSettings? If so a check is made to see if the user has selected any language in the browser.
- Third: Is there a language mapping on *?
- If none of the above is set: Fall back to use uiCulture on the globalization element in web.config

This is documented in more detail the developer guide:

- http://world.episerver.com/documentation/developer-guides/CMS/globalization/



Module A – Getting Started with Episerver CMS – Internationalization – Localizing content types

Localization service

Activating languages in the UI and translating page/block content is one side of localization…
…but what about translating text that isn't content? That's where the `LocalizationService` comes in.

- Namespace: `EPiServer.Framework.Localization`
- It is provider-based because:
  - Remove the requirement to put XML files in a folder named **~\lang** under the web root.
  - Still support simple XML files in a web folder.
  - Make it easier to create testable code that uses localization.
  - Make the service replaceable and extendable.
- System localizations are embedded in the Episerver assemblies, but can be overridden.
- Alternative localization providers include a database-driven provider:
  https://github.com/valdisiljuconoks/LocalizationProvider

Episerver 154

LocalizationService is used behind the scenes to localize the Episerver user interface, and your custom code. Use the GetStringByCulture() method if you want the resource string for a specific language, and send in the language in the form of a CultureInfo.
Configuration example from the Alloy site - Configuring a custom localization provider in Web.config:

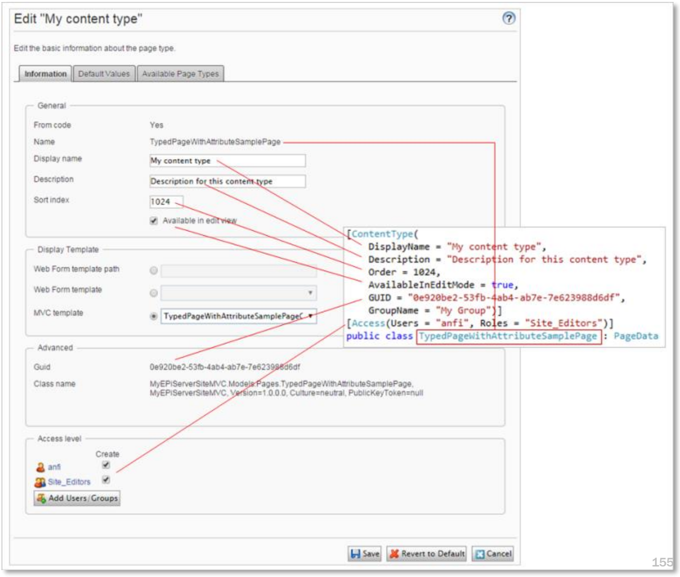
```
 <episerver.framework>
    ...
    <localization
        fallbackBehavior="Echo, MissingMessage, FallbackCulture" fallbackCulture="en">
      <providers>
        <add virtualPath="~/Resources/LanguageFiles"
             name="languageFiles”
             type="EPiServer.Framework.Localization.XmlResources.FileXmlLocalizationProvider,
                   EPiServer.Framework" />
      </providers>
    </localization>
    ...
  </episerver.framework>
```

The default fallback behavior is to echo the key without a missing message.
Performance considerations: Please note that a large number of providers will impact the time needed to find strings. Best performance is achieved with the least amount of providers.

Module A – Getting Started with Episerver CMS – Internationalization – Localizing content types

Content types in code/admin

1. Initial content type display name and description should be set in code using [ContentType] attribute. Values must be string literal or constant expressions.
2. Admins can override using UI, which updates the metadata in the database. Admins can revert back to the initial (code) values by clicking **Revert to Default** button.
3. Localization will override both.



Edit "My content type"

Edit the basic information about the page type.

General

From code	Yes
Name	TypedPageWithAttributeSamplePage
Display name	My content type
Description	Description for this content type
Sort index	1024
	✓ Available in edit view

Display Template

Web Form template path	○
Web Form template	○
MVC template	● TypedPageWithAttributeSamplePage(

Advanced

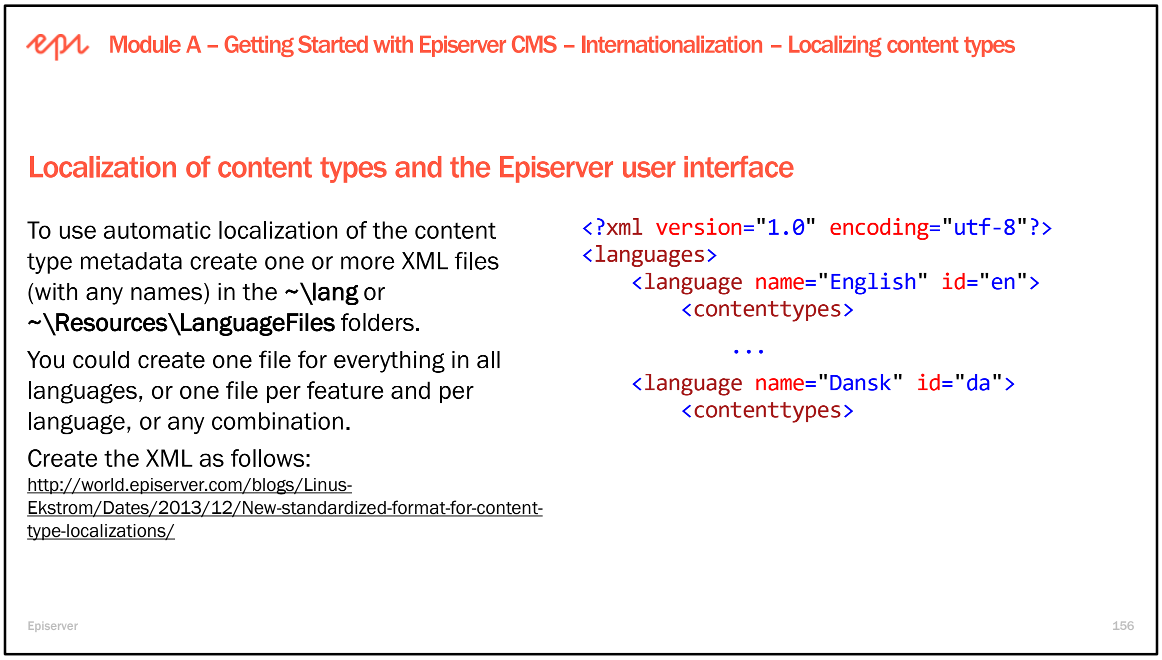
Guid	0e920be2-53fb-4ab4-ab7e-7e623988d6df
Class name	MyEPiServerSiteMVC.Models.Pages.TypedPageWithAttributeSamplePage, MyEPiServerSiteMVC, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

Access level

	Create
anfi	✓
Site_Editors	✓

```
[ContentType(
    DisplayName = "My content type",
    Description = "Description for this content type",
    Order = 1024,
    AvailableInEditMode = true,
    GUID = "0e920be2-53fb-4ab4-ab7e-7e623988d6df",
    GroupName = "My Group")]
[Access(Users = "anfi", Roles = "Site_Editors")]
public class TypedPageWithAttributeSamplePage : PageData
```

Episerver



Module A – Getting Started with Episerver CMS – Internationalization – Localizing content types

Localization of content types and the Episerver user interface

To use automatic localization of the content type metadata create one or more XML files (with any names) in the **~\lang** or **~\Resources\LanguageFiles** folders.

You could create one file for everything in all languages, or one file per feature and per language, or any combination.

Create the XML as follows:
http://world.episerver.com/blogs/Linus-Ekstrom/Dates/2013/12/New-standardized-format-for-content-type-localizations/

```
<?xml version="1.0" encoding="utf-8"?>
<languages>
    <language name="English" id="en">
        <contenttypes>
            ...
    <language name="Dansk" id="da">
        <contenttypes>
```

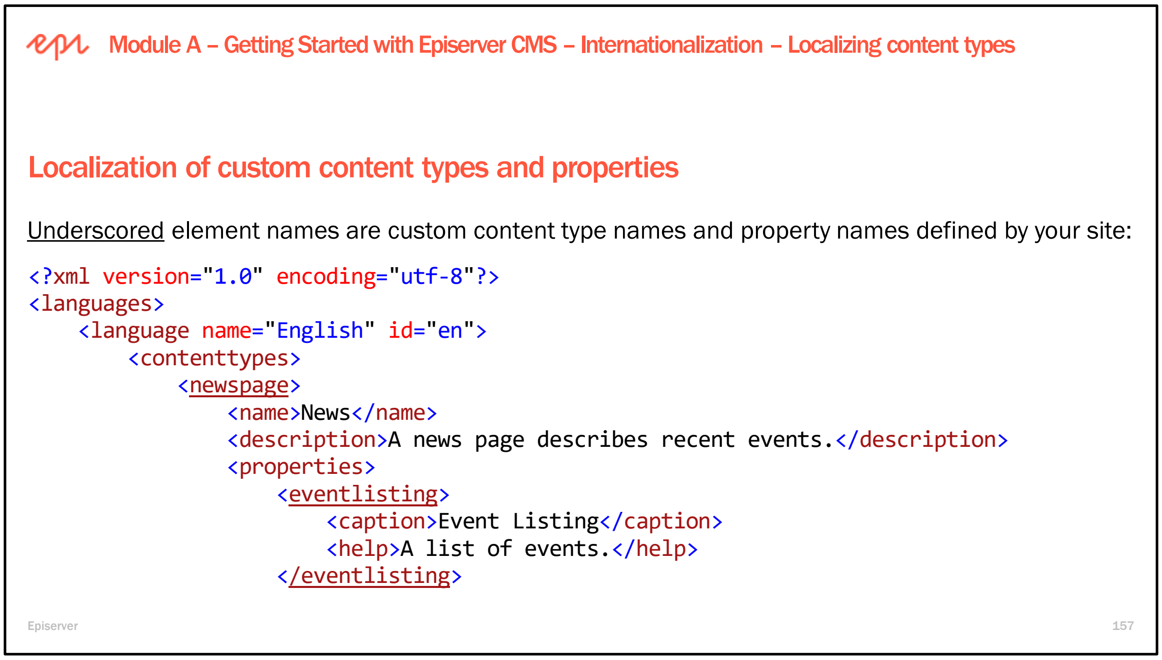
Episerver 156

Override Episerver's default UI texts

Sometimes you may want to translate the EPiServer user interface to a currently unsupported language or just want to change the text of some button or whatever.

https://getadigital.com/no/blogg/translating-episerver-ui/

https://ericceric.com/override-episervers-default-ui-texts/



Module A – Getting Started with Episerver CMS – Internationalization – Localizing content types

Localization of custom content types and properties

Underscored element names are custom content type names and property names defined by your site:

```
<?xml version="1.0" encoding="utf-8"?>
<languages>
    <language name="English" id="en">
        <contenttypes>
            <newspage>
                <name>News</name>
                <description>A news page describes recent events.</description>
                <properties>
                    <eventlisting>
                        <caption>Event Listing</caption>
                        <help>A list of events.</help>
                    </eventlisting>
```

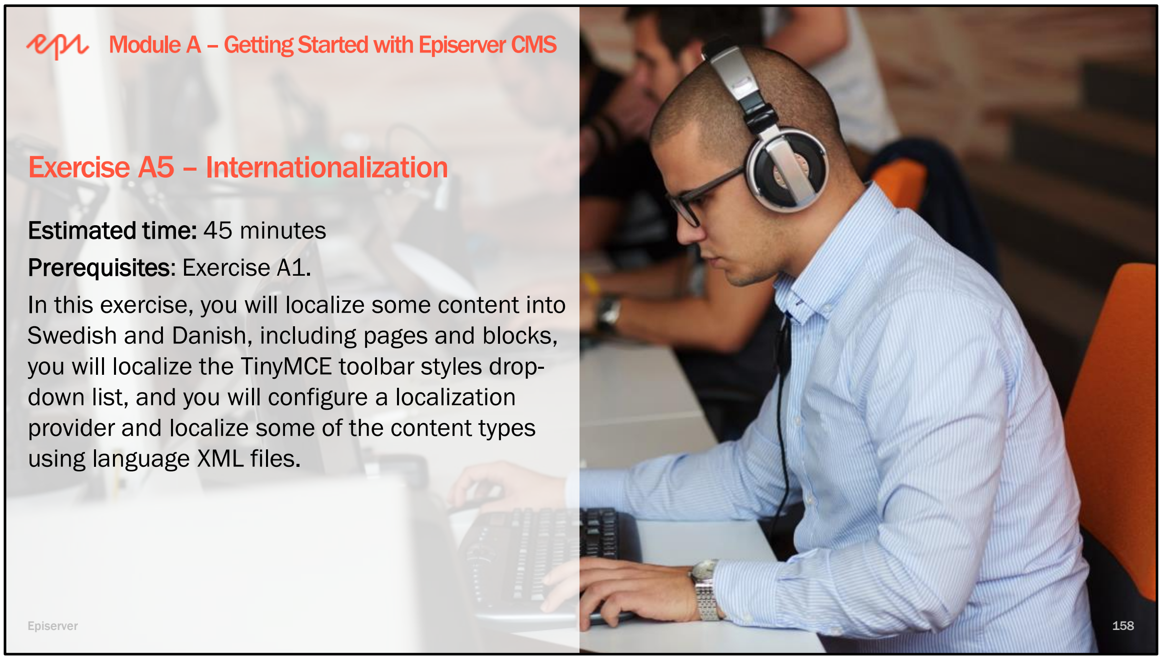
Episerver 157

You can set default values for interfaces and base classes that will then be used by all custom types that implement or inherit from them:

```
<?xml version="1.0" encoding="utf-8"?>
<languages>
    <language name="English" id="en">
        <contenttypes>
            <icontentdata>
                <properties>
                    <disableindexing>
                        <caption>Disable indexing</caption>
                        <help>Prevents the content from being indexed by Google.</help>
```

You can localize group/tab names like this:

```
<?xml version="1.0" encoding="utf-8"?>
<languages>
    <language name="English" id="en">
        <groups>
            <sitesettings>Site Settings</sitesettings>
            <eventinfo>Event Info</eventinfo>
```



Module A – Getting Started with Episerver CMS

Exercise A5 – Internationalization

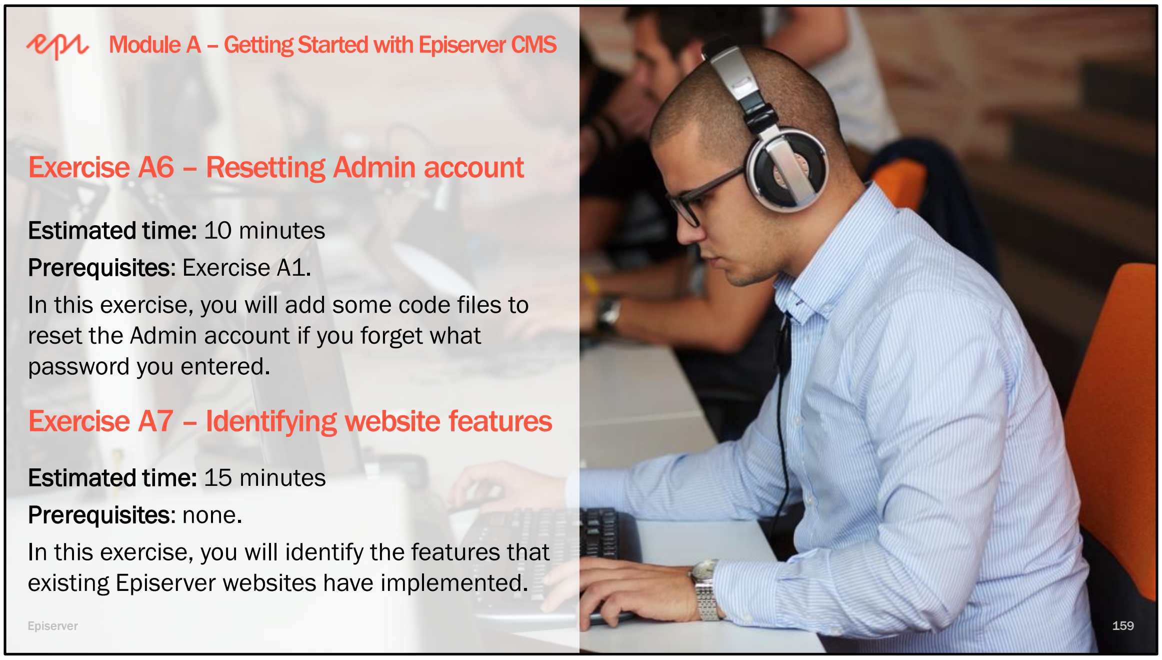
Estimated time: 45 minutes

Prerequisites: Exercise A1.

In this exercise, you will localize some content into Swedish and Danish, including pages and blocks, you will localize the TinyMCE toolbar styles drop-down list, and you will configure a localization provider and localize some of the content types using language XML files.

Episerver

158



Module A – Getting Started with Episerver CMS

Exercise A6 – Resetting Admin account

Estimated time: 10 minutes

Prerequisites: Exercise A1.

In this exercise, you will add some code files to reset the Admin account if you forget what password you entered.

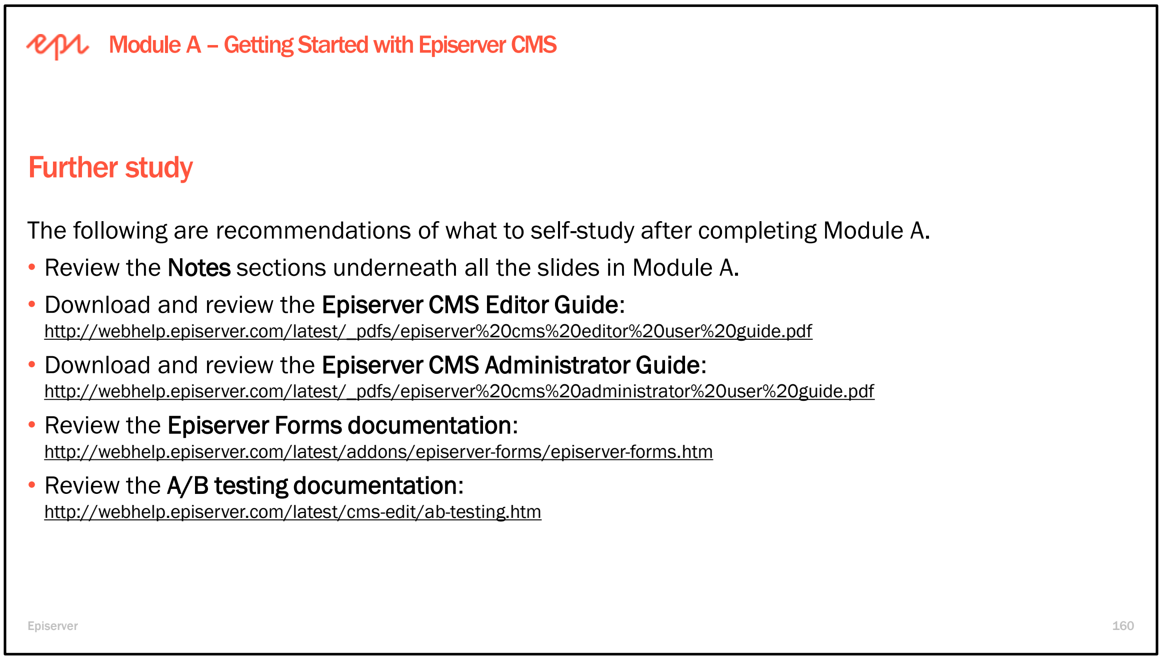
Exercise A7 – Identifying website features

Estimated time: 15 minutes

Prerequisites: none.

In this exercise, you will identify the features that existing Episerver websites have implemented.

Episerver 159



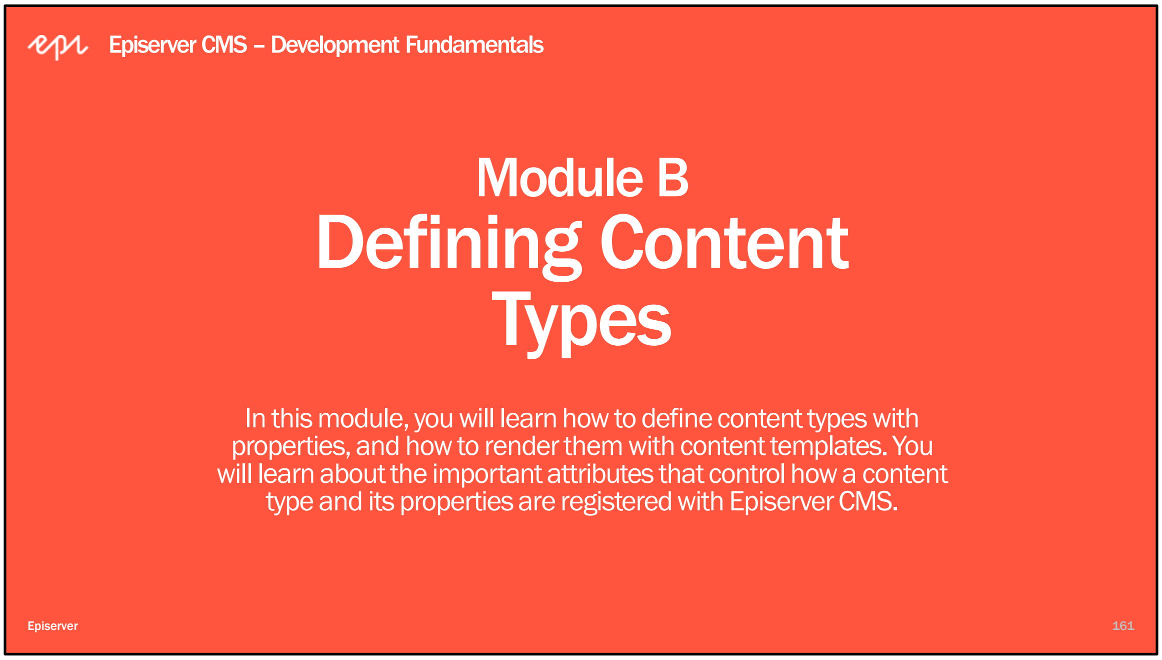
Module A – Getting Started with Episerver CMS

Further study

The following are recommendations of what to self-study after completing Module A.

- Review the **Notes** sections underneath all the slides in Module A.
- Download and review the **Episerver CMS Editor Guide**: http://webhelp.episerver.com/latest/_pdfs/episerver%20cms%20editor%20user%20guide.pdf
- Download and review the **Episerver CMS Administrator Guide**: http://webhelp.episerver.com/latest/_pdfs/episerver%20cms%20administrator%20user%20guide.pdf
- Review the **Episerver Forms documentation**: http://webhelp.episerver.com/latest/addons/episerver-forms/episerver-forms.htm
- Review the **A/B testing documentation**: http://webhelp.episerver.com/latest/cms-edit/ab-testing.htm

Episerver 160

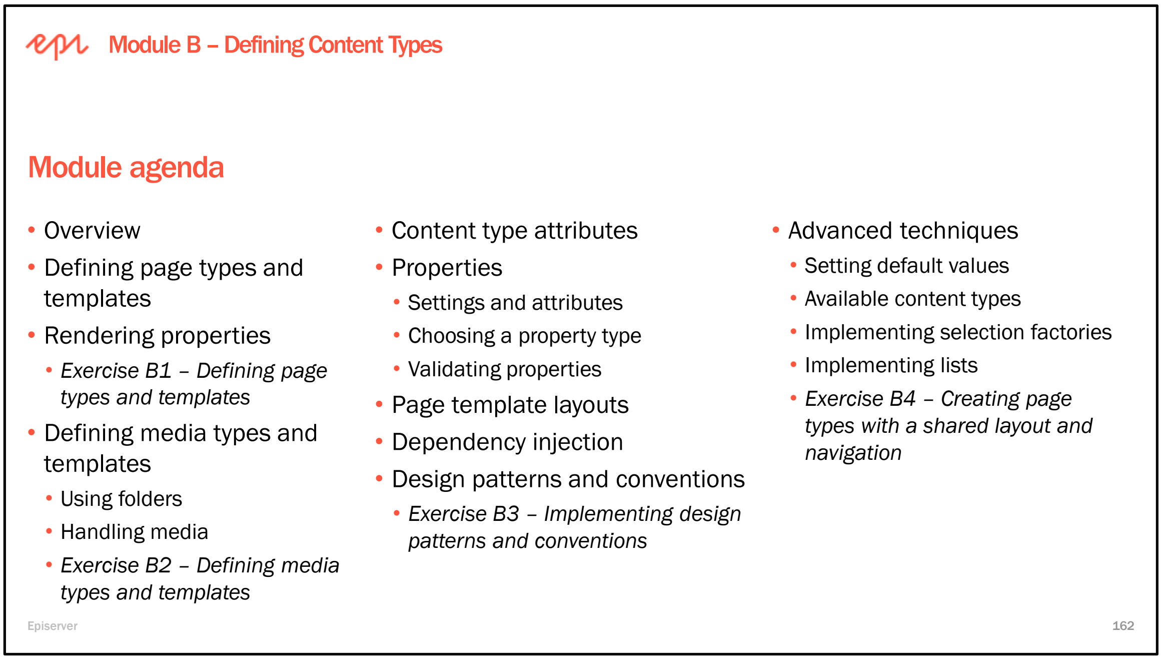


Episerver CMS – Development Fundamentals

Module B
Defining Content Types

In this module, you will learn how to define content types with properties, and how to render them with content templates. You will learn about the important attributes that control how a content type and its properties are registered with Episerver CMS.

Episerver 161

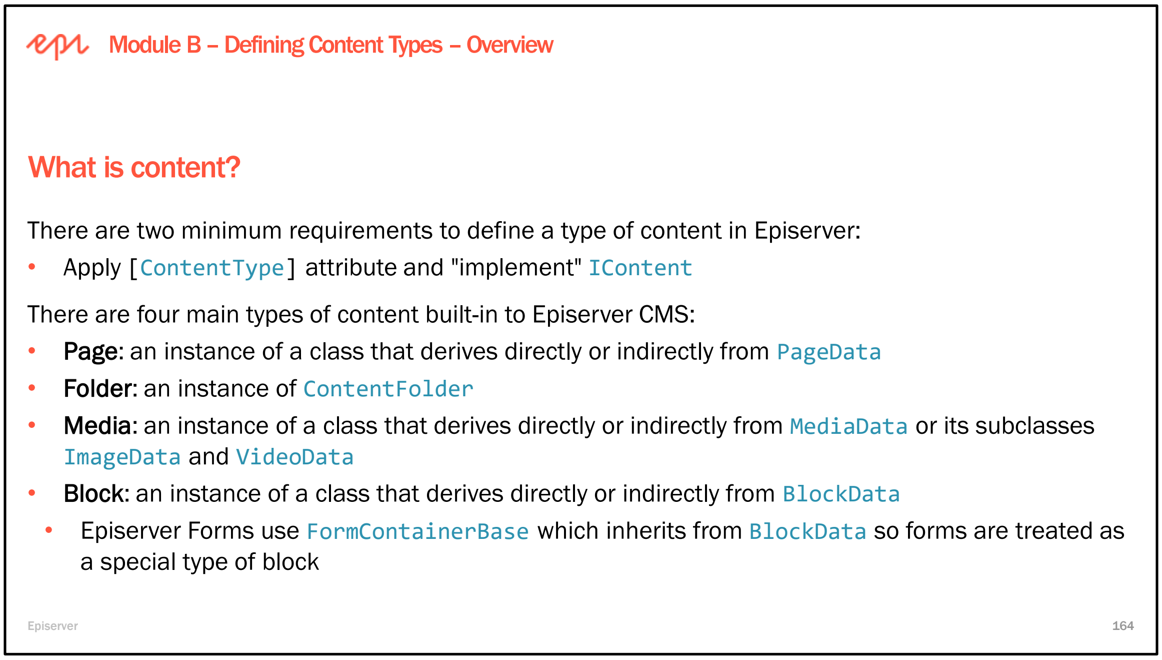


Module B – Defining Content Types

Module agenda

- Overview
- Defining page types and templates
- Rendering properties
  - *Exercise B1 – Defining page types and templates*
- Defining media types and templates
  - Using folders
  - Handling media
  - *Exercise B2 – Defining media types and templates*
- Content type attributes
- Properties
  - Settings and attributes
  - Choosing a property type
  - Validating properties
- Page template layouts
- Dependency injection
- Design patterns and conventions
  - *Exercise B3 – Implementing design patterns and conventions*
- Advanced techniques
  - Setting default values
  - Available content types
  - Implementing selection factories
  - Implementing lists
  - *Exercise B4 – Creating page types with a shared layout and navigation*

Episerver 162



Module B – Defining Content Types – Overview

What is content?

There are two minimum requirements to define a type of content in Episerver:

- Apply [`ContentType`] attribute and "implement" `IContent`

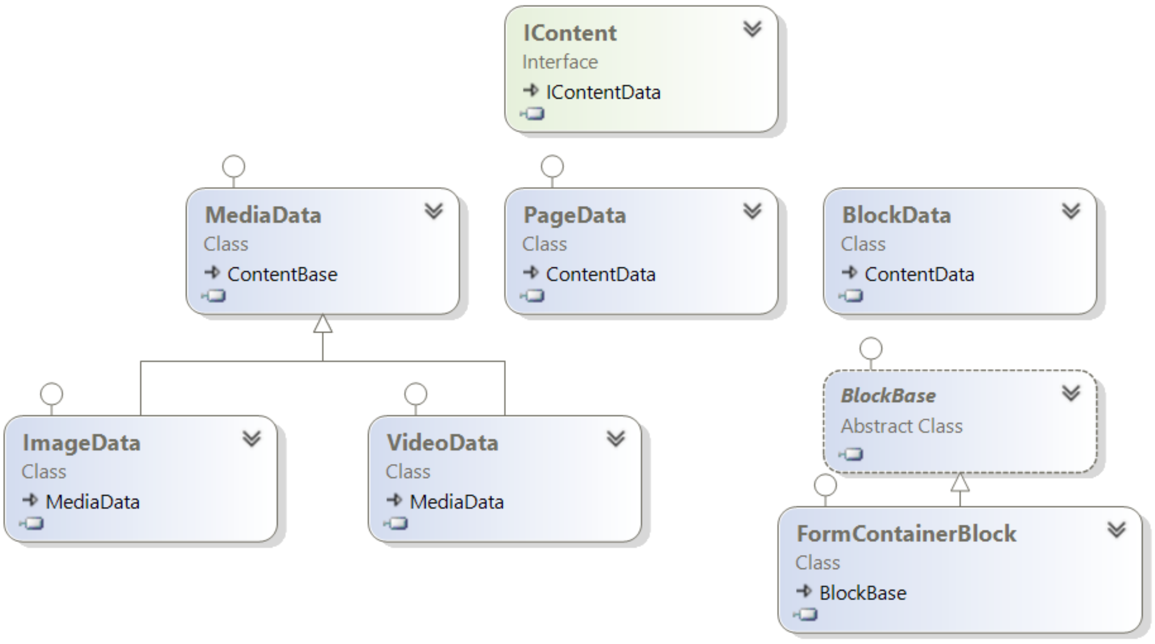
There are four main types of content built-in to Episerver CMS:

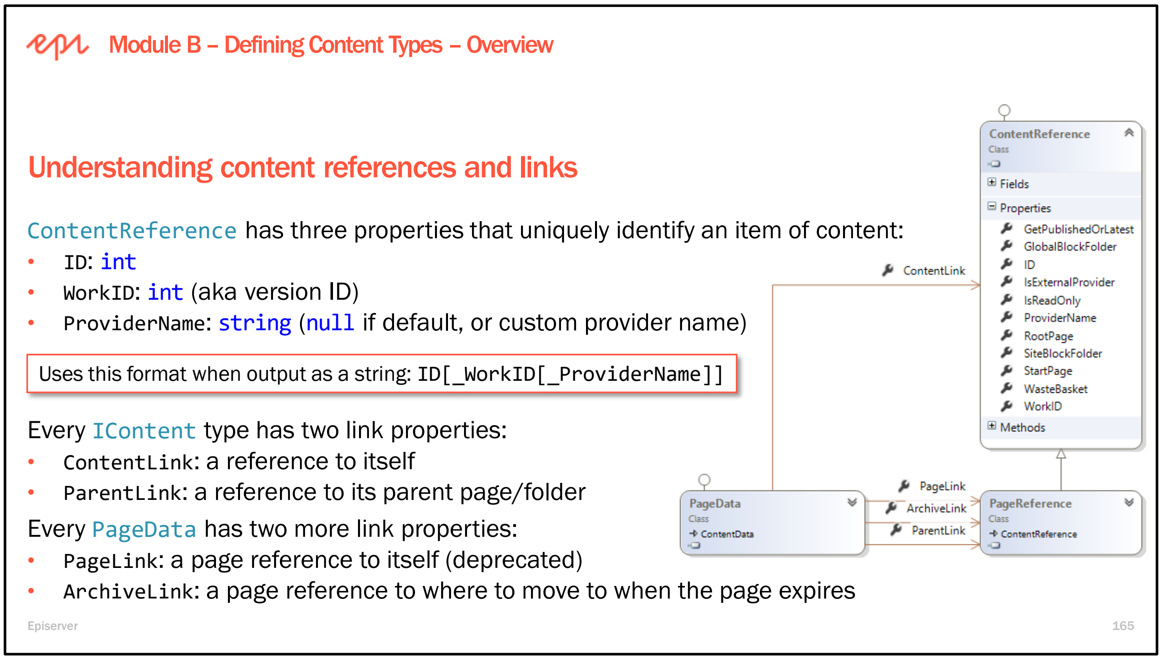
- **Page**: an instance of a class that derives directly or indirectly from `PageData`
- **Folder**: an instance of `ContentFolder`
- **Media**: an instance of a class that derives directly or indirectly from `MediaData` or its subclasses `ImageData` and `VideoData`
- **Block**: an instance of a class that derives directly or indirectly from `BlockData`
  - Episerver Forms use `FormContainerBase` which inherits from `BlockData` so forms are treated as a special type of block

Episerver 164

The following functionality is available for all content types in Episerver CMS

- Waste basket support, including moving, viewing and restoring from trash.
- Checking of references when deleting any content, that shows a dialog with links to affected content.
- Drag and drop support from the assets pane to any overlay or property that handles content references or URLs.





Module B – Defining Content Types – Overview

Understanding content references and links

`ContentReference` has three properties that uniquely identify an item of content:

- `ID`: `int`
- `WorkID`: `int` (aka version ID)
- `ProviderName`: `string` (`null` if default, or custom provider name)

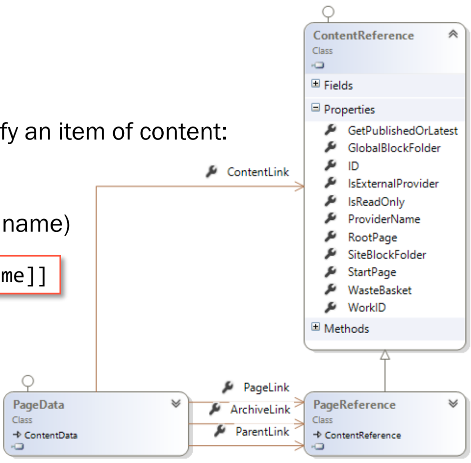
Uses this format when output as a string: `ID[_WorkID[_ProviderName]]`

Every `IContent` type has two link properties:

- `ContentLink`: a reference to itself
- `ParentLink`: a reference to its parent page/folder

Every `PageData` has two more link properties:

- `PageLink`: a page reference to itself (deprecated)
- `ArchiveLink`: a page reference to where to move to when the page expires



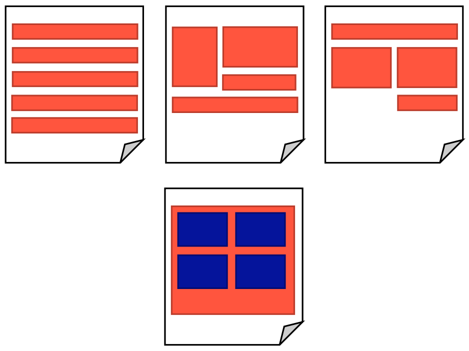
Episerver 165

Module B – Defining Content Types – Overview

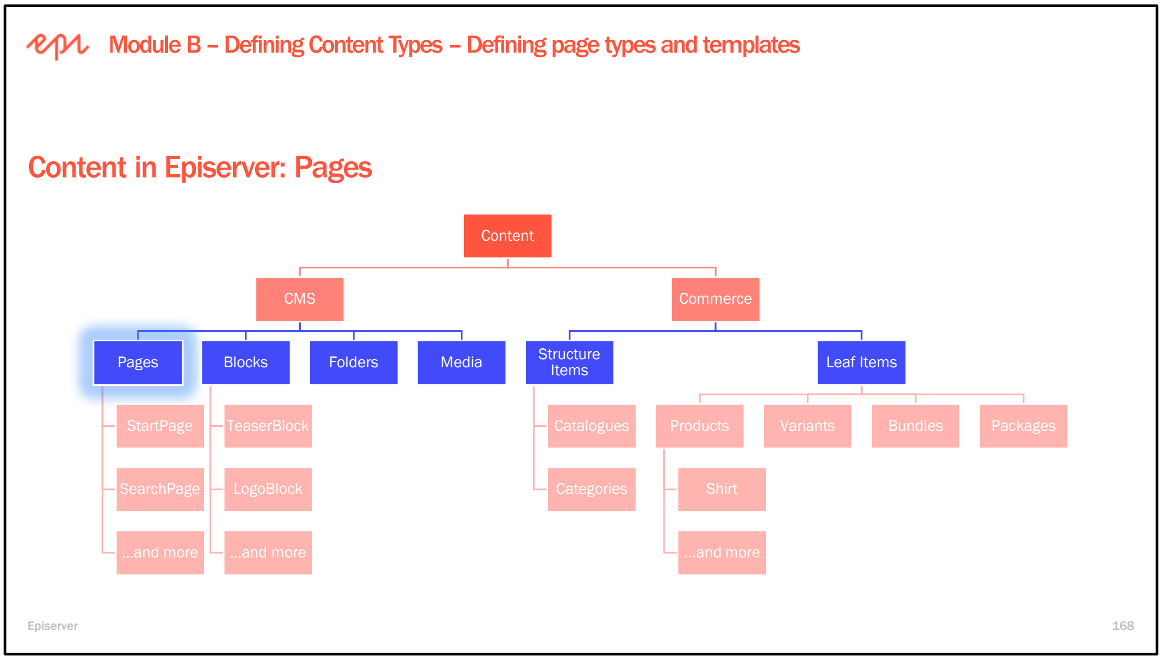
As a general rule, make the start page and any landing pages flexible by using content areas and make all other pages more strict.

Two extremes of website design

- **Many pages, no blocks**: some websites are designed to have strict layouts for the pages by defining dozens of specific page types with many simple data type properties and views that output the property values in fixed locations in the page template.
- **Few pages, many blocks**: Other websites are designed to have very flexible layouts for the pages by having only a few page types, with one general page type with either a **ContentArea** or **XhtmlString** property to allow the editor to add any combination of rich text, images, and blocks designed to be used anyway without restrictions.

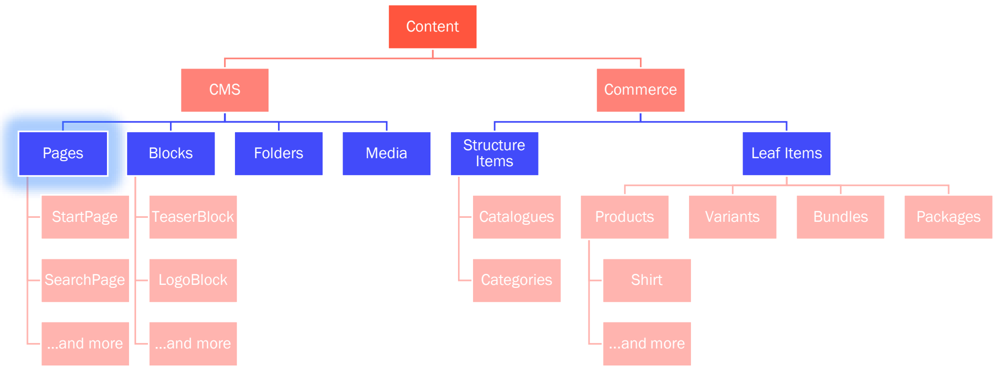


Episerver 166

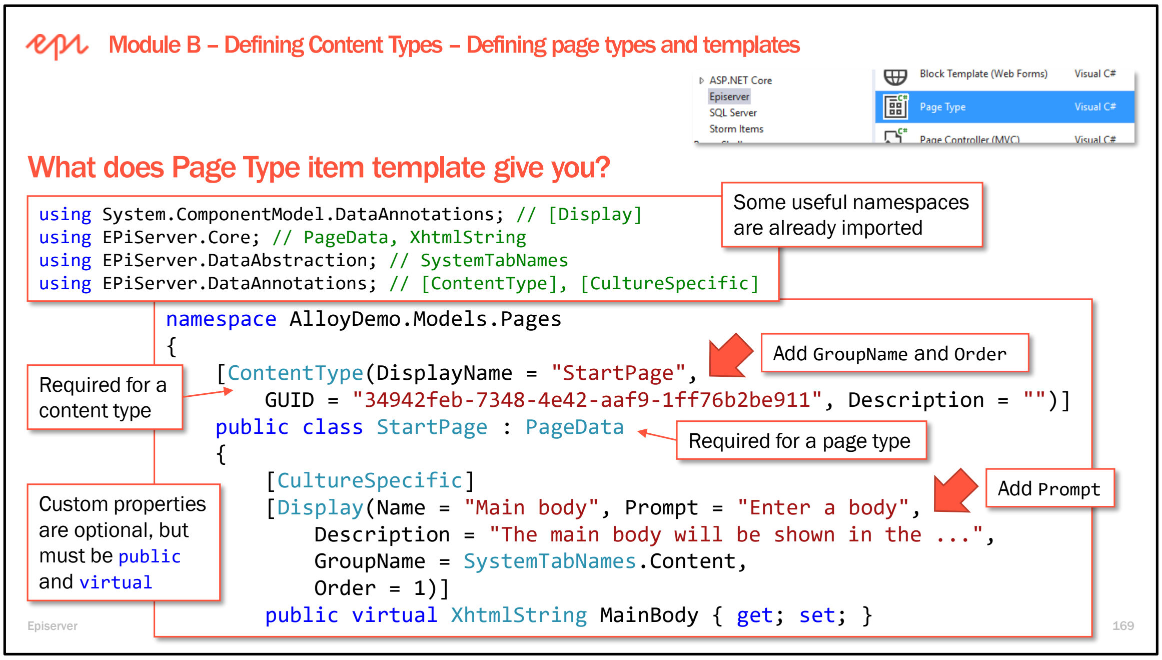


Module B – Defining Content Types – Defining page types and templates

Content in Episerver: Pages



Episerver 168



Module B – Defining Content Types – Defining page types and templates

ASP.NET Core
Episerver
SQL Server
Storm Items

Block Template (Web Forms)	Visual C#
Page Type	Visual C#
Page Controller (MVC)	Visual C#

What does Page Type item template give you?

```
using System.ComponentModel.DataAnnotations; // [Display]
using EPiServer.Core; // PageData, XhtmlString
using EPiServer.DataAbstraction; // SystemTabNames
using EPiServer.DataAnnotations; // [ContentType], [CultureSpecific]
namespace AlloyDemo.Models.Pages
{
    [ContentType(DisplayName = "StartPage",
        GUID = "34942feb-7348-4e42-aaf9-1ff76b2be911", Description = "")]
    public class StartPage : PageData
    {
        [CultureSpecific]
        [Display(Name = "Main body", Prompt = "Enter a body",
            Description = "The main body will be shown in the ...",
            GroupName = SystemTabNames.Content,
            Order = 1)]
        public virtual XhtmlString MainBody { get; set; }
```

Some useful namespaces are already imported

Add GroupName and Order

Required for a content type

Required for a page type

Add Prompt

Custom properties are optional, but must be public and virtual

Episerver

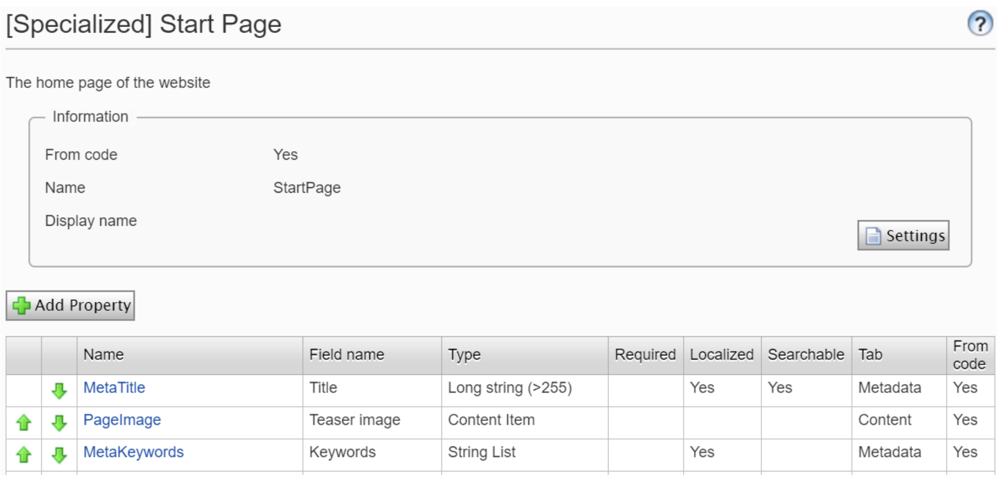
169

The page type is available as a project item type in the Episerver CMS Visual Studio Extension.

Created from code:

- A .NET class that inherits **EPiServer.Core.PageData**
- Decorated with ContentType attribute
- Will be registered in the Episerver Database when the website is initialized
- Becomes the default MVC model passed into your page controller

Page types can also be created from Admin, but this is for legacy reasons and not recommended because page types created in this way are not strongly typed. A strongly typed page type will have “From code” set to “Yes” in Admin, while a page type created from Admin has no indication of being created from code, and the page type name will be editable.



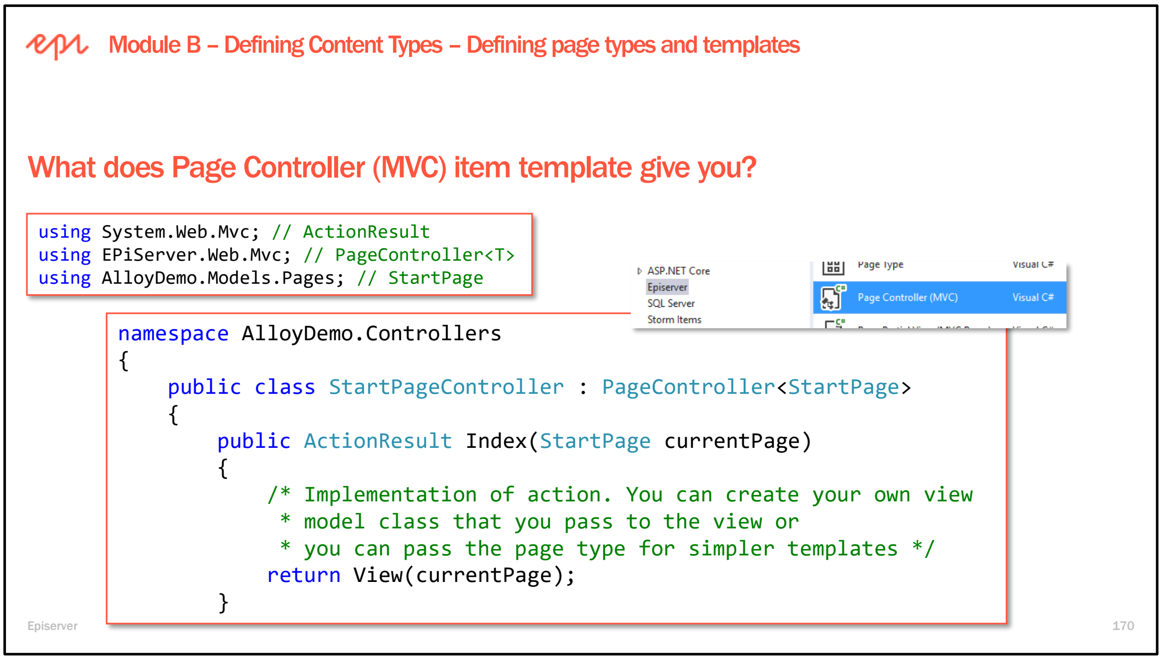
[Specialized] Start Page

The home page of the website

Information

From code	Yes
Name	StartPage
Display name	

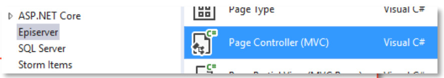
		Name	Field name	Type	Required	Localized	Searchable	Tab	From code
	⇩	MetaTitle	Title	Long string (>255)		Yes	Yes	Metadata	Yes
⇧	⇩	PageImage	Teaser image	Content Item				Content	Yes
⇧	⇩	MetaKeywords	Keywords	String List		Yes		Metadata	Yes



Module B – Defining Content Types – Defining page types and templates

What does Page Controller (MVC) item template give you?

```
using System.Web.Mvc; // ActionResult
using EPiServer.Web.Mvc; // PageController<T>
using AlloyDemo.Models.Pages; // StartPage
```



```
namespace AlloyDemo.Controllers
{
    public class StartPageController : PageController<StartPage>
    {
        public ActionResult Index(StartPage currentPage)
        {
            /* Implementation of action. You can create your own view
             * model class that you pass to the view or
             * you can pass the page type for simpler templates */
            return View(currentPage);
        }
```

Episerver 170

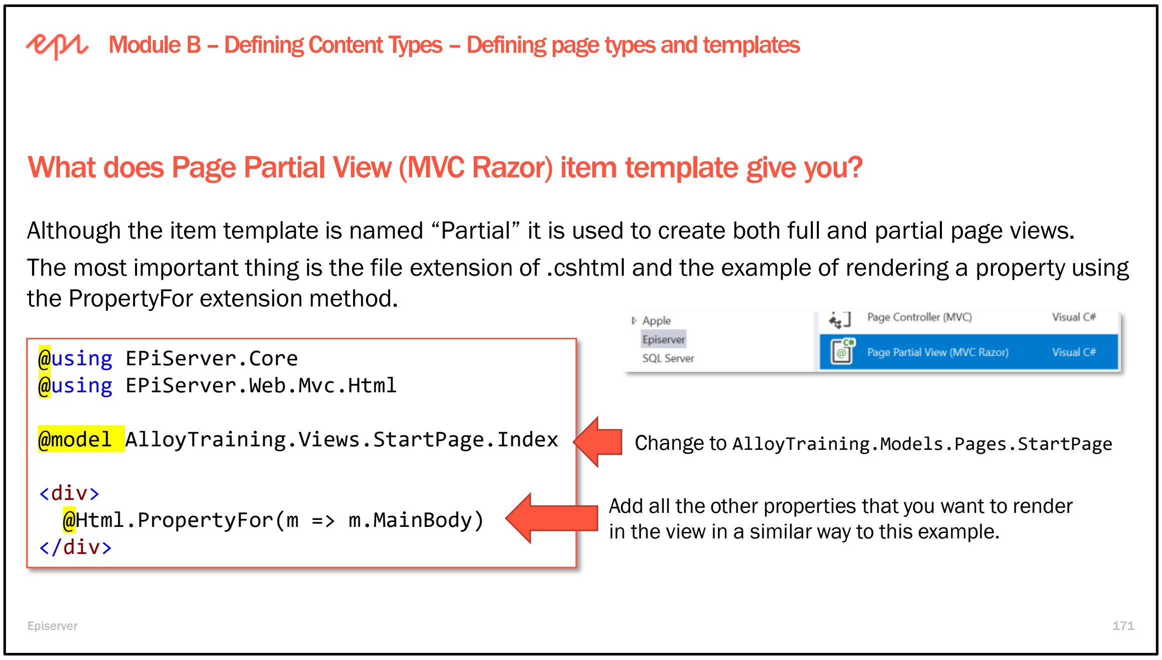
A Page Template generates output for pages of a given type

A page template is responsible for rendering a page type. In Episerver a template defines which page types it can render, not the other way around, which gives a clean separation of model and presentation and allows for easier extensibility. In MVC, the template consists of a controller and a view, where the controller contains the business logic and selects the view, and the view presents the content model.

When the strongly typed page type template is used, the rendering template will automatically be registered as a supported template for the specified page type (T) as long as the naming has the following convention:

- Page type name = Models/Pages/<something>Page.cs (example: “StandardPage.cs”)
- Page type Controller name = Controllers/<something>Controller.cs (example: “StandardPageController.cs”)
- Page type View name =Views/<something>Page/Index.cshtml (example: “StandardPage/Index.cshtml”)

To make the template supported for all page types in the system, use EPiServer.Core.PageData as the generic type (T).



Module B – Defining Content Types – Defining page types and templates

What does Page Partial View (MVC Razor) item template give you?

Although the item template is named "Partial" it is used to create both full and partial page views.

The most important thing is the file extension of .cshtml and the example of rendering a property using the PropertyFor extension method.

Apple
Episerver
SQL Server
Page Controller (MVC) Visual C#
Page Partial View (MVC Razor) Visual C#

```
@using EPiServer.Core
@using EPiServer.Web.Mvc.Html

@model AlloyTraining.Views.StartPage.Index

<div>
  @Html.PropertyFor(m => m.MainBody)
</div>
```

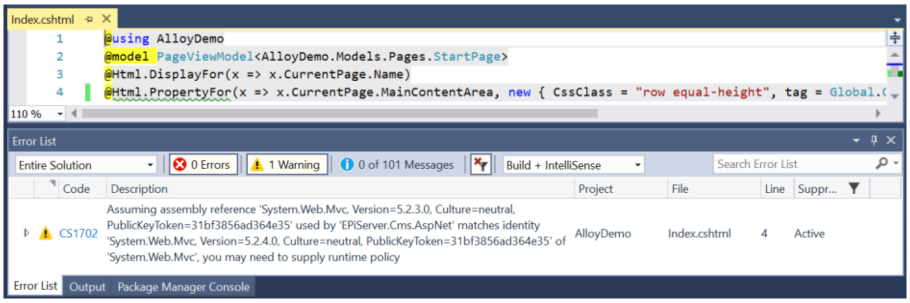
Change to `AlloyTraining.Models.Pages.StartPage`

Add all the other properties that you want to render in the view in a similar way to this example.

Episerver 171

Suppressing compiler warning CS1702

The EPiServer assemblies are currently compiled for Microsoft.AspNet.Mvc 5.2.3 but you are probably using a later version like 5.2.4. This causes compiler warnings when Razor views are open if they call Episerver extension methods like `PropertyFor()`



Index.cshtml

```
@using AlloyDemo
@model PageViewModel<AlloyDemo.Models.Pages.StartPage>
@Html.DisplayFor(x => x.CurrentPage.Name)
@Html.PropertyFor(x => x.CurrentPage.MainContentArea, new { CssClass = "row equal-height", tag = Global.(
```

Error List — Entire Solution | 0 Errors | 1 Warning | 0 of 101 Messages | Build + IntelliSense

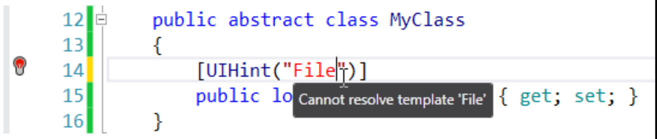
Code	Description	Project	File	Line	Suppr...
CS1702	Assuming assembly reference 'System.Web.Mvc, Version=5.2.3.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' used by 'EPiServer.Cms.AspNet' matches identity 'System.Web.Mvc, Version=5.2.4.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' of 'System.Web.Mvc', you may need to supply runtime policy	AlloyDemo	Index.cshtml	4	Active

Until Episerver releases new assemblies, you can suppress the CS1702 warning by adding the following to the root Web.config:

```
<system.codedom>
  <compilers>
    <compiler language="c#;cs;csharp"
              extension=".cshtml"
              compilerOptions="/nowarn:1702"
              type="Microsoft.CSharp.CSharpCodeProvider, System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
              warningLevel="4">
      <providerOption name="CompilerVersion" value="v4.0" />
      <providerOption name="WarnAsError" value="false" />
    </compiler>
  </compilers>
</system.codedom>
```

Module B – Defining Content Types – Defining page types and templates

Supporting ReSharper



```
public abstract class MyClass
{
    [UIHint("File")]
    public lo  Cannot resolve template 'File'  { get; set; }
}
```

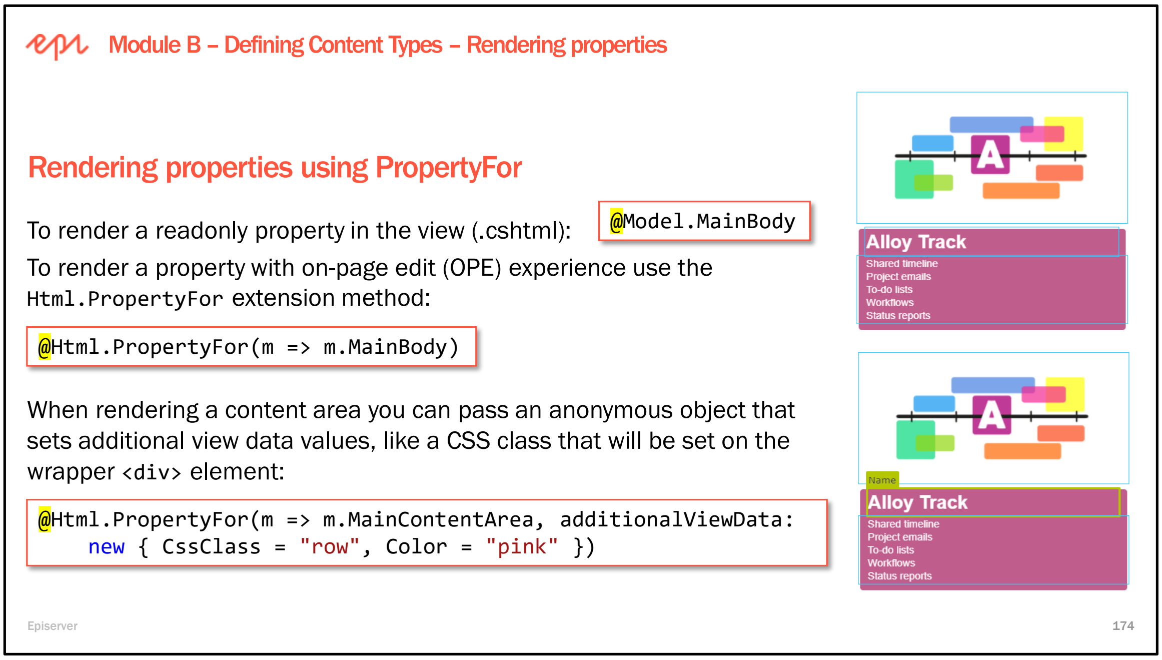
Many developers use ReSharper make Visual Studio a better IDE by providing better code analysis, generation, navigation, formatting, and refactoring.

In a default installation of ReSharper, it will show warnings about resolving templates, as described in the following Stackoverflow post:
https://stackoverflow.com/questions/24104526/uihint-can-not-resolve-template-in-abstract-models

Other useful articles about Episerver and ReSharper:

- Creating EPiServer Page Types using ReSharper File Templates:
  https://www.dcaric.com/blog/creating-episerver-page-types-using-resharper-file-templates
- Resharper templates for EPiServer properties:
  https://www.dcaric.com/blog/resharper-templates-for-episerver-properties

Episerver 172



Module B – Defining Content Types – Rendering properties

Rendering properties using PropertyFor

To render a readonly property in the view (.cshtml):

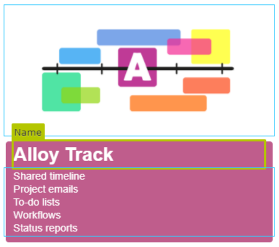
```
@Model.MainBody
```

To render a property with on-page edit (OPE) experience use the `Html.PropertyFor` extension method:

```
@Html.PropertyFor(m => m.MainBody)
```

When rendering a content area you can pass an anonymous object that sets additional view data values, like a CSS class that will be set on the wrapper `<div>` element:

```
@Html.PropertyFor(m => m.MainContentArea, additionalViewData:
    new { CssClass = "row", Color = "pink" })
```



Episerver 174

To read the additional view data, use `ViewContext.ParentActionViewContext.ViewData`, as shown in the following code:

~\Views\Shared_additionalViewData.cshtml

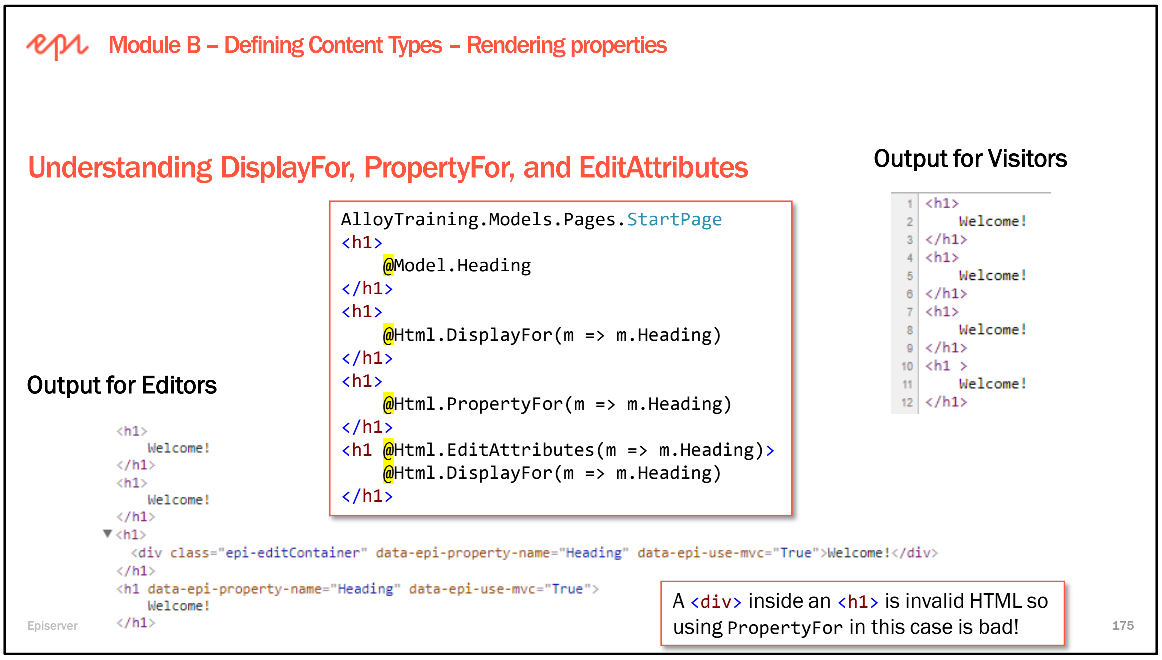
```
<h4>additionalViewData</h4>
<ul>
    @foreach (KeyValuePair<string, object> item
        in ViewContext.ParentActionViewContext.ViewData)
    {
        <li>@item.Key: @item.Value</li>
    }
</ul>
```

Inside a partial content template view:

```
@Html.Partial("_additionalViewData")
```

Rendering a content area and passing some additional view data to all the partial templates:

```
@Html.PropertyFor(m => m.MainContentArea,
    additionalViewData: new { CssClass = "row", Color = "pink" })
```



Module B – Defining Content Types – Rendering properties

Understanding DisplayFor, PropertyFor, and EditAttributes

```
AlloyTraining.Models.Pages.StartPage
<h1>
    @Model.Heading
</h1>
<h1>
    @Html.DisplayFor(m => m.Heading)
</h1>
<h1>
    @Html.PropertyFor(m => m.Heading)
</h1>
<h1 @Html.EditAttributes(m => m.Heading)>
    @Html.DisplayFor(m => m.Heading)
</h1>
```

Output for Visitors

```
<h1>
    Welcome!
</h1>
<h1>
    Welcome!
</h1>
<h1>
    Welcome!
</h1>
<h1 >
    Welcome!
</h1>
```

Output for Editors

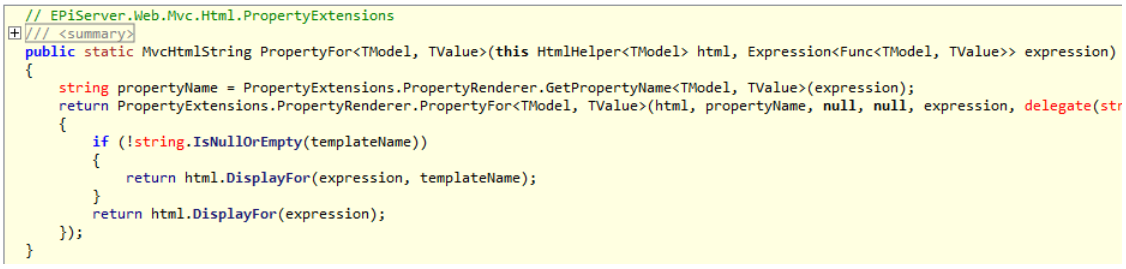
```
<h1>
    Welcome!
</h1>
<h1>
    Welcome!
</h1>
▼<h1>
    <div class="epi-editContainer" data-epi-property-name="Heading" data-epi-use-mvc="True">Welcome!</div>
</h1>
<h1 data-epi-property-name="Heading" data-epi-use-mvc="True">
    Welcome!
</h1>
```

A `<div>` inside an `<h1>` is invalid HTML so using `PropertyFor` in this case is bad!

Episerver

175

For visitors, PropertyFor simply calls Microsoft's DisplayFor:

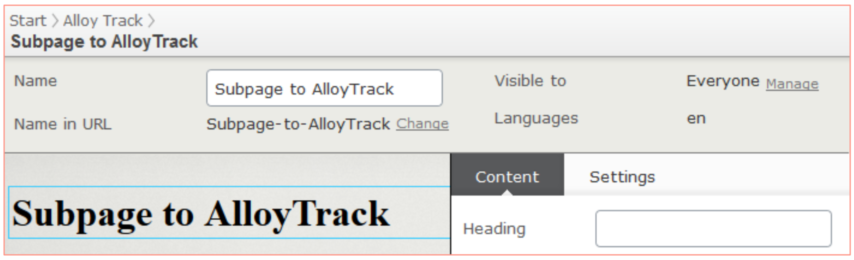


```
// EPiServer.Web.Mvc.Html.PropertyExtensions
/// <summary>
public static MvcHtmlString PropertyFor<TModel, TValue>(this HtmlHelper<TModel> html, Expression<Func<TModel, TValue>> expression)
{
    string propertyName = PropertyExtensions.PropertyRenderer.GetPropertyName<TModel, TValue>(expression);
    return PropertyExtensions.PropertyRenderer.PropertyFor<TModel, TValue>(html, propertyName, null, null, expression, delegate(str
    {
        if (!string.IsNullOrEmpty(templateName))
        {
            return html.DisplayFor(expression, templateName);
        }
        return html.DisplayFor(expression);
    });
}
```

Module B – Defining Content Types – Rendering properties

Taking control of on-page editing using EditAttributes

```
@* Render the Heading property but if it’s empty render Name instead *@
<h1 @Html.EditAttributes(x => x.Heading)>
        @(Model.CurrentPage.Heading ?? Model.CurrentPage.Name)
</h1>
```



Start › Alloy Track ›
Subpage to AlloyTrack

Name	Subpage to AlloyTrack	Visible to	Everyone Manage
Name in URL	Subpage-to-AlloyTrack Change	Languages	en

Episerver 176

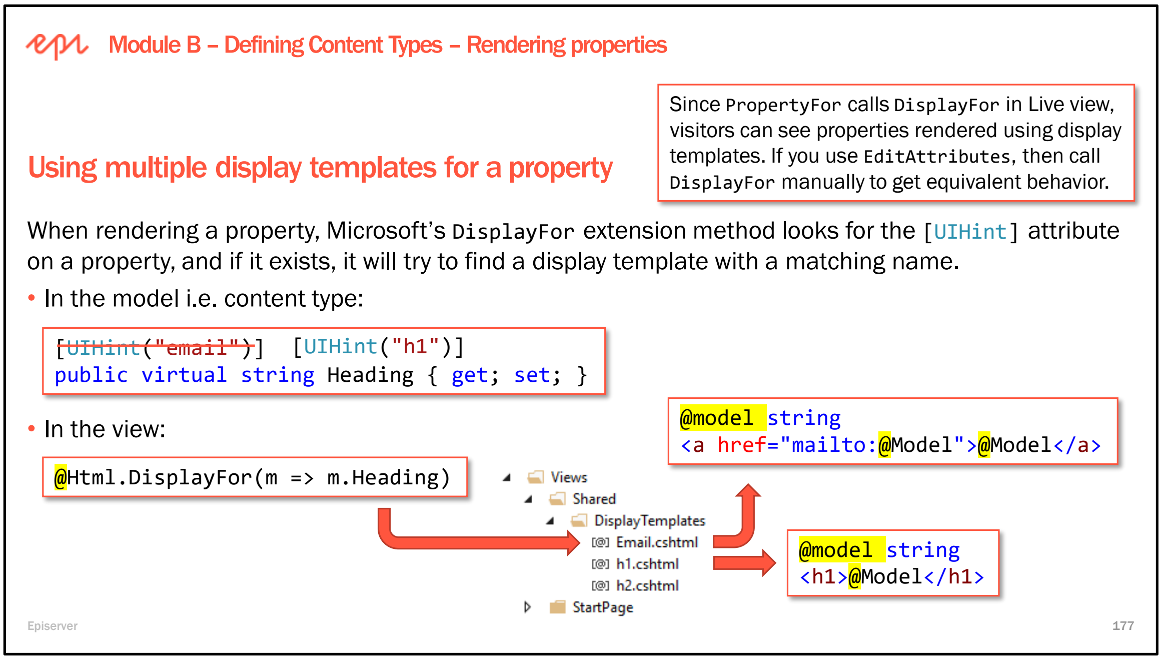
Null values
Episerver properties with an empty value are never stored in the database. If you access it from code, it will always be null – not an empty string, 0 or false as you maybe expected. Why null? It is by design and is very convenient if you want to check if something is not set by an editor or does not exist on this page. You just have to compare with null regardless of data type.

Using fallbacks
Always use fallbacks when working with Episerver properties and especially when rendering them out to the visitor (in inline code or code-behind). For example: in a page that has a user-defined property called Heading, use the built-in property Name to display the name of the page if the Heading value is missing: @(Model.Heading ?? Model.Name)

More information:
Best coding practices for Episerver properties: http://world.episerver.com/Articles/Items/Best-Coding-Practices/

Because the <h1> tag is connected to the Heading property in the back-end it is the value of the Heading property, not the Name, that will be updated when the editor makes changes to the property in Edit View.

More information:
A detailed version of this example can be found in the Episerver CMS Developer Guide: http://world.episerver.com/documentation/developer-guides/CMS/Content/Edit-hints-in-MVC/



Module B – Defining Content Types – Rendering properties

Using multiple display templates for a property

Since `PropertyFor` calls `DisplayFor` in Live view, visitors can see properties rendered using display templates. If you use `EditAttributes`, then call `DisplayFor` manually to get equivalent behavior.

When rendering a property, Microsoft's `DisplayFor` extension method looks for the `[UIHint]` attribute on a property, and if it exists, it will try to find a display template with a matching name.

- In the model i.e. content type:

~~[UIHint("email")]~~ `[UIHint("h1")]`

```
public virtual string Heading { get; set; }
```

- In the view:

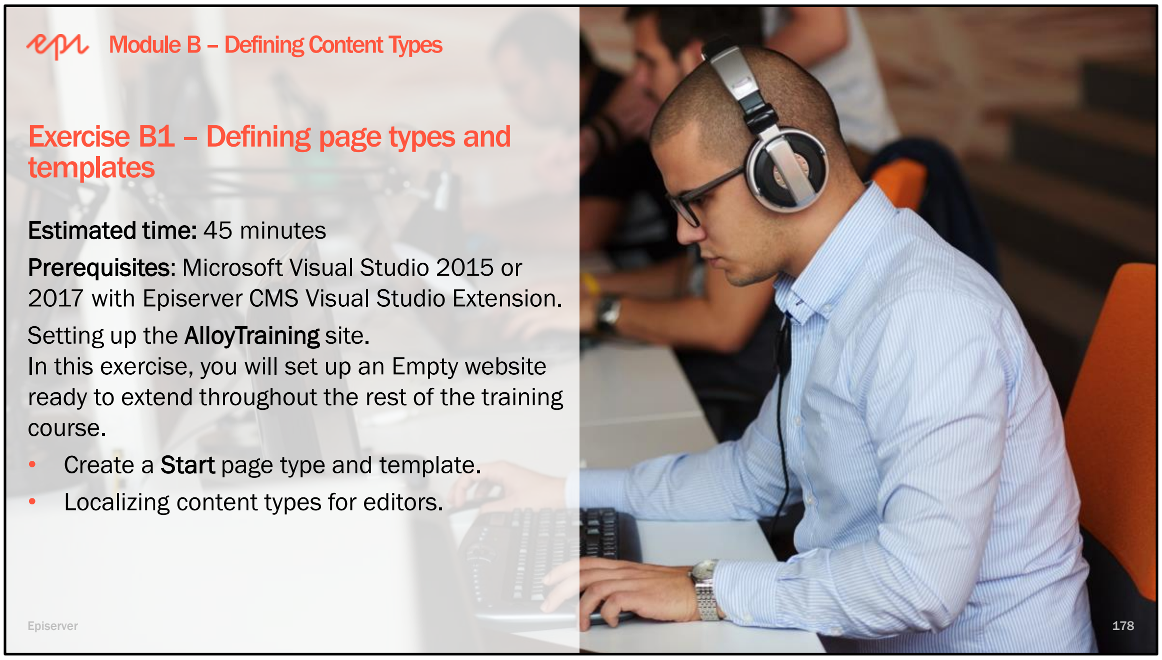
```
@Html.DisplayFor(m => m.Heading)
```

```
Views
  Shared
    DisplayTemplates
      Email.cshtml
      h1.cshtml
      h2.cshtml
  StartPage
```

```
@model string
<a href="mailto:@Model">@Model</a>
```

```
@model string
<h1>@Model</h1>
```

Episerver 177



Module B – Defining Content Types

Exercise B1 – Defining page types and templates

Estimated time: 45 minutes

Prerequisites: Microsoft Visual Studio 2015 or 2017 with Episerver CMS Visual Studio Extension.

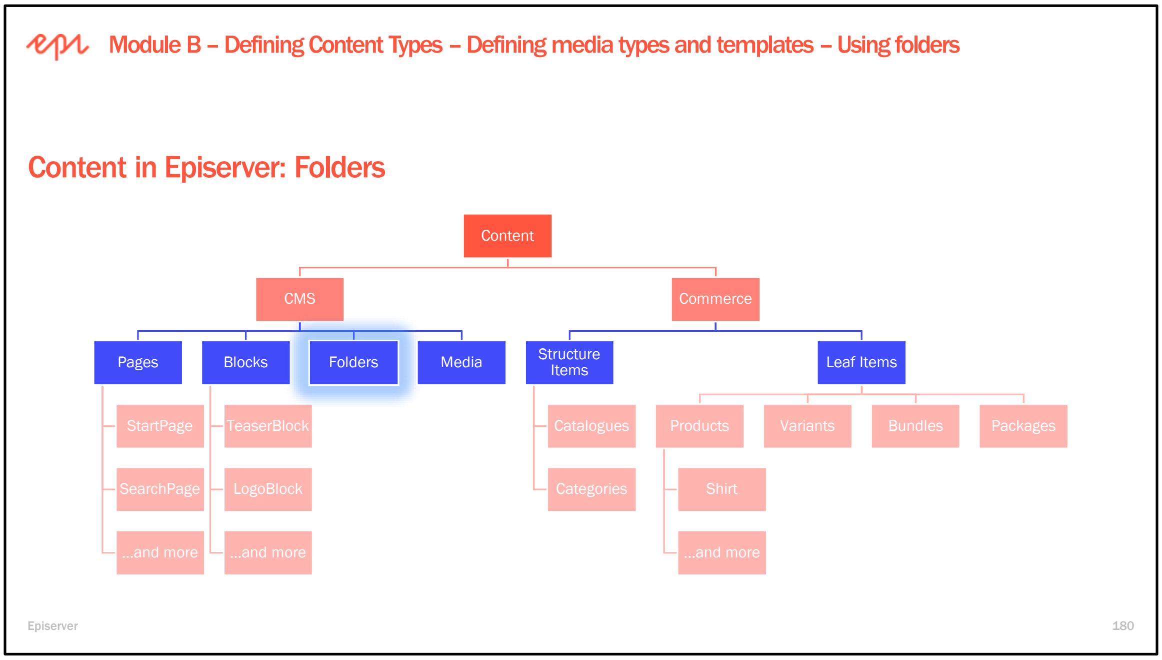
Setting up the **AlloyTraining** site.
In this exercise, you will set up an Empty website ready to extend throughout the rest of the training course.

- Create a **Start** page type and template.
- Localizing content types for editors.

Episerver 178

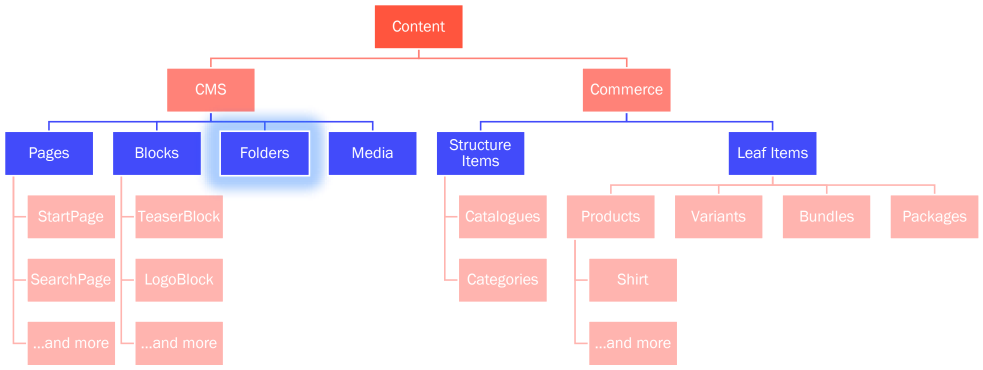
Troubleshooting problems with creating a new CMS project
The course environment in the classrooms on Episerver premises has a known working development tool setup that is based on the system requirements for the product available on Episerver World.
If the course is run in another location the setup is likely to differ on development tool versions, access rights etc. and this may cause problems when trying to create a new Episerver CMS website. Below are some useful references with solutions to the most common setup issues:

- http://world.episerver.com/Blogs/Jeff-Wallace/Dates/2012/12/Visual-Studio-Extension--Error-When-Creating-a-New-Site/
- http://world.episerver.com/Blogs/Eric-Pettersson/Dates/2012/12/Failed-to-register-URL-for-your-website-when-using-VS2012-and-IIS-Express/
- The Windows user account that is logged in when creating the new website must be an administrator in SQL Server in order to be able to create the new database. Check in SQL Management Studio (under Security>Logins) that the windows account is present and has the Server Role “sysadmin” selected.



Module B – Defining Content Types – Defining media types and templates – Using folders

Content in Episerver: Folders



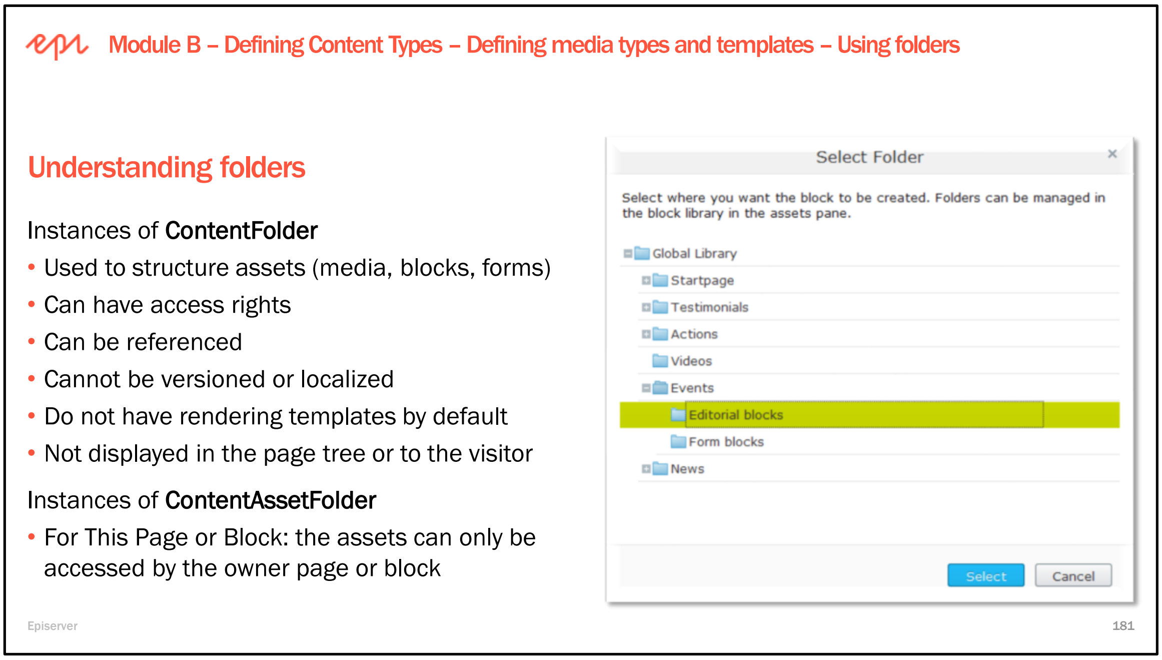
Episerver 180

EPiServer.Core.ContentFolder:

- Used to structure content and has no visual appearance on the site.

EPiServer.Core.ContentAssetFolder:

- Inherits from ContentFolder.
- Used to host assets related to a specific content item, e.g. **For This Page** and **For This Block**.
- Resources stored as content assets are to be seen as exclusive assets for that content instance and hence the resources are not selectable from other content instances.



Module B – Defining Content Types – Defining media types and templates – Using folders

Understanding folders

Instances of **ContentFolder**

- Used to structure assets (media, blocks, forms)
- Can have access rights
- Can be referenced
- Cannot be versioned or localized
- Do not have rendering templates by default
- Not displayed in the page tree or to the visitor

Instances of **ContentAssetFolder**

- For This Page or Block: the assets can only be accessed by the owner page or block

Select Folder

Select where you want the block to be created. Folders can be managed in the block library in the assets pane.

Episerver 181

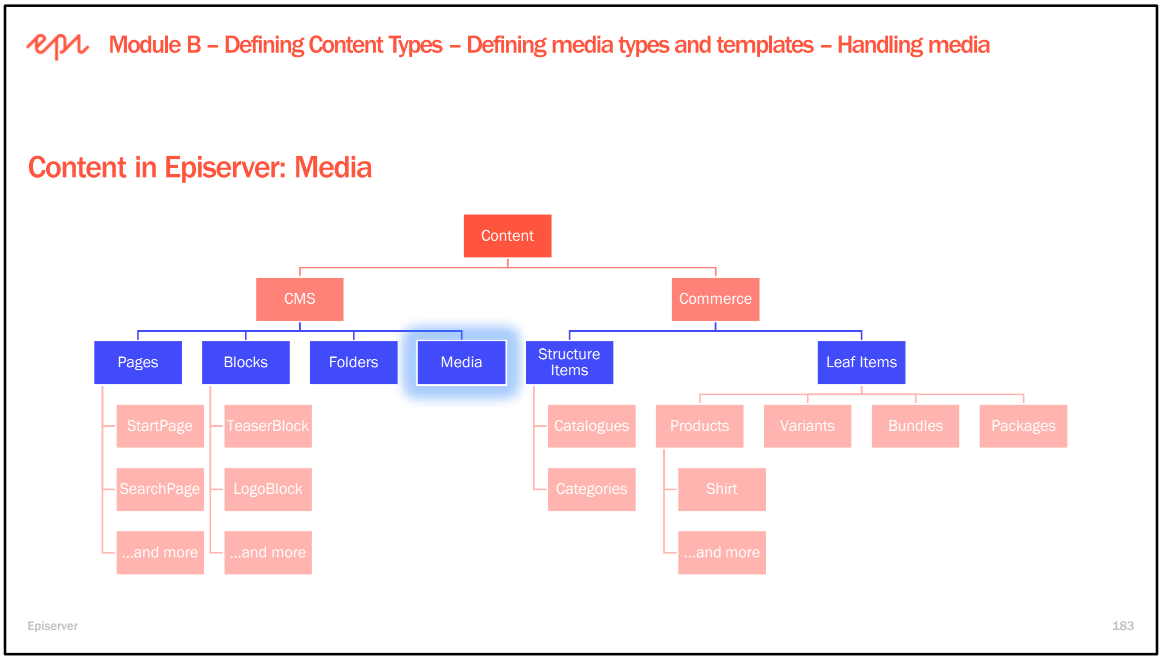
A folder is an instance of EPiServer.Core.ContentFolder and is used to structure content. A content folder does not have any visual appearance on the site.

Structures shared blocks

Shared blocks are structured with use of folders. A folder in the shared blocks structure can have other folders or Shared Blocks as children. A Shared Block can not have any children. The editorial access is set on the folders to specify which folders that should be available for the editor.

Multi-site support

There is a global folder root given by EPiServer.Core.ContentReference.GlobalBlockFolder that is the root folder for Shared Blocks that should be available for all sites in an enterprise scenario. There is also a site specific folder EPiServer.Core.ContentReference.SiteBlockFolder that contains the folder structure for shared blocks that are site specific.



Module B – Defining Content Types – Defining media types and templates – Handling media

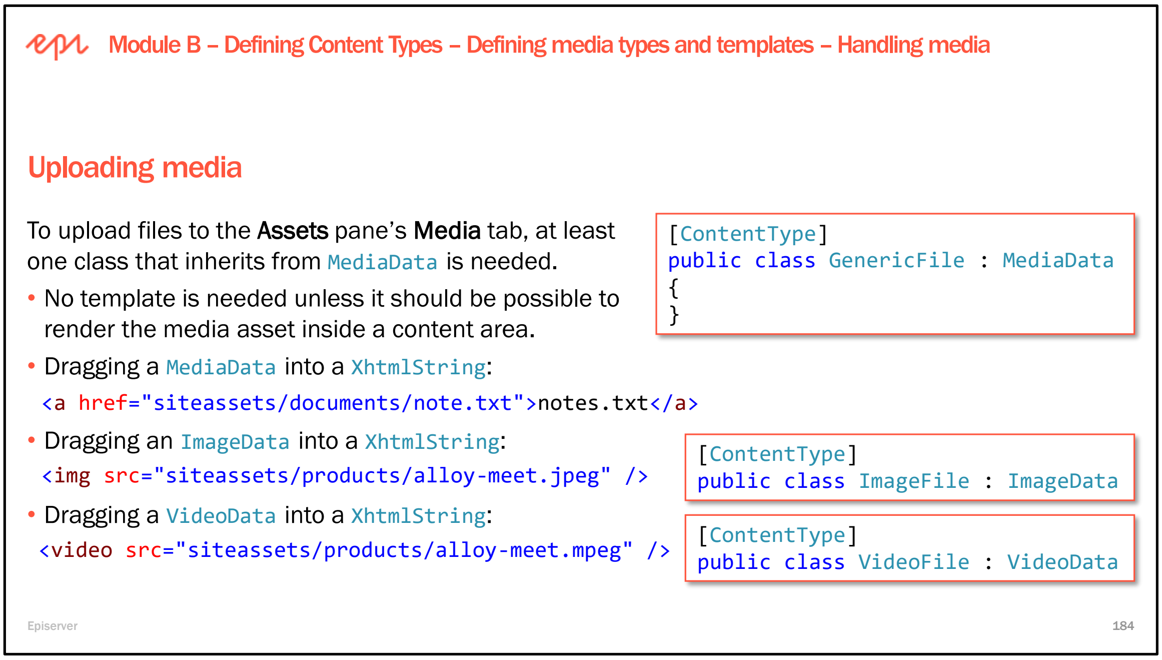
Content in Episerver: Media

Episerver 183

The assets system is based on a typed model with support for the following property types:

- ContentReference property type with a UIHint “image” will be displayed and edited as an image.
- ContentReference property type with a UIHint “video” will be displayed and edited as a video.
- ContentReference property type with a UIHint “mediafile” will be displayed and edited as any file.
- Url property type with a UIHint “image” will be displayed and edited as an image.
- Url property type with a UIHint “video” will be displayed and edited as a video.
- Url property type with a UIHint “document” will be displayed and edited as any file.

A BLOB provider is also available, to make it possible to change storage model for media.
Detailed examples of the basic classes needed to support documents, images and video can be found in the Alloy sample site.



Module B – Defining Content Types – Defining media types and templates – Handling media

Uploading media

To upload files to the **Assets** pane's **Media** tab, at least one class that inherits from `MediaData` is needed.

- No template is needed unless it should be possible to render the media asset inside a content area.
- Dragging a `MediaData` into a `XhtmlString`:

```
<a href="siteassets/documents/note.txt">notes.txt</a>
```

- Dragging an `ImageData` into a `XhtmlString`:

```
<img src="siteassets/products/alloy-meet.jpeg" />
```

- Dragging a `VideoData` into a `XhtmlString`:

```
<video src="siteassets/products/alloy-meet.mpeg" />
```

```
[ContentType]
public class GenericFile : MediaData
{
}
```

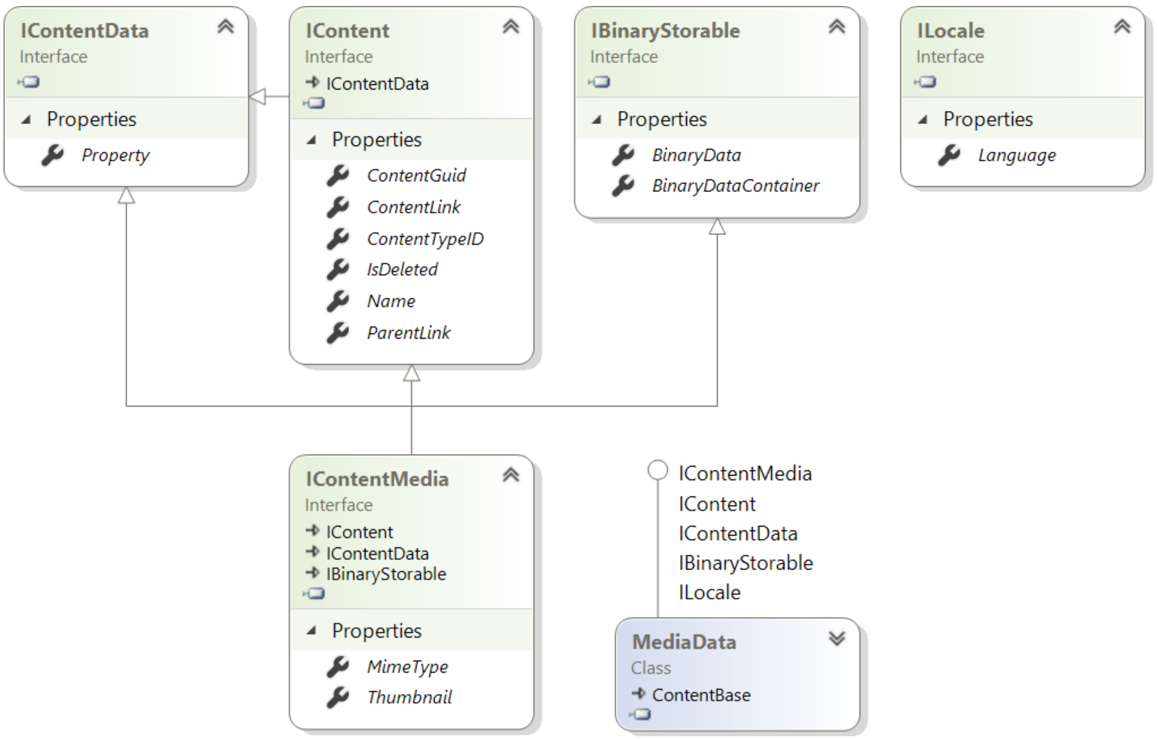
```
[ContentType]
public class ImageFile : ImageData
```

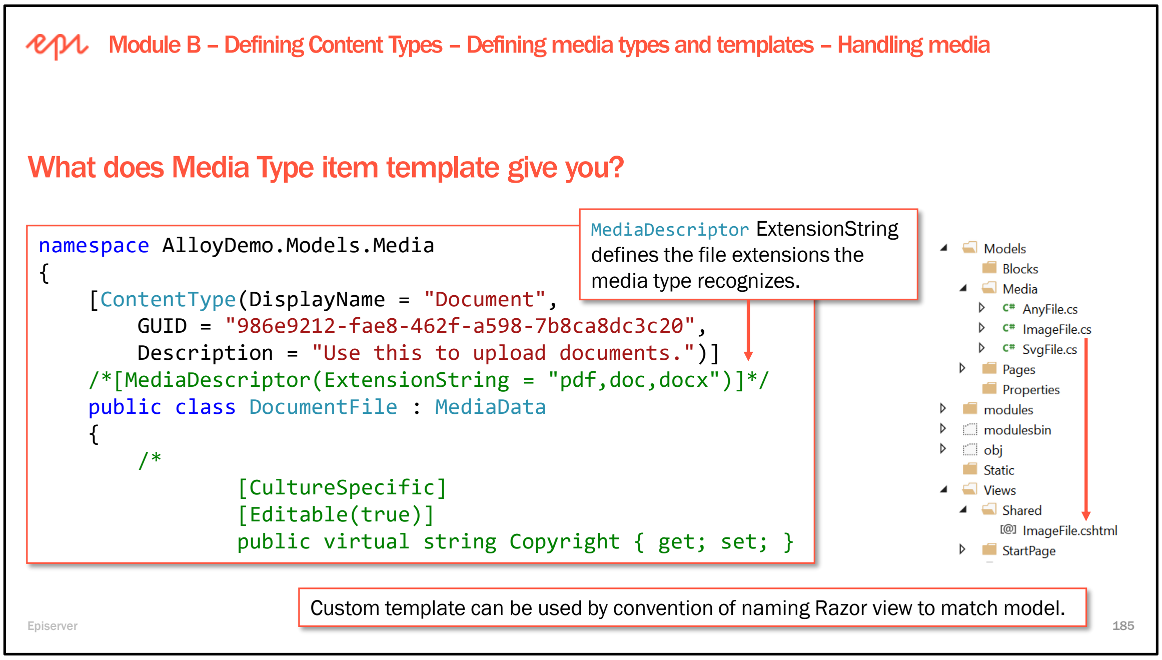
```
[ContentType]
public class VideoFile : VideoData
```

Episerver 184

Media types do not support localization

A media asset has a language but it does not implement ILocalizable so it cannot have language branches.





Module B – Defining Content Types – Defining media types and templates – Handling media

What does Media Type item template give you?

```
namespace AlloyDemo.Models.Media
{
    [ContentType(DisplayName = "Document",
        GUID = "986e9212-fae8-462f-a598-7b8ca8dc3c20",
        Description = "Use this to upload documents.")]
    /*[MediaDescriptor(ExtensionString = "pdf,doc,docx")]*/
    public class DocumentFile : MediaData
    {
        /*
                [CultureSpecific]
                [Editable(true)]
                public virtual string Copyright { get; set; }
```

MediaDescriptor ExtensionString defines the file extensions the media type recognizes.

```
Models
    Blocks
    Media
        AnyFile.cs
        ImageFile.cs
        SvgFile.cs
    Pages
    Properties
modules
modulesbin
obj
Static
Views
    Shared
        ImageFile.cshtml
    StartPage
```

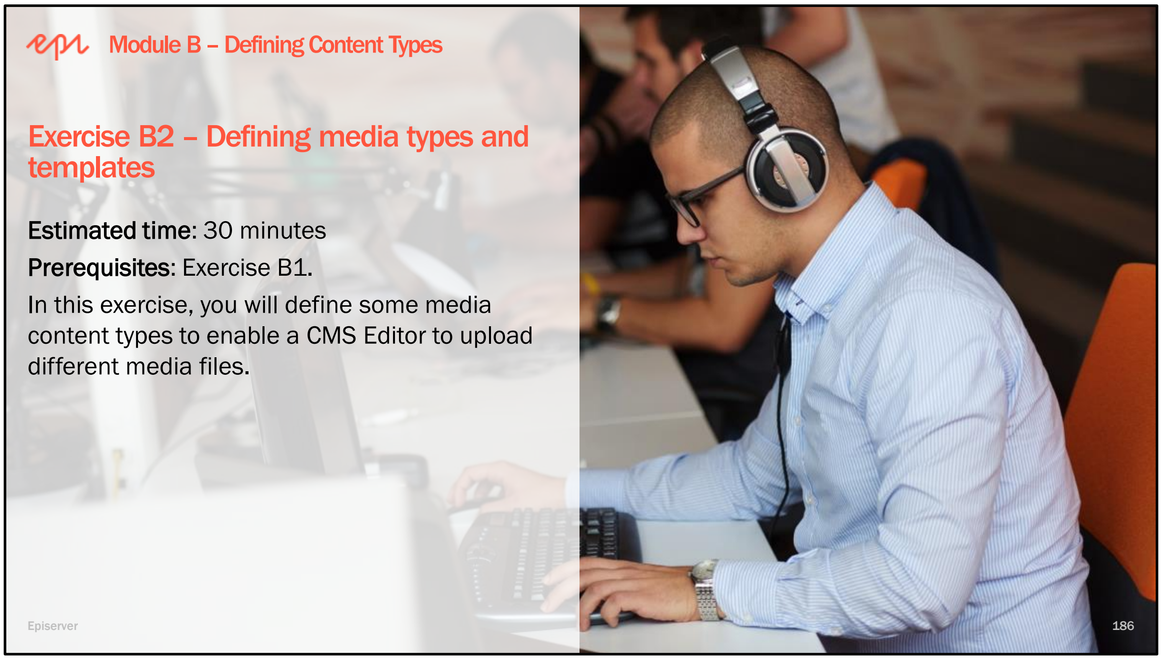
Custom template can be used by convention of naming Razor view to match model.

Episerver 185

Be careful with the extensions that the media types support. If a media item is uploaded when the class has one set of extensions and the extension is later removed, the previously uploaded media will be “broken” in the UI.

Specialized media types
ImageData and VideoData are two specialized classes that allow the system to distinguish images and videos from other generic media in order to apply special handling in the user interface. Both ImageData and VideoData inherit from MediaData. If images or videos are types of media that editors need to deal with regularly then creating content types for them is a good idea.

Media descriptor attribute
As you may have noticed in the ImageFile content type above, there is a MediaDescriptor attribute that defines a list of file extensions. This attribute is used to associate specific file types to a given content type. This allows the system to create content of the correct content type when a user uploads media via the user interface.
When creating media content from the server side it is also possible to have this same content type resolving by using the ContentMediaResolver class.



Module B – Defining Content Types

Exercise B2 – Defining media types and templates

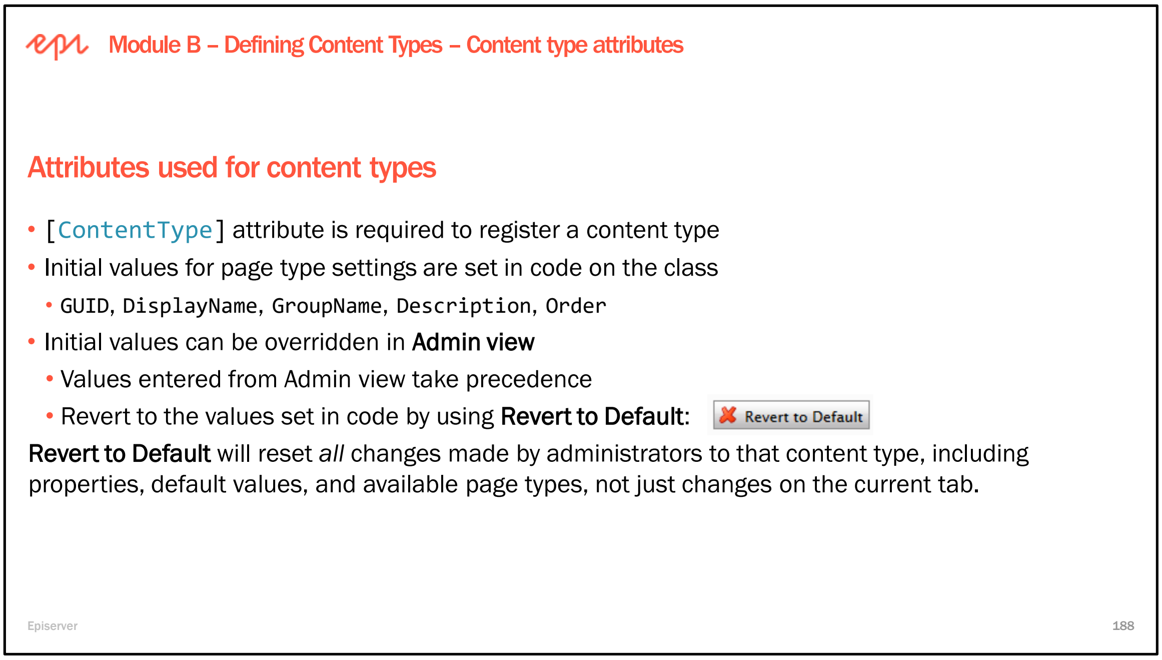
Estimated time: 30 minutes

Prerequisites: Exercise B1.

In this exercise, you will define some media content types to enable a CMS Editor to upload different media files.

Episerver

186



Module B – Defining Content Types – Content type attributes

Attributes used for content types

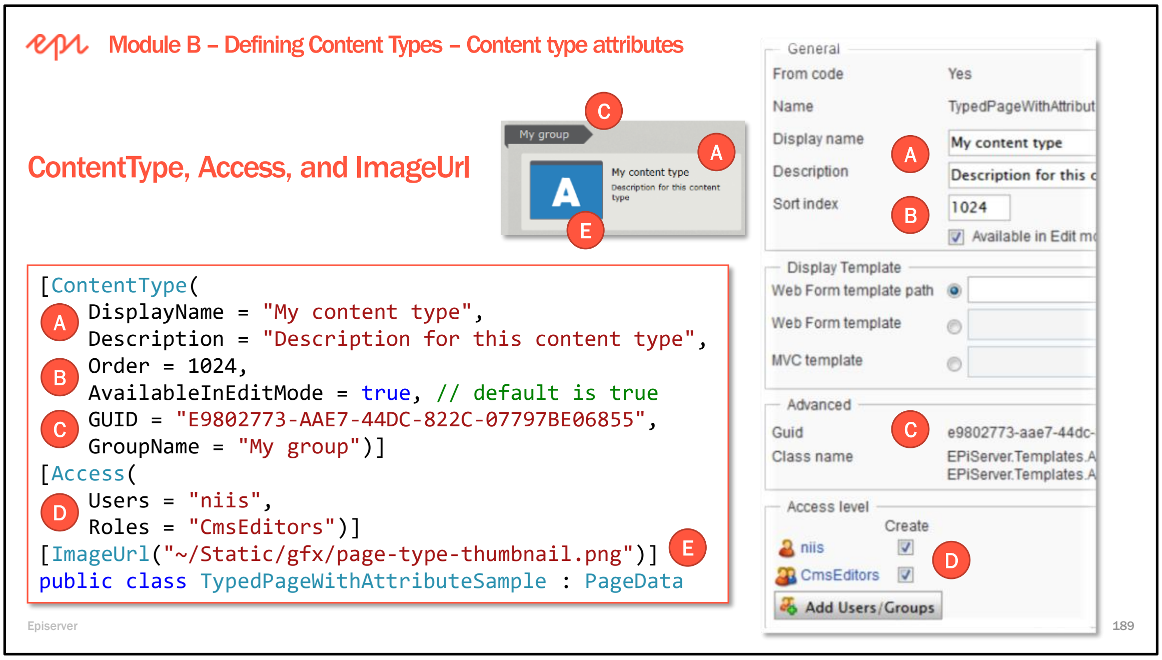
- `[ContentType]` attribute is required to register a content type
- Initial values for page type settings are set in code on the class
  - `GUID`, `DisplayName`, `GroupName`, `Description`, `Order`
- Initial values can be overridden in **Admin view**
  - Values entered from Admin view take precedence
  - Revert to the values set in code by using **Revert to Default**: Revert to Default

Revert to Default will reset *all* changes made by administrators to that content type, including properties, default values, and available page types, not just changes on the current tab.

Episerver 188

Content type attributes are set in code on the strongly typed page types but can be overridden from Admin. There is a function in Admin to revert overridden values to the default values set in the strongly typed page type. It is available as a **Revert to Default** button on the **Page Type Settings** and **Property Settings** pages.
In this section of the course the most commonly used attributes are mentioned. The complete list of available attributes and their default values are available in the CMS SDK, under Developer Guide > Content > Pages and Blocks > Attributes.
Further information about working with a mix of code and Admin mode Page Type configuration:

- In the CMS SDK: Knowledge Base > Developer Guide > Content > Pages and Blocks > Synchronisation
- Blog article on Synchronisation of typed models, available on Episerver World: http://world.episerver.com/Blogs/Per-Bjurstrom/Archive/2012/10/Synchronization-of-typed-models/



Module B – Defining Content Types – Content type attributes

ContentType, Access, and ImageUrl

```
[ContentType(
    DisplayName = "My content type",
    Description = "Description for this content type",
    Order = 1024,
    AvailableInEditMode = true, // default is true
    GUID = "E9802773-AAE7-44DC-822C-07797BE06855",
    GroupName = "My group")]
[Access(
    Users = "niis",
    Roles = "CmsEditors")]
[ImageUrl("~/Static/gfx/page-type-thumbnail.png")]
public class TypedPageWithAttributeSample : PageData
```

My group

My content type
Description for this content type

General

From code	Yes
Name	TypedPageWithAttribut
Display name	My content type
Description	Description for this c
Sort index	1024
	✓ Available in Edit m

Display Template

Web Form template path	●
Web Form template	○
MVC template	○

Advanced

Guid	e9802773-aae7-44dc-
Class name	EPiServer.Templates.A EPiServer.Templates.A

Access level

	Create
niis	✓
CmsEditors	✓

Add Users/Groups

Episerver 189

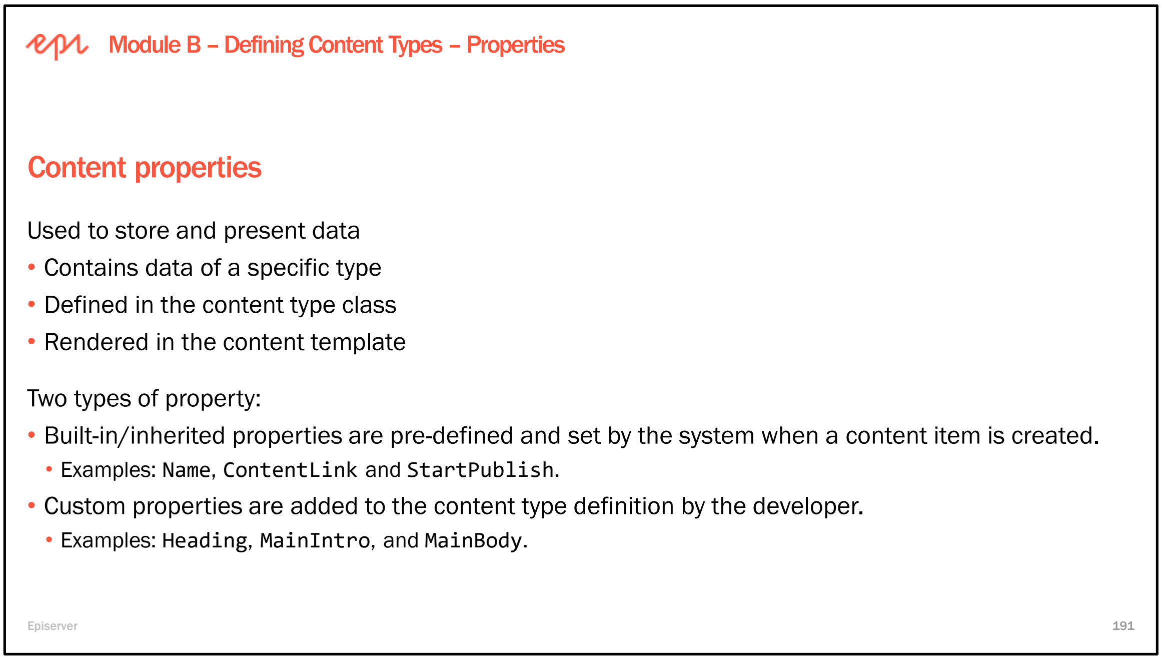
ContentType GUID

Always include a GUID on the page type (the Visual Studio Episerver template will generate one for you). The GUID is the unique ID for the page type.

When renaming a content type class a new content type will be created as long as no GUID is specified in the ContentType attribute of the class. If there is data on the old name, in other words if there are pages created with the old name, then the old page type will remain and the old pages will use the old content type. When viewing this content type in Admin under the Page Type or Block Type tab the old content type will be marked as it is missing its code. If there is no data, the old content type will be deleted.

If a GUID is specified in the ContentType attribute and the GUID matches an existing content type it will be renamed and any old data will use the renamed content type. The GUID of an existing content type is available in Admin when editing the basic information for a content type.

When migrating a solution from CMS 6 to CMS 7 or later, it is possible to take the GUID from the CMS 6 page type (found in the database) and add to the corresponding new strongly typed page type and the system will recognise it and use it for the associated pages.



Module B – Defining Content Types – Properties

Content properties

Used to store and present data

- Contains data of a specific type
- Defined in the content type class
- Rendered in the content template

Two types of property:

- Built-in/inherited properties are pre-defined and set by the system when a content item is created.
  - Examples: `Name`, `ContentLink` and `StartPublish`.
- Custom properties are added to the content type definition by the developer.
  - Examples: `Heading`, `MainIntro`, and `MainBody`.

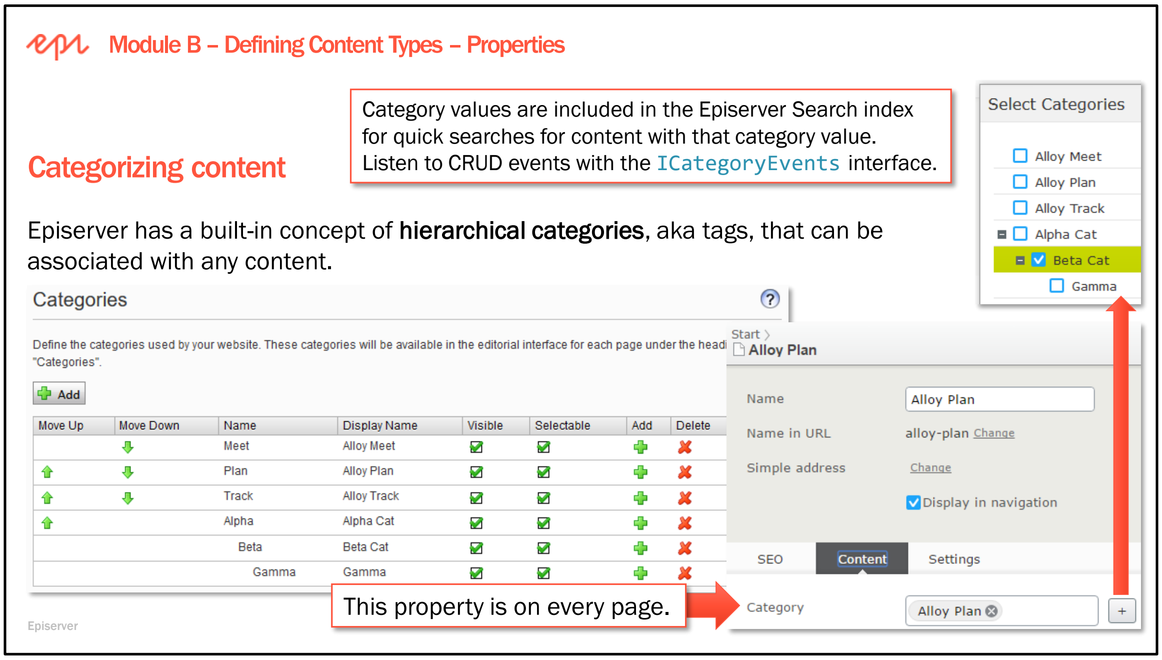
Episerver 191

A property is a part of the content item which contains data of specific types, and is used to store and present data. The type of property dictates what kind of values/content that can be entered or rendered. Properties can be added to a page type through code (or from the administrative interface). In order for a property to be rendered, it must be added to a page or block template that is linked to that particular block or page type.

Properties on a page type can also be changed or created from Admin. Properties and property settings created/updated from Admin:

- are not strongly typed
- are saved to the Episerver Database just like strongly typed properties
- overrides the strongly typed property/setting

Property	Deprecated name	Description
ParentLink		Reference to the parent of this page.
ContentTypeID	PageTypeID	The ID for the Page Type that is being used for the page.
ContentLink	PageLink	Reference to the current page.
Name	PageName	The name of the page.
ContentGuid	PageGuid	
StartPublish		Start publish date and time for the page. Nullable in CMS 10.
StopPublish		Expiry date and time for the page. Nullable in CMS 10.
VisibleInMenu		Determines if the page should be visible in menus.
LinkURL		An internal GUID-based URL to the page.
ExistingLanguages	PageLanguages	Gets or sets the existing languages for this instance.
Language	LanguageID, LanguageBranch	Gets or sets the language for this instance.



Module B – Defining Content Types – Properties

Categorizing content

Category values are included in the Episerver Search index for quick searches for content with that category value. Listen to CRUD events with the `ICategoryEvents` interface.

Episerver has a built-in concept of **hierarchical categories**, aka tags, that can be associated with any content.

Categories

Define the categories used by your website. These categories will be available in the editorial interface for each page under the headi "Categories".

Add

Move Up	Move Down	Name	Display Name	Visible	Selectable	Add	Delete
	⬇	Meet	Alloy Meet	✓	✓	+	✗
⬆	⬇	Plan	Alloy Plan	✓	✓	+	✗
⬆	⬇	Track	Alloy Track	✓	✓	+	✗
⬆		Alpha	Alpha Cat	✓	✓	+	✗
		Beta	Beta Cat	✓	✓	+	✗
		Gamma	Gamma	✓	✓	+	✗

Select Categories

- ☐ Alloy Meet
- ☐ Alloy Plan
- ☐ Alloy Track
- ☐ Alpha Cat
  - ☑ Beta Cat
    - ☐ Gamma

Start 〉 Alloy Plan

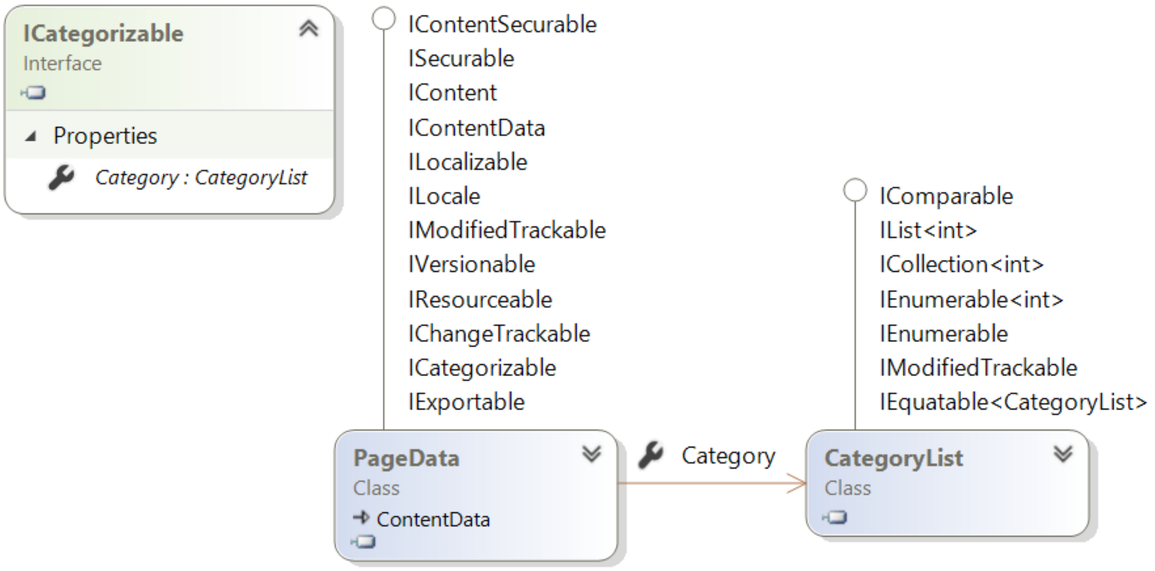
Name	Alloy Plan
Name in URL	alloy-plan Change
Simple address	Change
	☑ Display in navigation

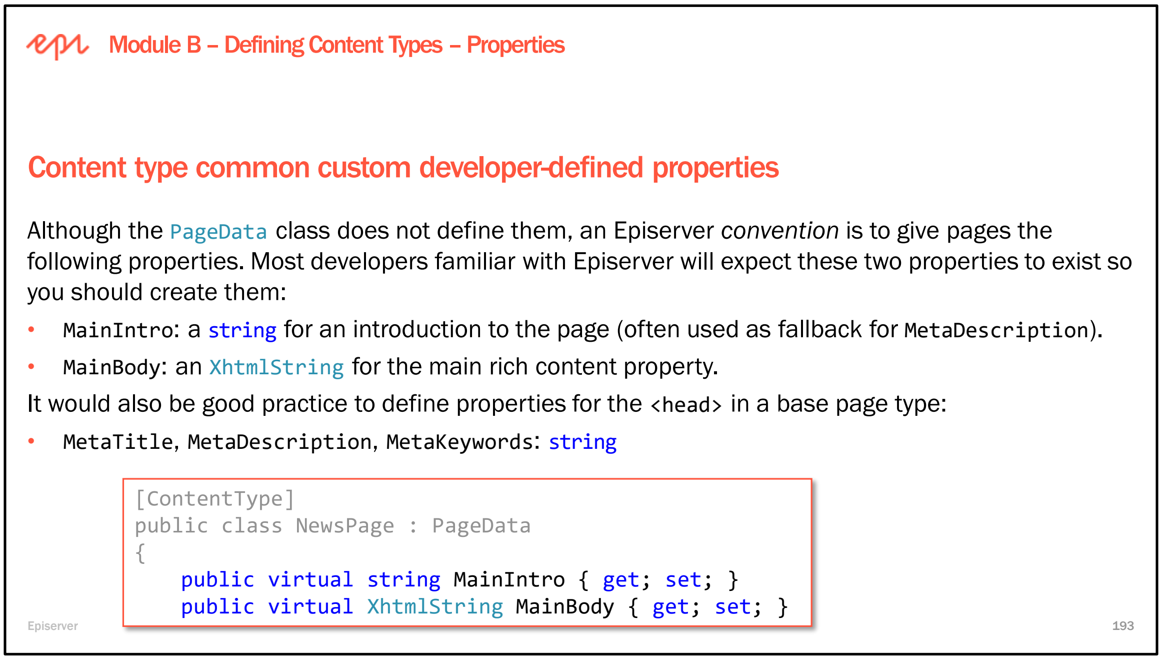
SEO | Content | Settings

This property is on every page.

Category: Alloy Plan

Episerver





Module B – Defining Content Types – Properties

Content type common custom developer-defined properties

Although the `PageData` class does not define them, an Episerver *convention* is to give pages the following properties. Most developers familiar with Episerver will expect these two properties to exist so you should create them:

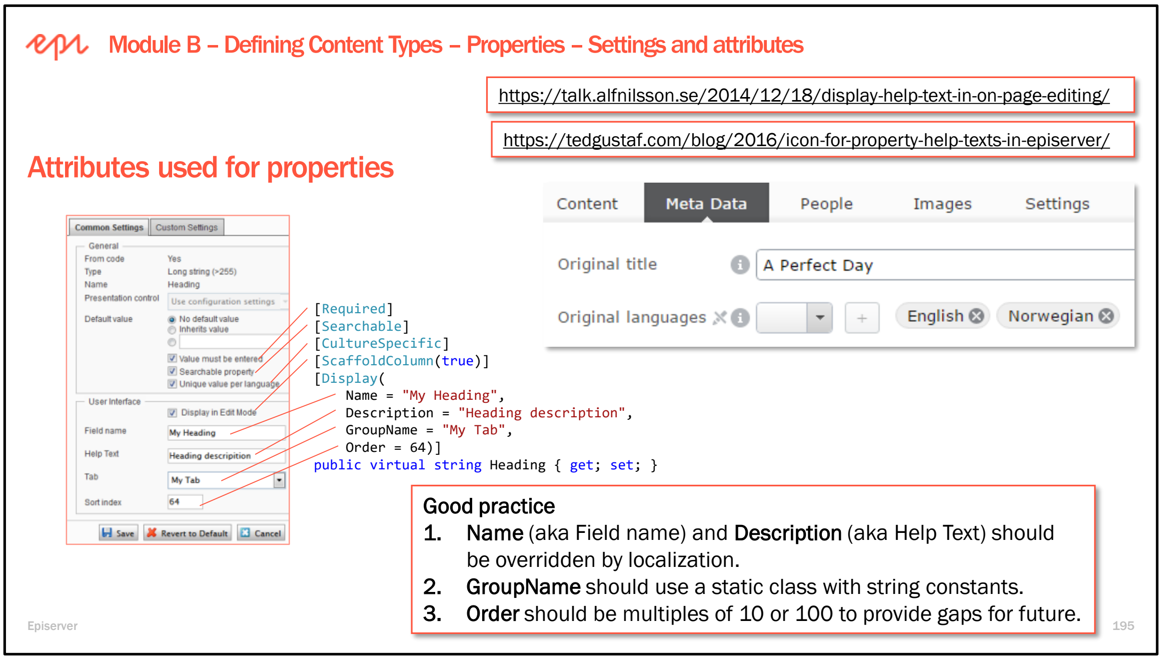
- `MainIntro`: a `string` for an introduction to the page (often used as fallback for `MetaDescription`).
- `MainBody`: an `XhtmlString` for the main rich content property.

It would also be good practice to define properties for the `<head>` in a base page type:

- `MetaTitle`, `MetaDescription`, `MetaKeywords`: `string`

```
[ContentType]
public class NewsPage : PageData
{
    public virtual string MainIntro { get; set; }
    public virtual XhtmlString MainBody { get; set; }
```

Episerver 193



Module B – Defining Content Types – Properties – Settings and attributes

https://talk.alfnilsson.se/2014/12/18/display-help-text-in-on-page-editing/

https://tedgustaf.com/blog/2016/icon-for-property-help-texts-in-episerver/

Attributes used for properties

Common Settings	Custom Settings
General	
From code	Yes
Type	Long string (>255)
Name	Heading
Presentation control	Use configuration settings
Default value	[✓ No default value] Inherits value
	☑ Value must be entered
	☑ Searchable property
	☑ Unique value per language
User Interface	
	☑ Display in Edit Mode
Field name	My Heading
Help Text	Heading descripition
Tab	My Tab
Sort index	64

Save Revert to Default Cancel

```
[Required]
[Searchable]
[CultureSpecific]
[ScaffoldColumn(true)]
[Display(
    Name = "My Heading",
    Description = "Heading description",
    GroupName = "My Tab",
    Order = 64)]
public virtual string Heading { get; set; }
```

Content	**Meta Data**	People	Images	Settings

Original title	A Perfect Day
Original languages	English, Norwegian

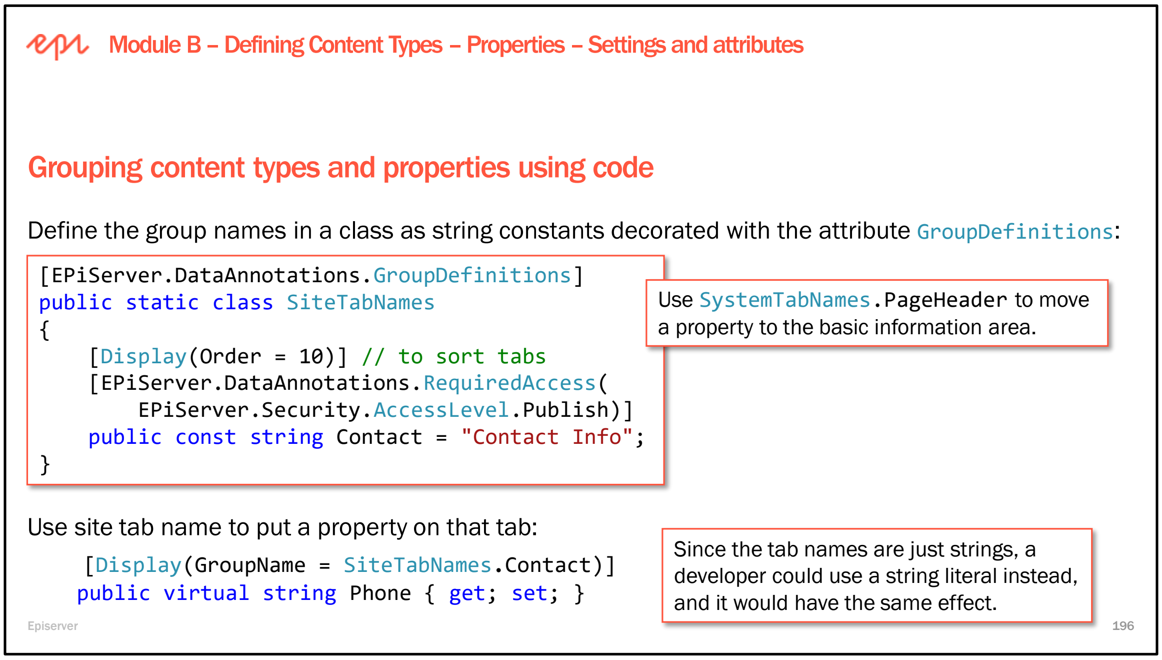
Good practice

1. **Name** (aka Field name) and **Description** (aka Help Text) should be overridden by localization.
2. **GroupName** should use a static class with string constants.
3. **Order** should be multiples of 10 or 100 to provide gaps for future.

Episerver 195

Default behavior if no attribute values are specified (attribute name in typed class in brackets):

- Name = the name of the property
- Value must be entered (Required) = false (i.e. value not required)
- Searchable property (Searchable) = true for strings, false for all other property types
- Unique value per language (CultureSpecific) = false (i.e. the property is Global by default)
- Display in Edit Mode (ScaffoldColumn) = true (i.e. visible by default)
- Field name (Display: Name) = the name of the property
- Help Text (Display: Description) is NULL
- Tab (Display: GroupName) is by default set to the Tab with the lowest sort order (which is the “Content” tab if no custom tabs have been added)
- Sort index (Display: Order) defaults to the order the properties are written in the page type class



Module B – Defining Content Types – Properties – Settings and attributes

Grouping content types and properties using code

Define the group names in a class as string constants decorated with the attribute `GroupDefinitions`:

```
[EPiServer.DataAnnotations.GroupDefinitions]
public static class SiteTabNames
{
    [Display(Order = 10)] // to sort tabs
    [EPiServer.DataAnnotations.RequiredAccess(
        EPiServer.Security.AccessLevel.Publish)]
    public const string Contact = "Contact Info";
}
```

Use `SystemTabNames.PageHeader` to move a property to the basic information area.

Use site tab name to put a property on that tab:

```
    [Display(GroupName = SiteTabNames.Contact)]
    public virtual string Phone { get; set; }
```

Since the tab names are just strings, a developer could use a string literal instead, and it would have the same effect.

Episerver 196

GroupName corresponds to the tab where that property is contained in the All Properties view of the page. Normally, these are defined as a list of constants that becomes available in the **Display** attribute.
The supporting attributes for group names such as GroupDefinitions and RequiredAccess are available from Episerver CMS version 8.
An example of when you would like to set access on a group is when you have a “Site Settings” tab on the start page (containing all the site-wide setup properties) and want to restrict it so that only a particular user group can see and edit it.

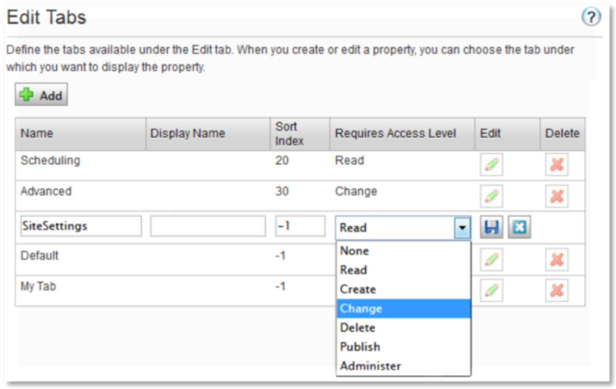
Module B – Defining Content Types – Properties – Settings and attributes

Grouping properties using Admin view

CMS Admins can create, change, or set access for the tabs from Admin view, but only for tabs *they create*. Any tabs defined in code cannot be edited or deleted.

We recommend:

- Properties that will be editable in On-Page Editing view should be on the Content tab and require Change access level.
- Properties that need higher access levels should be on a separate tab and not be editable in On-Page Editing.



Edit Tabs

Define the tabs available under the Edit tab. When you create or edit a property, you can choose the tab under which you want to display the property.

Add

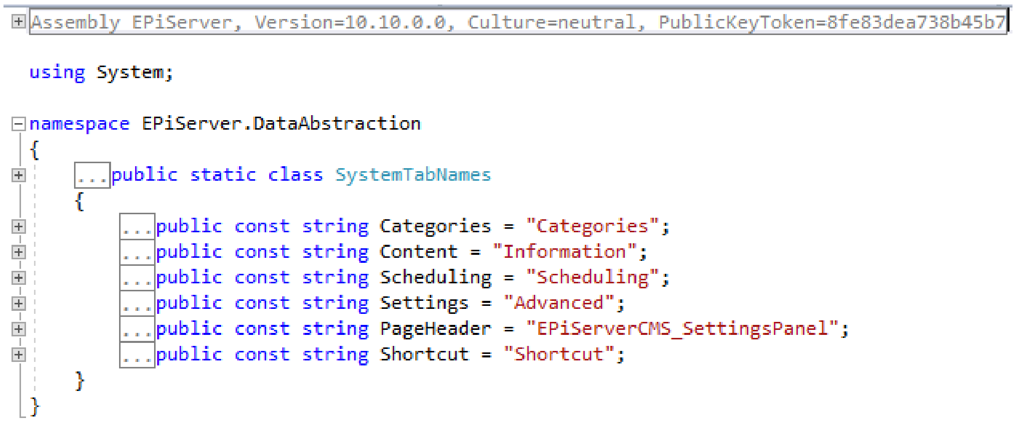
Name	Display Name	Sort Index	Requires Access Level	Edit	Delete
Scheduling		20	Read		
Advanced		30	Change		
SiteSettings		-1	Read		
Default		-1			
My Tab		-1			

Episerver 197

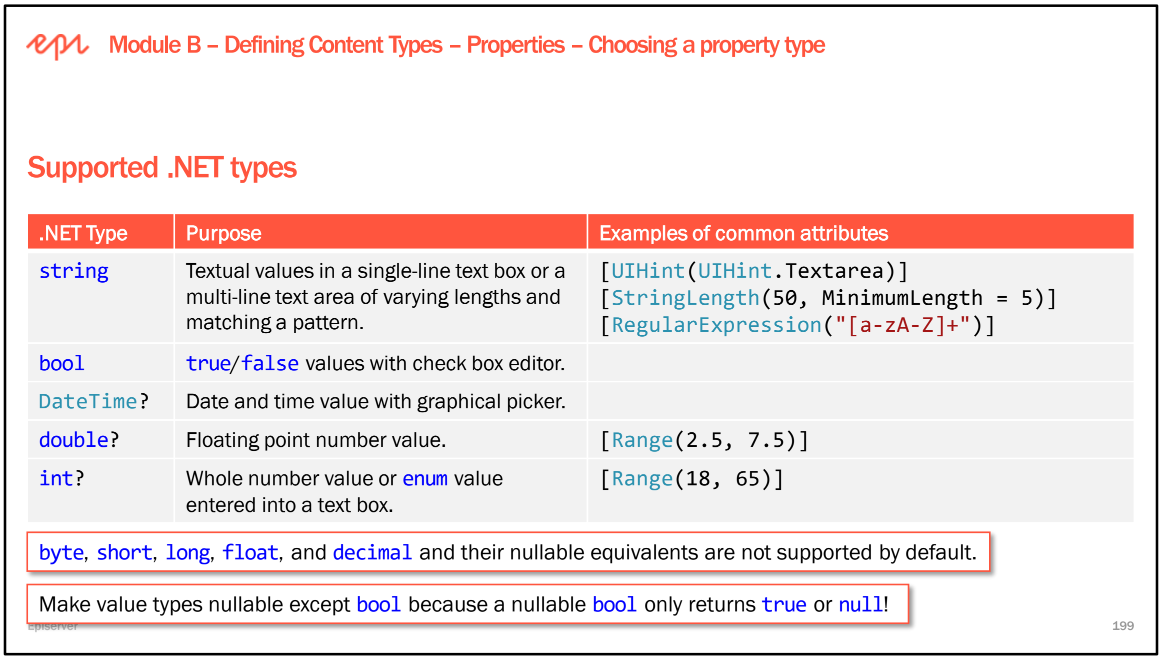
Tabs can be added from code, which is the preferred option, as discussed on the next slide.
You can however alter existing tabs and add your own tabs from Admin.

Episerver has predefined a static class with string constants for built-in tabs.

NOTE: You cannot use the administrative interface to edit groups that are defined in code.
Blog by Per Bjurström - http://world.episerver.com/blogs/Per-Bjurstrom/2015/2/typed-tabsgroups/
Episerver World - http://world.episerver.com/documentation/developer-guides/CMS/Content/grouping-content-types-and-properties/



```
Assembly EPiServer, Version=10.10.0.0, Culture=neutral, PublicKeyToken=8fe83dea738b45b7

using System;

namespace EPiServer.DataAbstraction
{
    ...public static class SystemTabNames
    {
        ...public const string Categories = "Categories";
        ...public const string Content = "Information";
        ...public const string Scheduling = "Scheduling";
        ...public const string Settings = "Advanced";
        ...public const string PageHeader = "EPiServerCMS_SettingsPanel";
        ...public const string Shortcut = "Shortcut";
    }
}
```



Module B – Defining Content Types – Properties – Choosing a property type

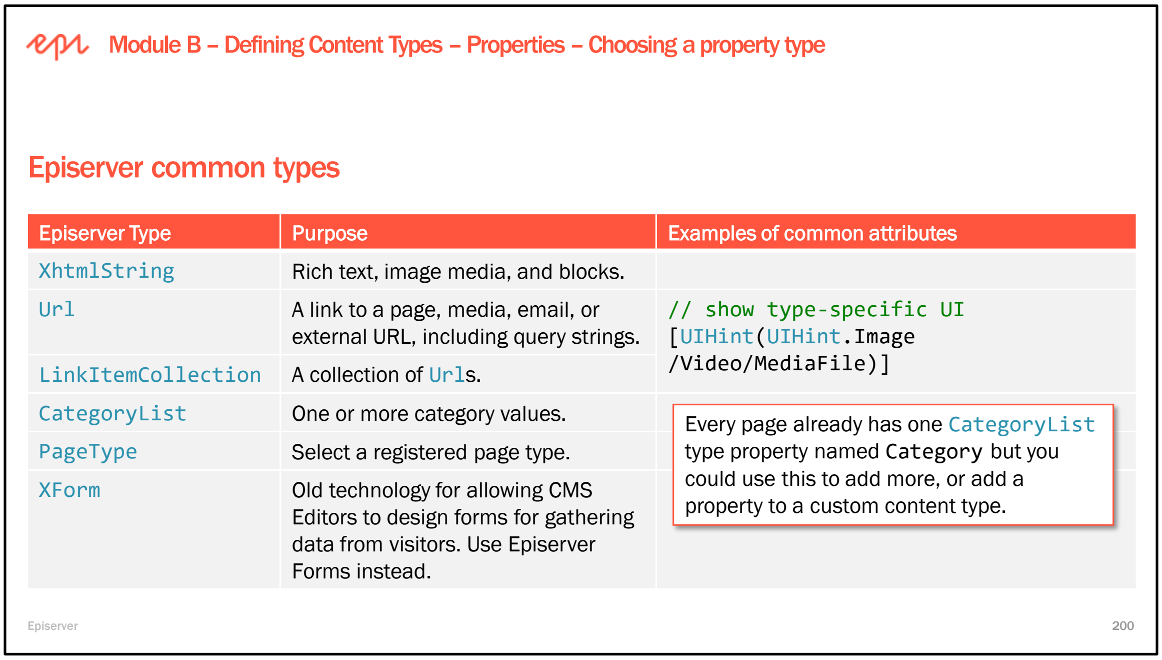
Supported .NET types

.NET Type	Purpose	Examples of common attributes
`string`	Textual values in a single-line text box or a multi-line text area of varying lengths and matching a pattern.	`[UIHint(UIHint.Textarea)]` `[StringLength(50, MinimumLength = 5)]` `[RegularExpression("[a-zA-Z]+")]`
`bool`	`true`/`false` values with check box editor.	
`DateTime?`	Date and time value with graphical picker.	
`double?`	Floating point number value.	`[Range(2.5, 7.5)]`
`int?`	Whole number value or `enum` value entered into a text box.	`[Range(18, 65)]`

`byte`, `short`, `long`, `float`, and `decimal` and their nullable equivalents are not supported by default.

Make value types nullable except `bool` because a nullable `bool` only returns `true` or `null`!

Episerver 199

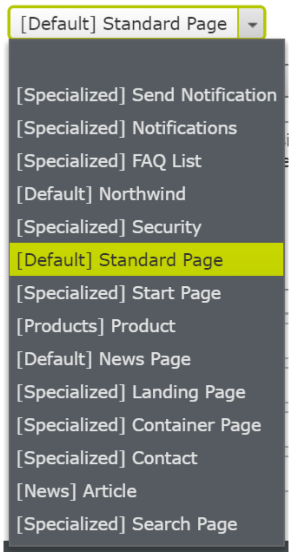


Module B – Defining Content Types – Properties – Choosing a property type

Episerver common types

Episerver Type	Purpose	Examples of common attributes
XhtmlString	Rich text, image media, and blocks.	
Url	A link to a page, media, email, or external URL, including query strings.	`// show type-specific UI` `[UIHint(UIHint.Image /Video/MediaFile)]`
LinkItemCollection	A collection of Urls.	
CategoryList	One or more category values.	Every page already has one CategoryList type property named Category but you could use this to add more, or add a property to a custom content type.
PageType	Select a registered page type.	
XForm	Old technology for allowing CMS Editors to design forms for gathering data from visitors. Use Episerver Forms instead.	

Episerver 200

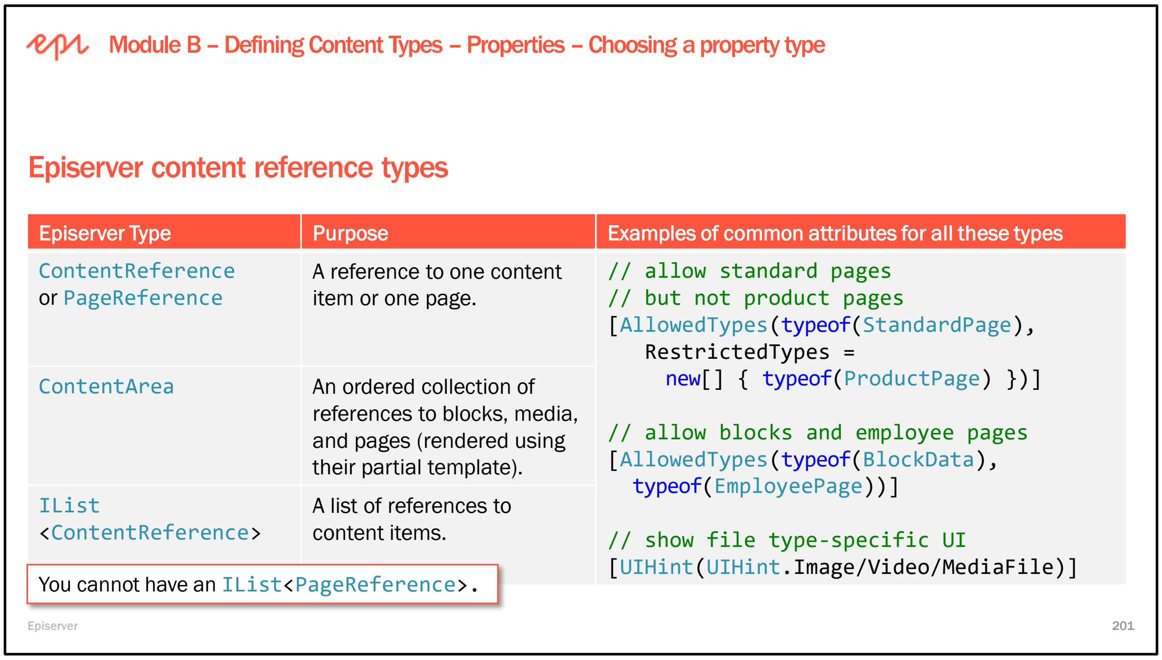


The most common property types are described in detail in the developer guide under Content > Properties.

Some examples of usage:

- Long string: Any non-HTML string properties such as Headings, introductions, shorter editorial texts without HTML.
- XHTML string (>255): Text that needs to include HTML formatting. Can contain links.
- Link collection: Links in a footer, link list in an article or blog.
- Page or content reference: For example Start node for news archive or a Contact Us page.
- URL to Image: Logotype. When you want the editor to include one image in one specific place.
- Selected/not selected: Is NULL or true, never false.
- XForms form: A “contact us” form or a simple voting function.
- PageType: if the Editor needs to choose one of the registered page types from a dropdown list.
- BLOB: Used to hold binary data (for example an image). A BLOB can be routed to with pattern <Url to content>/BlobPropertyName.

Recommendations when using string properties: In earlier versions of Episerver CMS the built in property type PropertyString (”String (<=255)”) was preferred for shorter strings (for example headings, titles and names used in menus). Upgraded sites might therefore still use it and it will still work, but the recommendation is to use PropertyLongString (”Long string (>255)”) for any non-HTML string properties. Refer to the section “Using string properties” in the Episerver CMS SDK for more information.



Module B – Defining Content Types – Properties – Choosing a property type

Episerver content reference types

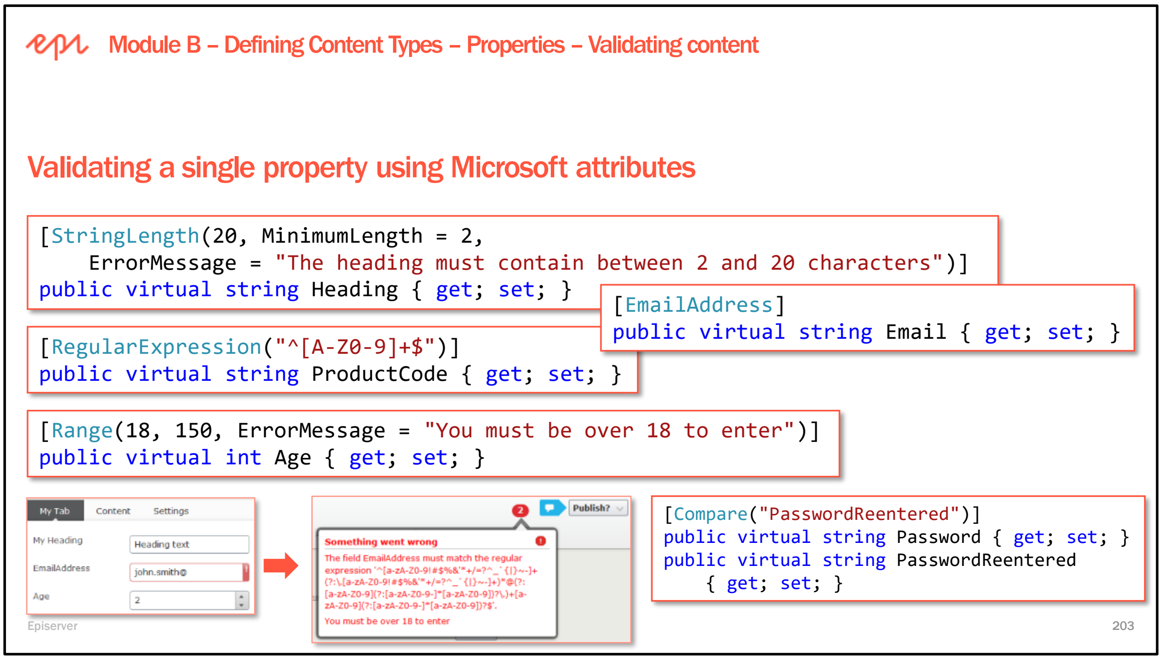
Episerver Type	Purpose	Examples of common attributes for all these types

`ContentReference` or `PageReference`	A reference to one content item or one page.	`// allow standard pages`
		`// but not product pages`
		`[AllowedTypes(typeof(StandardPage),`
		`   RestrictedTypes =`
		`     new[] { typeof(ProductPage) })]`
		
		`// allow blocks and employee pages`
		`[AllowedTypes(typeof(BlockData),`
		`  typeof(EmployeePage))]`
		
		`// show file type-specific UI`
		`[UIHint(UIHint.Image/Video/MediaFile)]`

`ContentArea`	An ordered collection of references to blocks, media, and pages (rendered using their partial template).	
`IList <ContentReference>`	A list of references to content items.	

You cannot have an `IList<PageReference>`.

Episerver 201



Module B – Defining Content Types – Properties – Validating content

Validating a single property using Microsoft attributes

```
[StringLength(20, MinimumLength = 2,
    ErrorMessage = "The heading must contain between 2 and 20 characters")]
public virtual string Heading { get; set; }
```

```
[EmailAddress]
public virtual string Email { get; set; }
```

```
[RegularExpression("^[A-Z0-9]+$")]
public virtual string ProductCode { get; set; }
```

```
[Range(18, 150, ErrorMessage = "You must be over 18 to enter")]
public virtual int Age { get; set; }
```

My Heading: Heading text
EmailAddress: john.smith@
Age: 2

Something went wrong

The field EmailAddress must match the regular expression '^[a-zA-Z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-zA-Z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-zA-Z0-9](?:[a-zA-Z0-9-]*[a-zA-Z0-9])?\.)+[a-zA-Z0-9](?:[a-zA-Z0-9-]*[a-zA-Z0-9])?$'.

You must be over 18 to enter

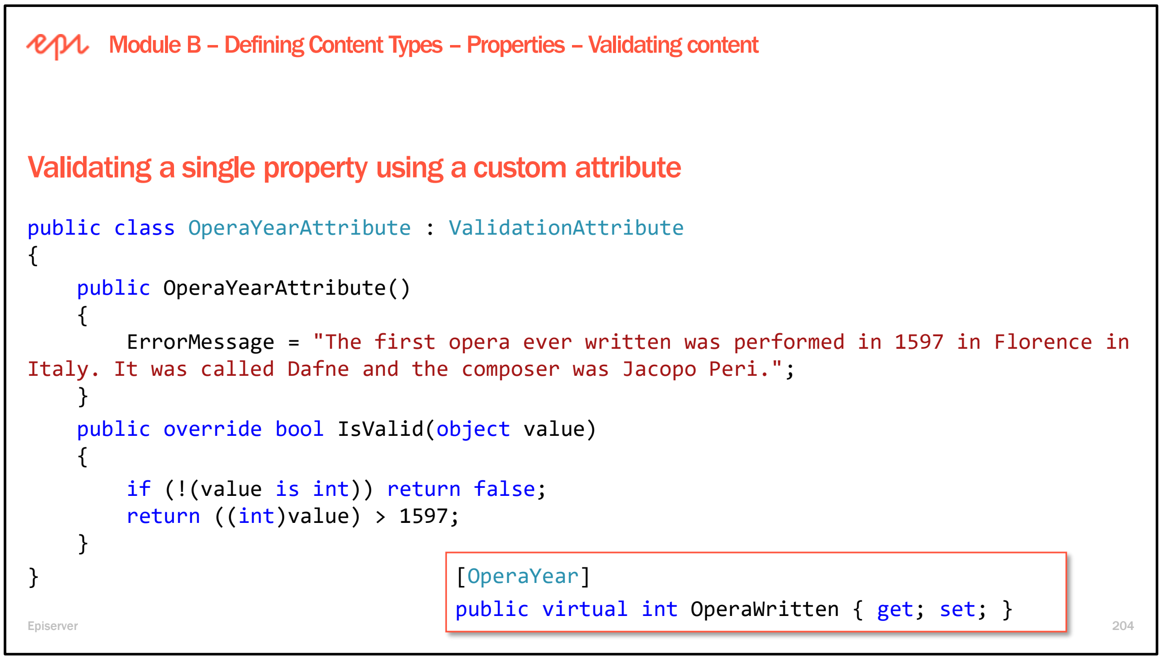
```
[Compare("PasswordReentered")]
public virtual string Password { get; set; }
public virtual string PasswordReentered
    { get; set; }
```

Episerver

203

Behavior if no attribute values are specified:

- StringLength = No length restriction
- RegularExpression = No validation of the input
- Range = numeric properties. No validation of range except the minimum/maximum values for the value type (For instance Int32.MinValue and Int32.MaxValue)



Module B – Defining Content Types – Properties – Validating content

Validating a single property using a custom attribute

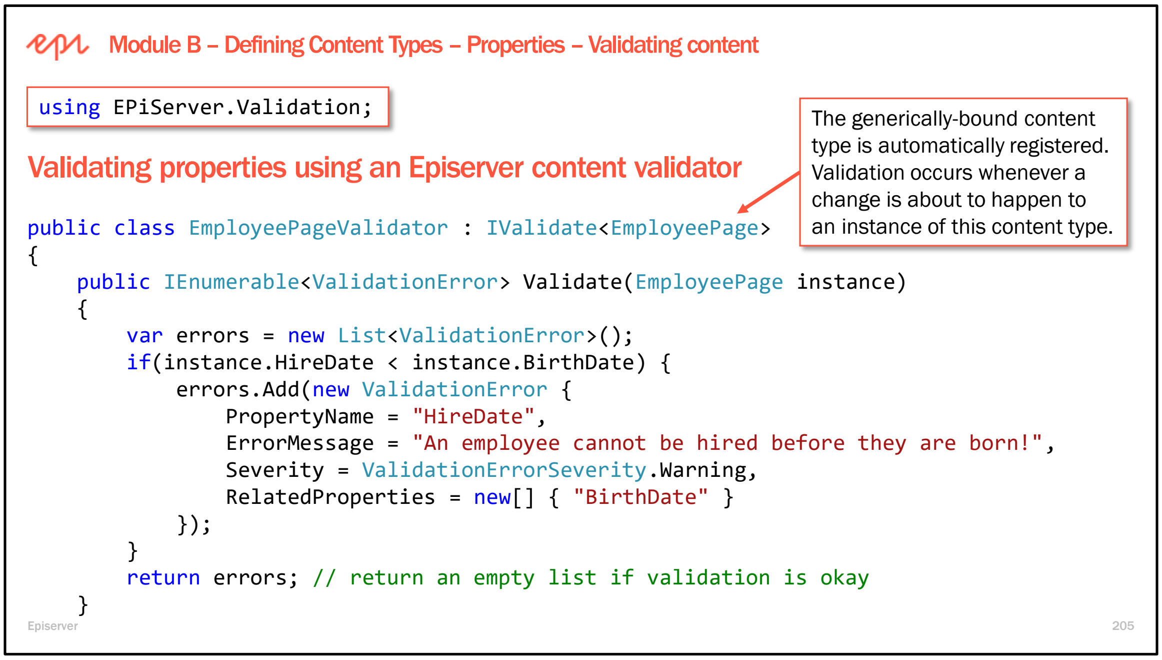
```
public class OperaYearAttribute : ValidationAttribute
{
    public OperaYearAttribute()
    {
        ErrorMessage = "The first opera ever written was performed in 1597 in Florence in
Italy. It was called Dafne and the composer was Jacopo Peri.";
    }
    public override bool IsValid(object value)
    {
        if (!(value is int)) return false;
        return ((int)value) > 1597;
    }
}
```

```
[OperaYear]
public virtual int OperaWritten { get; set; }
```

Episerver 204

The following alternative example inherits from ValidationAttribute and overrides the IsValid method that returns a ValidationResult instead of a Boolean:

```
[MyCustomValidation]
public virtual string Heading { get; set; }

public class MyCustomValidationAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        if (value.ToString().Contains("notallowed"))
        {
            return new ValidationResult("This is not allowed");
        }
        return ValidationResult.Success;
    }
}
```



Module B – Defining Content Types – Properties – Validating content

```
using EPiServer.Validation;
```

Validating properties using an Episerver content validator

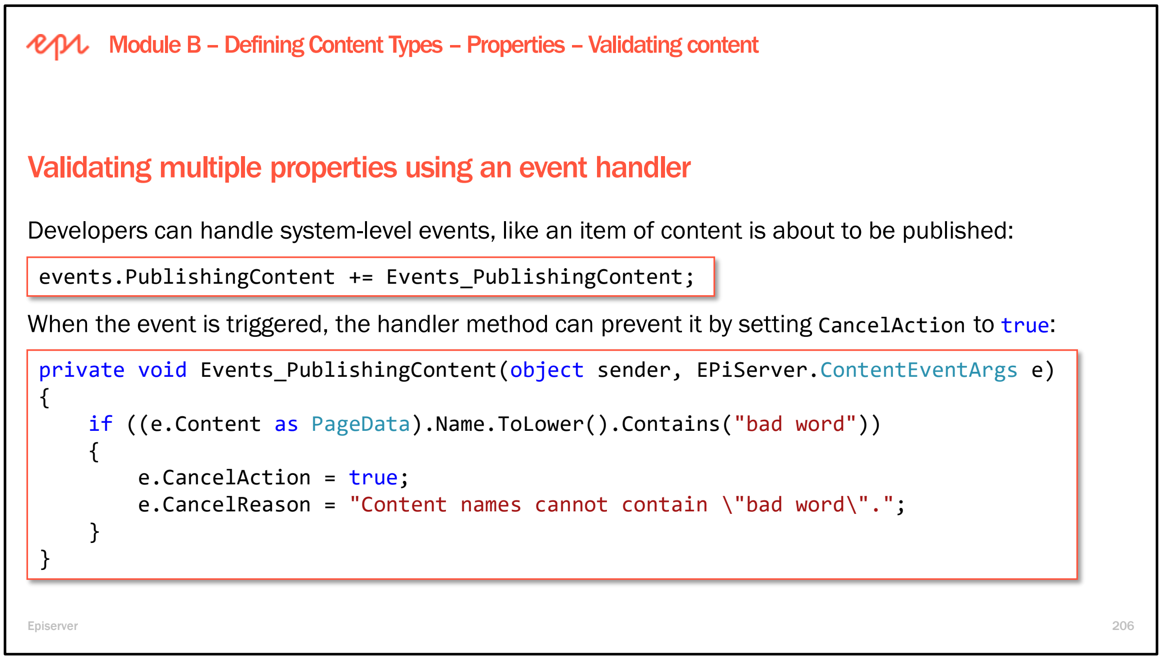
```
public class EmployeePageValidator : IValidate<EmployeePage>
{
    public IEnumerable<ValidationError> Validate(EmployeePage instance)
    {
        var errors = new List<ValidationError>();
        if(instance.HireDate < instance.BirthDate) {
            errors.Add(new ValidationError {
                PropertyName = "HireDate",
                ErrorMessage = "An employee cannot be hired before they are born!",
                Severity = ValidationErrorSeverity.Warning,
                RelatedProperties = new[] { "BirthDate" }
            });
        }
        return errors; // return an empty list if validation is okay
    }
```

The generically-bound content type is automatically registered. Validation occurs whenever a change is about to happen to an instance of this content type.

Episerver 205

If you need more complex validation, for example that the value of one property should be validated depending on the value of another property, then you can implement EPiServer.Validation.IValidate<T> (where T is the type to validate). No registration is needed, the initialization scanning will register all implementations automatically.

The validator will be called during Save for each content instance that can be assigned to T. Note however that this validation will be done on server side only.



Module B – Defining Content Types – Properties – Validating content

Validating multiple properties using an event handler

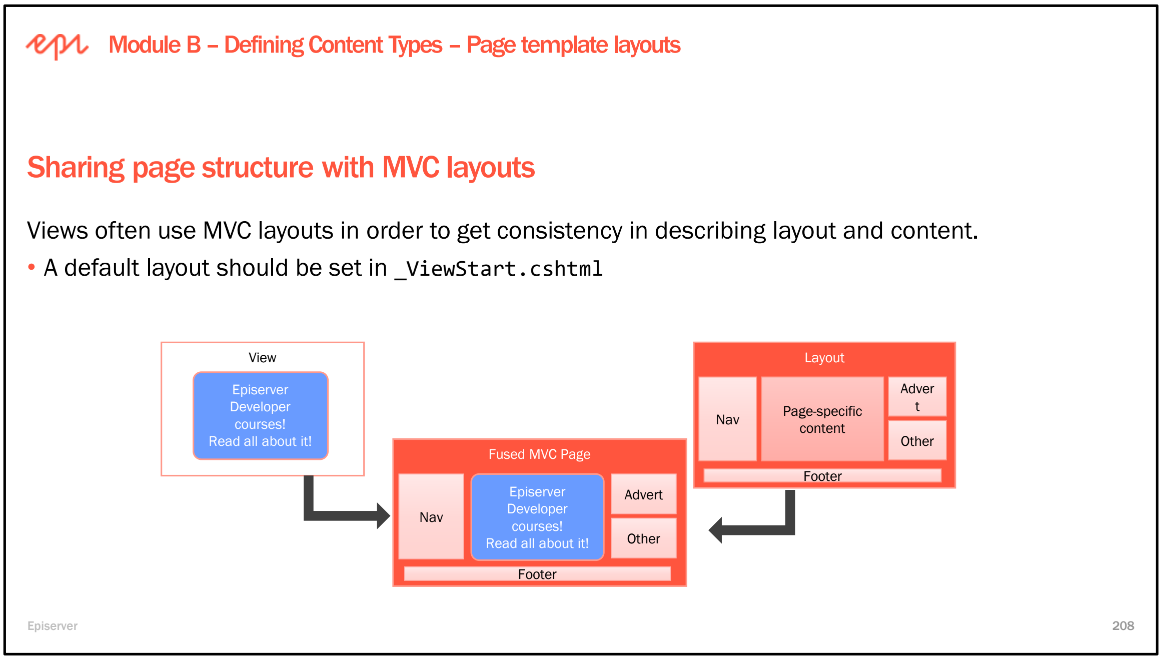
Developers can handle system-level events, like an item of content is about to be published:

```
events.PublishingContent += Events_PublishingContent;
```

When the event is triggered, the handler method can prevent it by setting `CancelAction` to `true`:

```
private void Events_PublishingContent(object sender, EPiServer.ContentEventArgs e)
{
    if ((e.Content as PageData).Name.ToLower().Contains("bad word"))
    {
        e.CancelAction = true;
        e.CancelReason = "Content names cannot contain \"bad word\".";
    }
}
```

Episerver 206

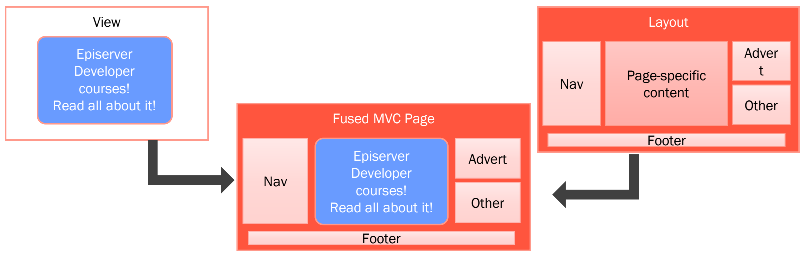


Module B – Defining Content Types – Page template layouts

Sharing page structure with MVC layouts

Views often use MVC layouts in order to get consistency in describing layout and content.

- A default layout should be set in `_ViewStart.cshtml`


View
Episerver Developer courses! Read all about it!
Fused MVC Page
Nav
Episerver Developer courses! Read all about it!
Advert
Other
Footer
Layout
Nav
Page-specific content
Adver t
Other
Footer

Episerver 208

Using layouts

MVC looks for ~/Views/_ViewStart.cshtml when a developer calls the View() method in a controller (but not when a developer calls PartialView() method). _ViewStart.cshtml can be used to set a default layout for views. If you want a specific view to use a different Layout, you can set it in the top of the specific view:

```
@{
    Layout = "~/Views/Shared/Layouts/_Root.cshtml";
}
```

Also, you can set `Layout = null;` if this page does not need a layout.

Anything in the view not wrapped in a `@section NameOfSection { }` block will go into the `@RenderBody()`

Anything in the view wrapped in a `@section NameOfSection { }` block will go into `@RenderSection(”NameOfSection”) { }`

Improving search engine optimization (SEO)

CanonicalLink extension method

- Ensure that there is only one canonical content URL from a search engine perspective.
- Canonical links will always display the primary host name (or relative on the primary host name itself).

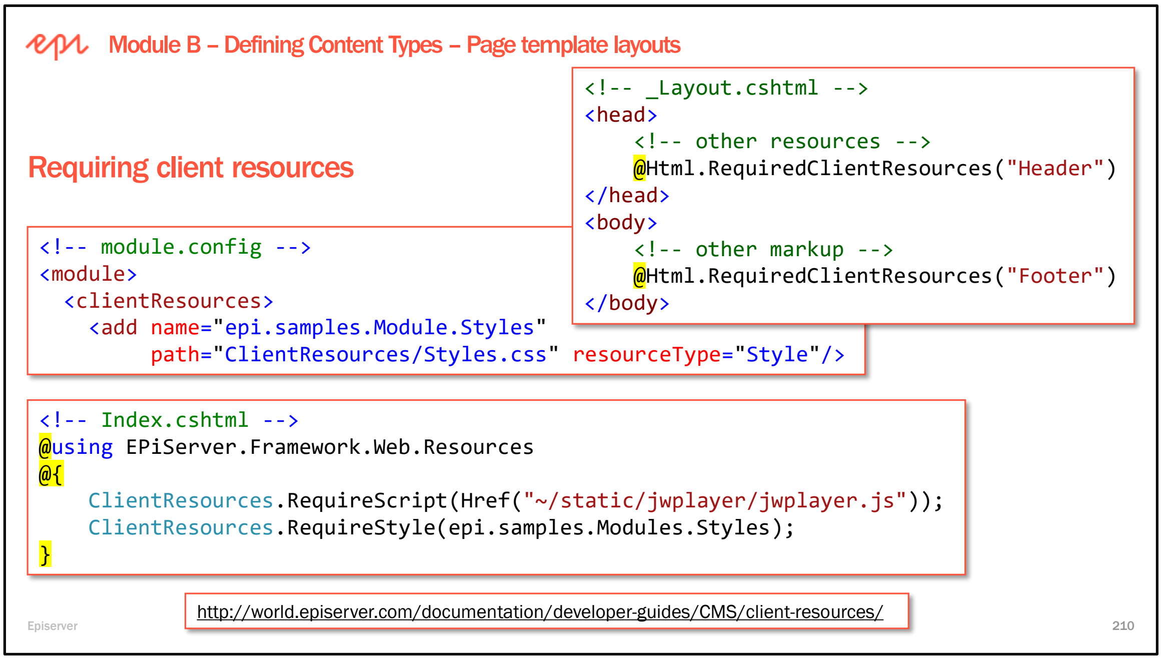
AlternateLinks extension method

- Shows alternate languages for the page.

```
    @Html.CanonicalLink()
    @Html.AlternateLinks()
</head>
```

For example, a request for the root path /, returns HTML response with following in <head>:

```
<link href="/en/" rel="canonical" />
<link href="/en/" hreflang="en" rel="alternate" />
<link href="/sv/" hreflang="sv" rel="alternate" />
```



Module B – Defining Content Types – Page template layouts

Requiring client resources

```
<!-- module.config -->
<module>
  <clientResources>
    <add name="epi.samples.Module.Styles"
         path="ClientResources/Styles.css" resourceType="Style"/>
```

```
<!-- _Layout.cshtml -->
<head>
    <!-- other resources -->
    @Html.RequiredClientResources("Header")
</head>
<body>
    <!-- other markup -->
    @Html.RequiredClientResources("Footer")
</body>
```

```
<!-- Index.cshtml -->
@using EPiServer.Framework.Web.Resources
@{
    ClientResources.RequireScript(Href("~/static/jwplayer/jwplayer.js"));
    ClientResources.RequireStyle(epi.samples.Modules.Styles);
}
```

http://world.episerver.com/documentation/developer-guides/CMS/client-resources/

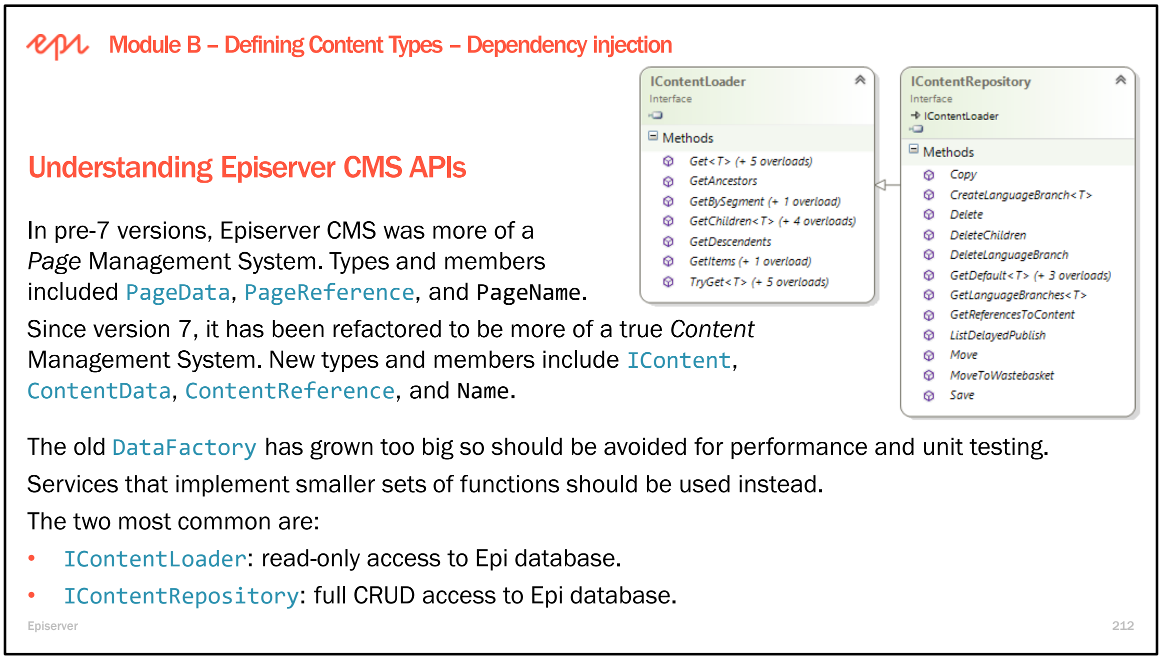
Episerver 210

When and why you should use required client resources

It is possible to require certain client resources to be rendered on the page in a specific area. Usually this approach is used when developing modules, add-ons and various plug-ins when the developers cannot access and modify the site templates to add client resources directly.

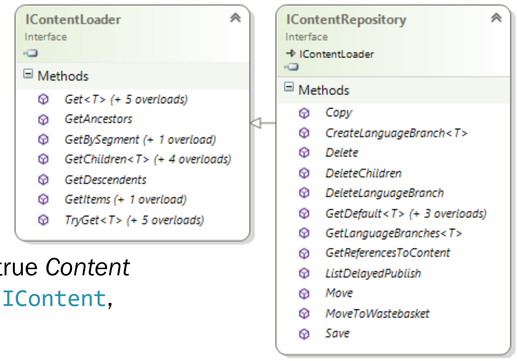
Good Practice

Any template must be able to render required client resources at least for the two default areas. Resources for the Header area should be rendered inside the <head> tag. Resources for the Footer area should be rendered in the bottom of the page, before the closing </body> tag. This is best practice to enable Episerver CMO, Live Monitor and other modules and add-ons that require script and style injections on pages.



Module B – Defining Content Types – Dependency injection

Understanding Episerver CMS APIs



In pre-7 versions, Episerver CMS was more of a *Page* Management System. Types and members included `PageData`, `PageReference`, and `PageName`.

Since version 7, it has been refactored to be more of a true *Content* Management System. New types and members include `IContent`, `ContentData`, `ContentReference`, and `Name`.

The old `DataFactory` has grown too big so should be avoided for performance and unit testing.

Services that implement smaller sets of functions should be used instead.

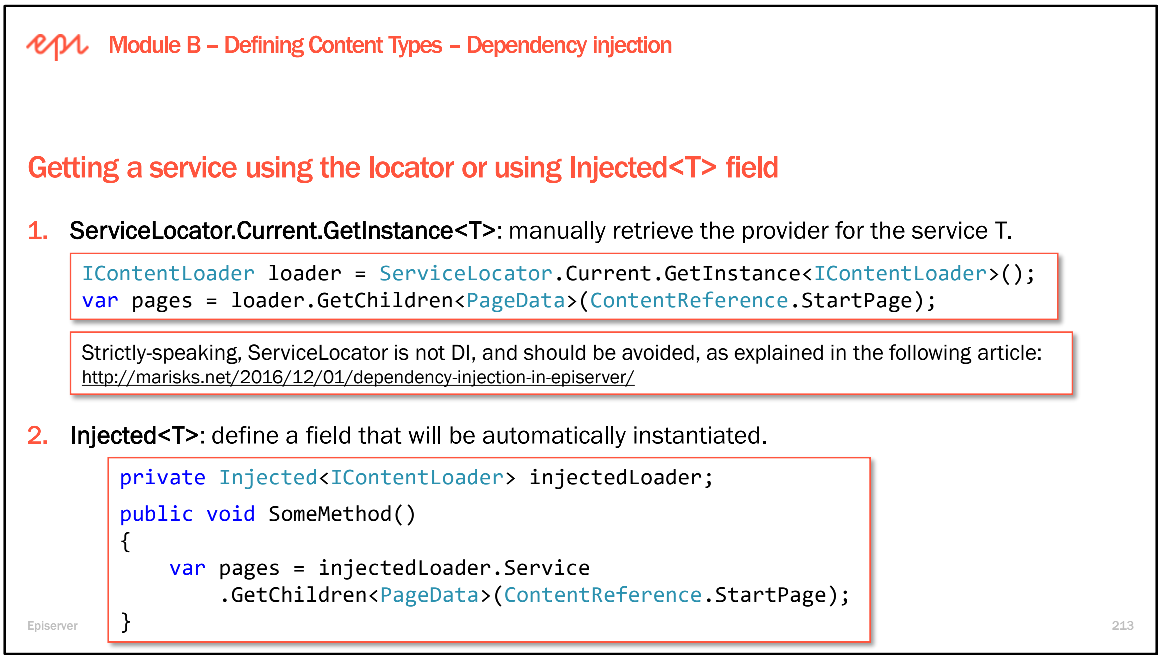
The two most common are:

- `IContentLoader`: read-only access to Epi database.
- `IContentRepository`: full CRUD access to Epi database.

Episerver 212

Other Episerver interfaces and types for services
IContentTypeRepository: CRUD with content types in Episerver database.
IContentVersionRepository: list and delete content versions in Episerver database.
IContentEvents: listen for events during CMS lifecycle, e.g. publishing a page.
IPageCriteriaQueryService: search for content in Episerver database (not indexed).
UrlResolver: convert content reference to public URL and other tasks.
LocalizationService: read localized strings from XML files (or custom provider).
DisplayOptions: allow editor to customize which template is used for a block.

Dependency injection
http://world.episerver.com/documentation/developer-guides/CMS/initialization/dependency-injection/



Module B – Defining Content Types – Dependency injection

Getting a service using the locator or using Injected<T> field

1. **ServiceLocator.Current.GetInstance<T>**: manually retrieve the provider for the service T.

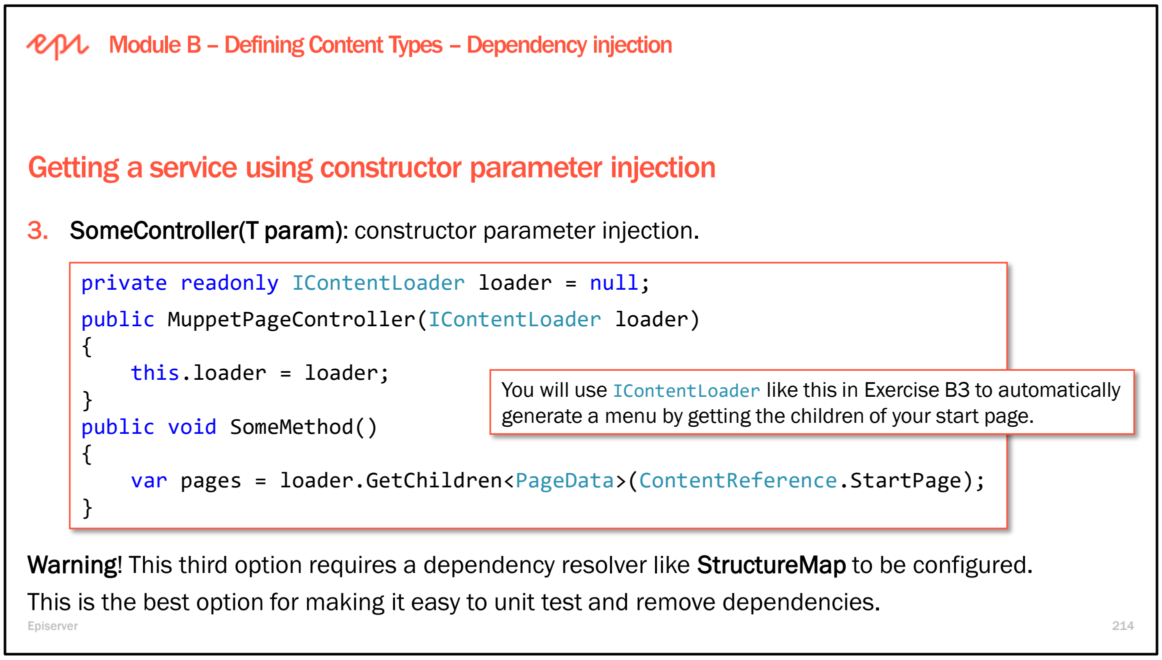
```
IContentLoader loader = ServiceLocator.Current.GetInstance<IContentLoader>();
var pages = loader.GetChildren<PageData>(ContentReference.StartPage);
```

Strictly-speaking, ServiceLocator is not DI, and should be avoided, as explained in the following article:
http://marisks.net/2016/12/01/dependency-injection-in-episerver/

2. **Injected<T>**: define a field that will be automatically instantiated.

```
private Injected<IContentLoader> injectedLoader;
public void SomeMethod()
{
    var pages = injectedLoader.Service
        .GetChildren<PageData>(ContentReference.StartPage);
}
```

Episerver 213



Module B – Defining Content Types – Dependency injection

Getting a service using constructor parameter injection

3. **SomeController(T param):** constructor parameter injection.

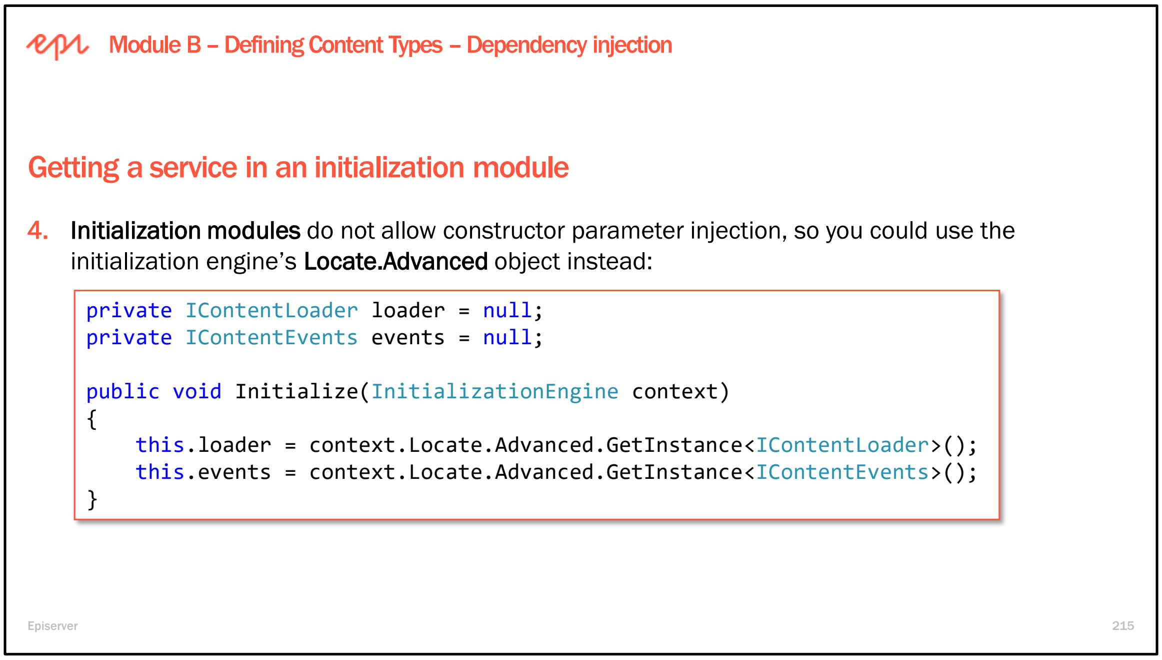
```
private readonly IContentLoader loader = null;
public MuppetPageController(IContentLoader loader)
{
    this.loader = loader;
}
public void SomeMethod()
{
    var pages = loader.GetChildren<PageData>(ContentReference.StartPage);
}
```

You will use `IContentLoader` like this in Exercise B3 to automatically generate a menu by getting the children of your start page.

Warning! This third option requires a dependency resolver like **StructureMap** to be configured.

This is the best option for making it easy to unit test and remove dependencies.

Episerver 214



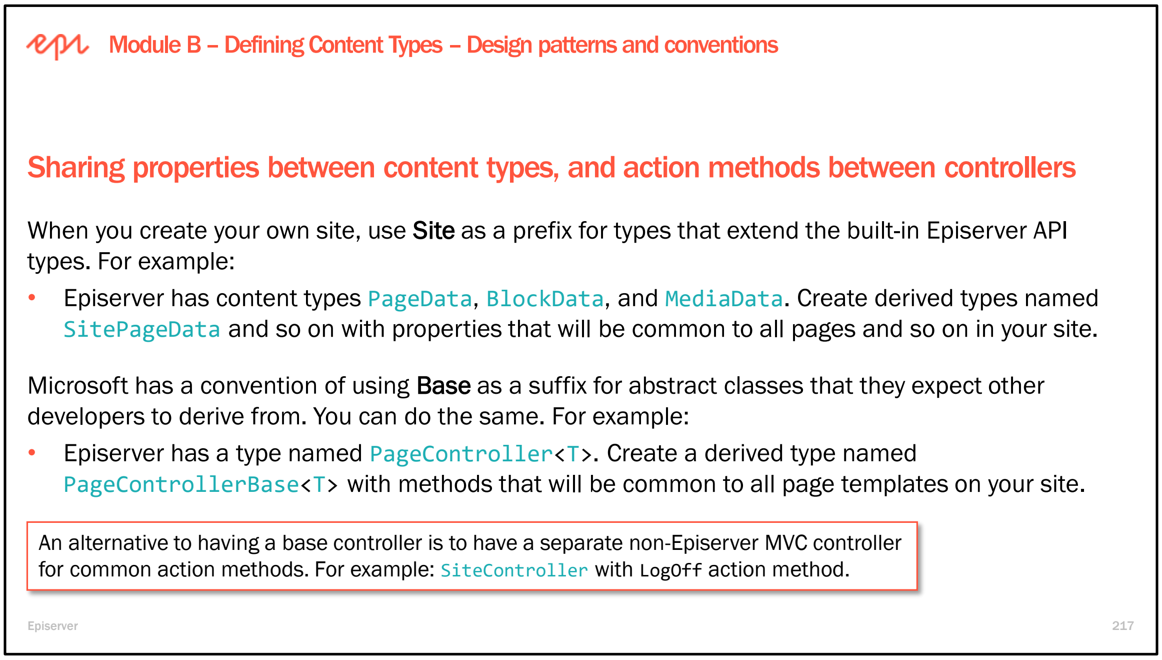
Module B – Defining Content Types – Dependency injection

Getting a service in an initialization module

4. **Initialization modules** do not allow constructor parameter injection, so you could use the initialization engine's **Locate.Advanced** object instead:

```
private IContentLoader loader = null;
private IContentEvents events = null;

public void Initialize(InitializationEngine context)
{
    this.loader = context.Locate.Advanced.GetInstance<IContentLoader>();
    this.events = context.Locate.Advanced.GetInstance<IContentEvents>();
}
```

Episerver 215



Module B – Defining Content Types – Design patterns and conventions

Sharing properties between content types, and action methods between controllers

When you create your own site, use **Site** as a prefix for types that extend the built-in Episerver API types. For example:

- Episerver has content types `PageData`, `BlockData`, and `MediaData`. Create derived types named `SitePageData` and so on with properties that will be common to all pages and so on in your site.

Microsoft has a convention of using **Base** as a suffix for abstract classes that they expect other developers to derive from. You can do the same. For example:

- Episerver has a type named `PageController<T>`. Create a derived type named `PageControllerBase<T>` with methods that will be common to all page templates on your site.

An alternative to having a base controller is to have a separate non-Episerver MVC controller for common action methods. For example: `SiteController` with `LogOff` action method.

Episerver

217

Example of a base controller for shared action methods:

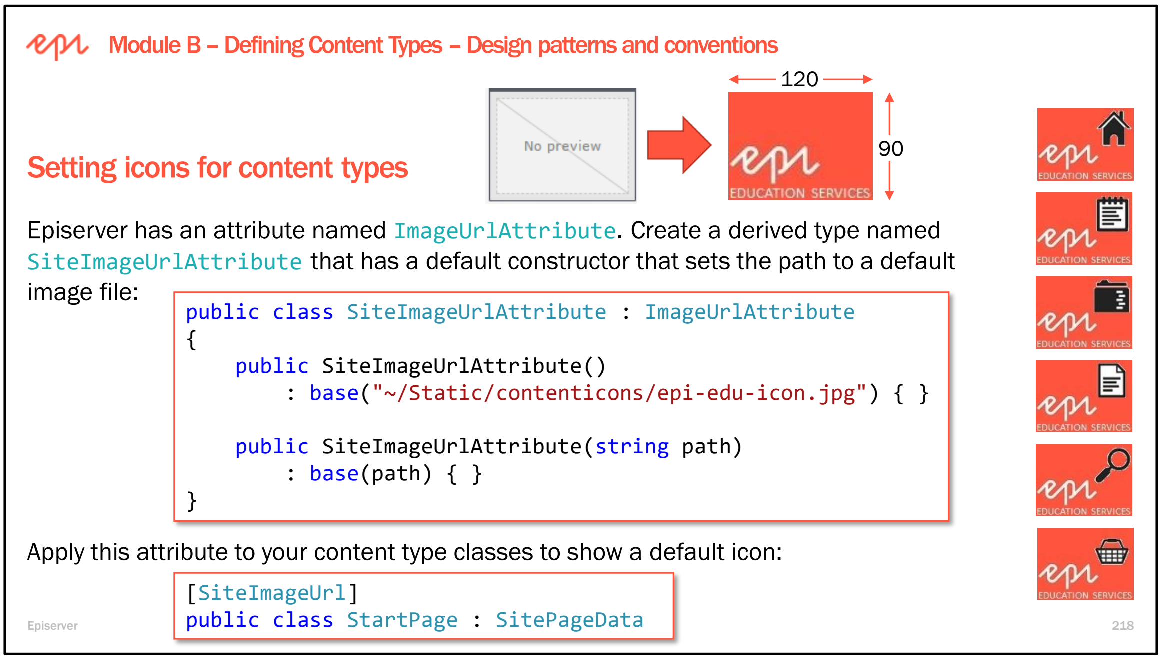
```
using AlloyTraining.Models.Pages;
using EPiServer.Web.Mvc;
using System.Web.Mvc;
using System.Web.Security;

namespace AlloyTraining.Controllers
{
    public abstract class PageControllerBase<T> : PageController<T> where T : SitePageData
    {
        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Index");
        }
    }
}
```

```
<a href="/en/about-us/news--events/logout">Log out</a>
```

Example of a separate controller for shared action methods:

```
using System.Web.Mvc;
using System.Web.Security;

namespace AlloyTraining.Controllers
{
    public class SiteController : Controller
    {
        public ActionResult Logout(string returnUrl)
        {
            FormsAuthentication.SignOut();
            return Redirect(returnUrl);
        }
    }
}
```

```
<a href="/site/logout?@Url.ContentUrl(Model.CurrentPage.ContentLink)">Log out</a>
```



Module B – Defining Content Types – Design patterns and conventions

Setting icons for content types

No preview
120
90
EDUCATION SERVICES

Episerver has an attribute named `ImageUrlAttribute`. Create a derived type named `SiteImageUrlAttribute` that has a default constructor that sets the path to a default image file:

```
public class SiteImageUrlAttribute : ImageUrlAttribute
{
    public SiteImageUrlAttribute()
        : base("~/Static/contenticons/epi-edu-icon.jpg") { }

    public SiteImageUrlAttribute(string path)
        : base(path) { }
}
```

Apply this attribute to your content type classes to show a default icon:

```
[SiteImageUrl]
public class StartPage : SitePageData
```

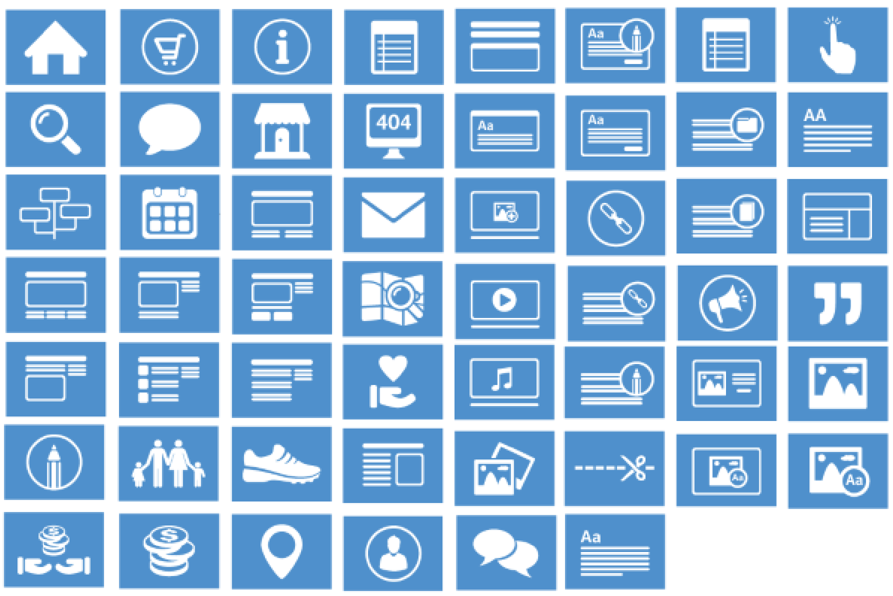
Episerver

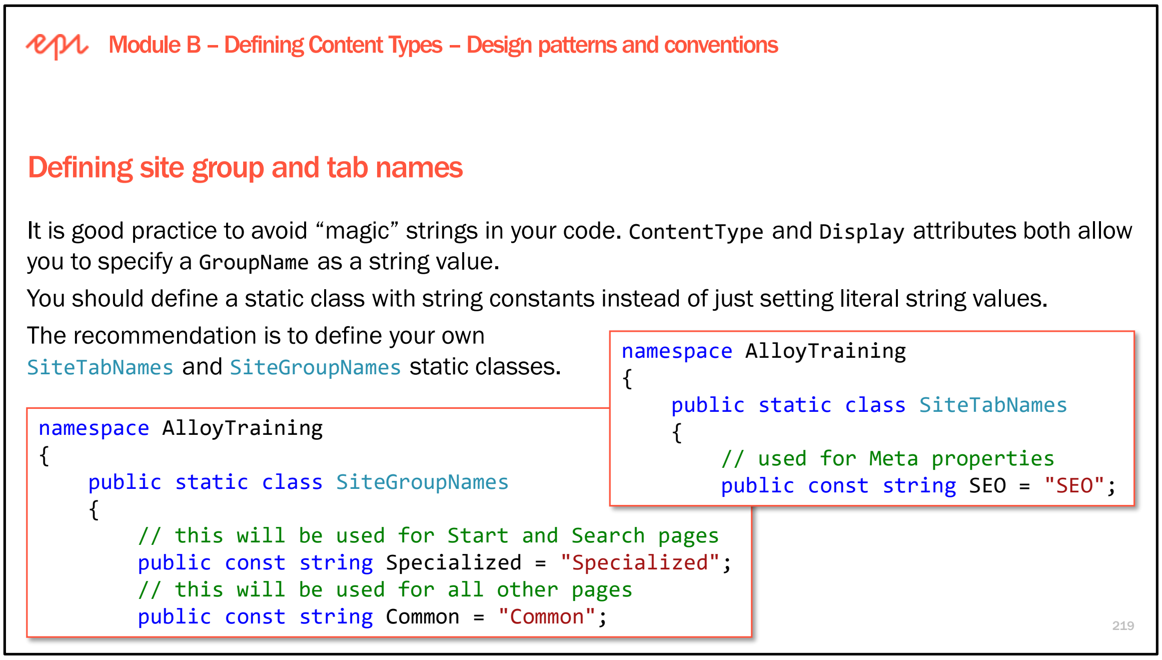
218

CONTENT ICONS FOR EPISERVER

A collection of 87 icons for EPiServer 7+ CMS content types.

https://www.markeverard.com/2014/11/17/content-icons-for-episerver/





Module B – Defining Content Types – Design patterns and conventions

Defining site group and tab names

It is good practice to avoid “magic” strings in your code. `ContentType` and `Display` attributes both allow you to specify a `GroupName` as a string value.

You should define a static class with string constants instead of just setting literal string values.

The recommendation is to define your own `SiteTabNames` and `SiteGroupNames` static classes.

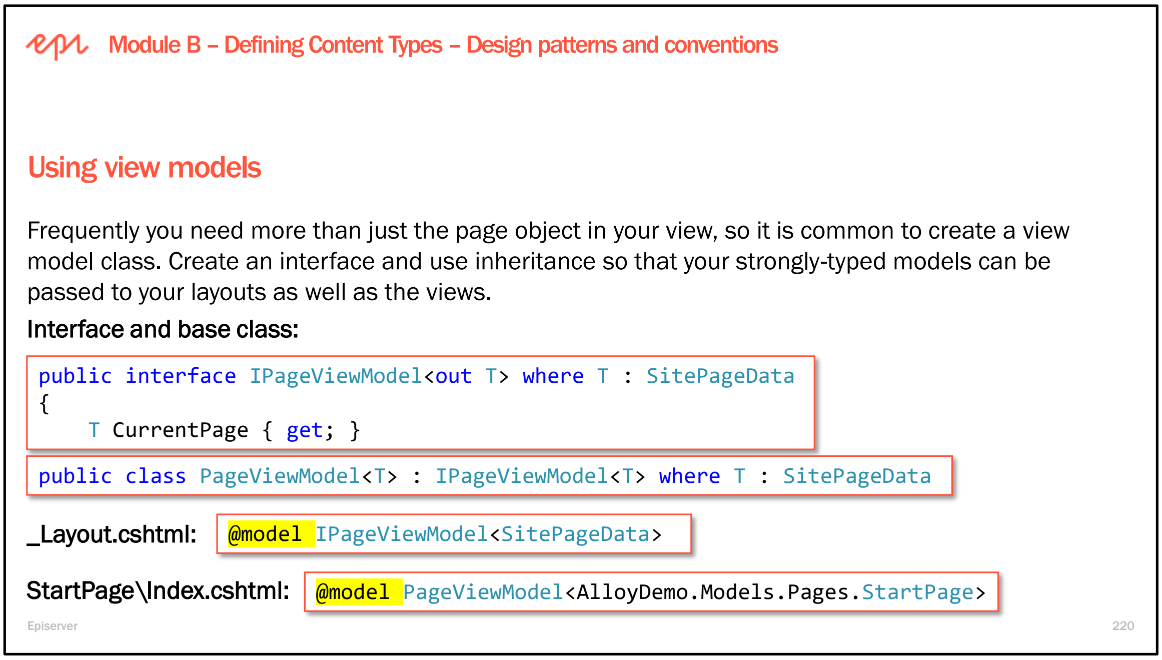
```
namespace AlloyTraining
{
    public static class SiteTabNames
    {
        // used for Meta properties
        public const string SEO = "SEO";
```

```
namespace AlloyTraining
{
    public static class SiteGroupNames
    {
        // this will be used for Start and Search pages
        public const string Specialized = "Specialized";
        // this will be used for all other pages
        public const string Common = "Common";
```

219

```
namespace AlloyTraining
{
    public static class SiteGroupNames
    {
        // this will be used for Start and Search pages
        public const string Specialized = "Specialized";

        // this will be used for all other pages
        public const string Common = "Common";

        // this will be used for News Landing and Article pages
        public const string News = "News";
    }
}
```

```
using EPiServer.DataAnnotations;
using EPiServer.Security;
using System.ComponentModel.DataAnnotations;

namespace AlloyTraining
{
    [GroupDefinitions]
    public static class SiteTabNames
    {
        [Display(Order = 10)]
        [RequiredAccess(AccessLevel.Edit)]
        public const string SEO = "SEO";

        [Display(Order = 20)]
        [RequiredAccess(AccessLevel.Administer)]
        public const string SiteSettings = "Site Settings";
    }
}
```



Module B – Defining Content Types – Design patterns and conventions

Using view models

Frequently you need more than just the page object in your view, so it is common to create a view model class. Create an interface and use inheritance so that your strongly-typed models can be passed to your layouts as well as the views.

Interface and base class:

```
public interface IPageViewModel<out T> where T : SitePageData
{
    T CurrentPage { get; }
```

```
public class PageViewModel<T> : IPageViewModel<T> where T : SitePageData
```

_Layout.cshtml:
```
@model IPageViewModel<SitePageData>
```

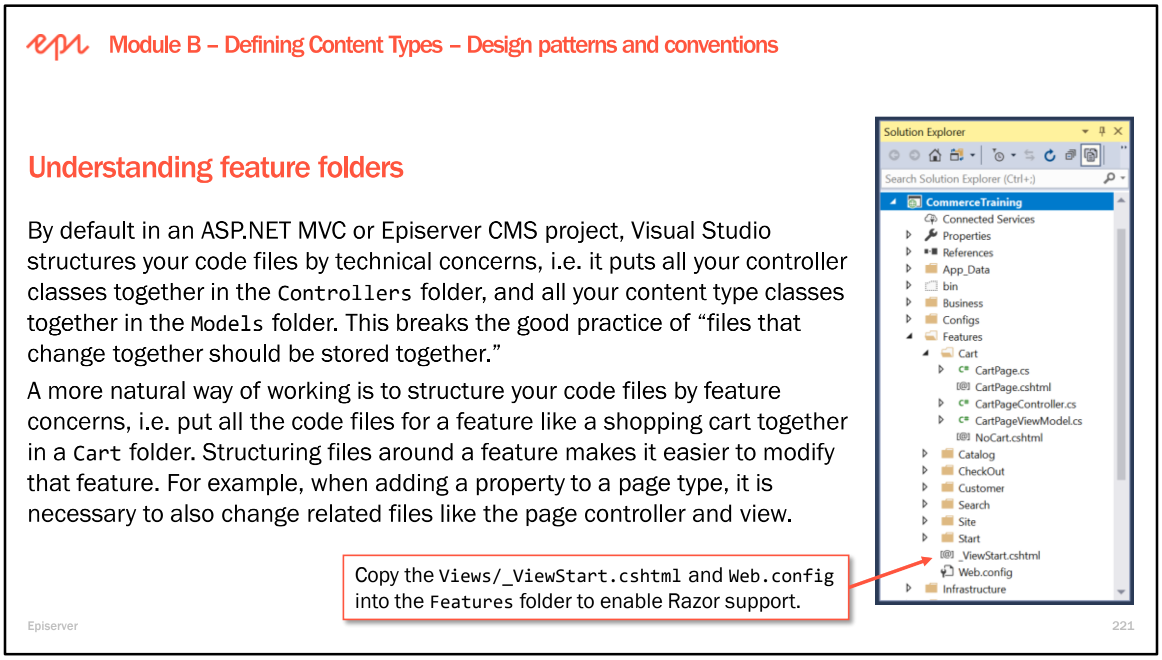
StartPage\Index.cshtml:
```
@model PageViewModel<AlloyDemo.Models.Pages.StartPage>
```

Episerver 220

An alternative to having a view model is to decorate properties with the `[Ignore]` attribute. This prevents them from being stored in the CMS. But beware of caching!

http://blog.q1.se/2016/03/08/rule-of-thumb-never-have-ignore-properties-in-a-contenttypemodel/

More information:
To use View Model or not to use View Model:
http://joelabrahamsson.com/episerver-and-mvc-what-is-the-view-model/

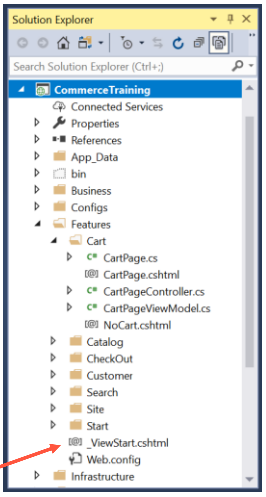


Module B – Defining Content Types – Design patterns and conventions

Understanding feature folders

By default in an ASP.NET MVC or Episerver CMS project, Visual Studio structures your code files by technical concerns, i.e. it puts all your controller classes together in the `Controllers` folder, and all your content type classes together in the `Models` folder. This breaks the good practice of “files that change together should be stored together.”

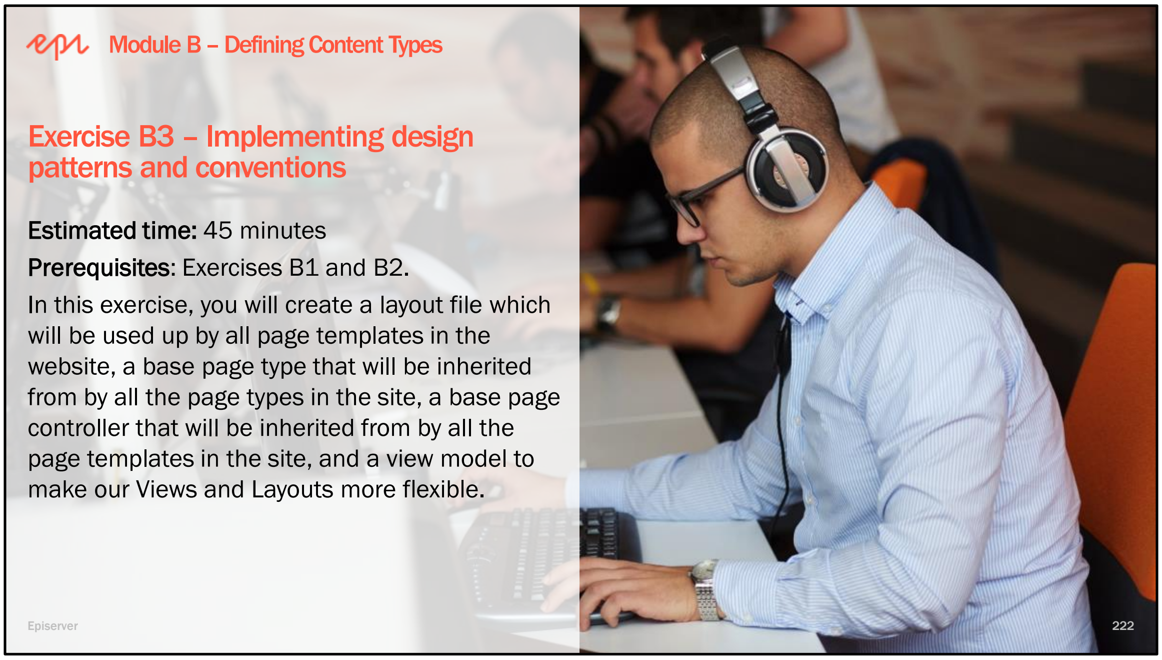
A more natural way of working is to structure your code files by feature concerns, i.e. put all the code files for a feature like a shopping cart together in a `Cart` folder. Structuring files around a feature makes it easier to modify that feature. For example, when adding a property to a page type, it is necessary to also change related files like the page controller and view.



Copy the `Views/_ViewStart.cshtml` and `Web.config` into the `Features` folder to enable Razor support.

Episerver 221

“Feature Folders” structure in ASP.NET MVC
http://haselt.com/feature-folders-structure-in-asp-net-mvc/
http://kurtdowswell.com/software-development/asp-net-core-mvc-feature-folders/



Module B – Defining Content Types

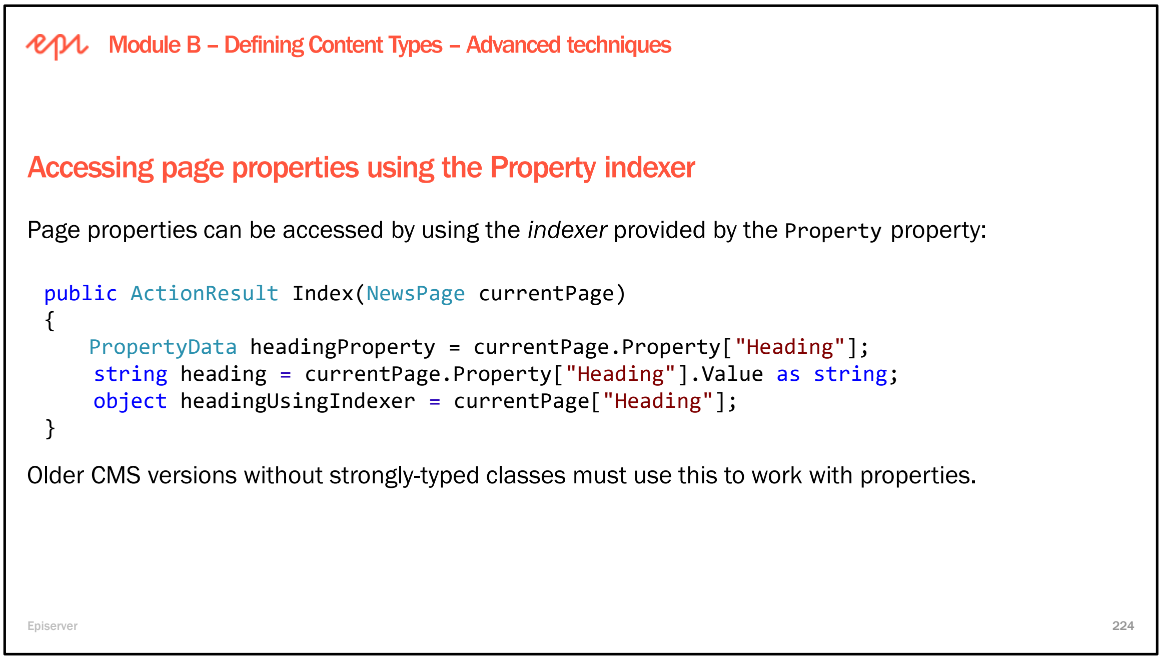
Exercise B3 – Implementing design patterns and conventions

Estimated time: 45 minutes

Prerequisites: Exercises B1 and B2.

In this exercise, you will create a layout file which will be used up by all page templates in the website, a base page type that will be inherited from by all the page types in the site, a base page controller that will be inherited from by all the page templates in the site, and a view model to make our Views and Layouts more flexible.

Episerver 222



Module B – Defining Content Types – Advanced techniques

Accessing page properties using the Property indexer

Page properties can be accessed by using the *indexer* provided by the `Property` property:

```
public ActionResult Index(NewsPage currentPage)
{
    PropertyData headingProperty = currentPage.Property["Heading"];
     string heading = currentPage.Property["Heading"].Value as string;
     object headingUsingIndexer = currentPage["Heading"];
}
```

Older CMS versions without strongly-typed classes must use this to work with properties.

Episerver 224

If the property is not found, the indexer returns `null`.

Looping through the properties in the `currentPage.Property` dictionary will only return the page-specific properties, not properties in shared blocks on the page.

`EPiServer.Core` has an extension method `GetPropertyValue()` that is beneficial to use when working with non-strongly-typed content. The method and its overloads handle null checks and type validation

Example: currentPage.GetPropertyValue(“MyProperty”)

Examples with fallback

If a string must be returned use a fallback value as follows:

- string s = currentPage.GetPropertyValue("StringProperty“, string.Empty);

Getting typed values:

- DateTime date = currentPage.GetPropertyValue<DateTime>("DateProperty", DateTime.Now);
- int i = currentPage.GetPropertyValue<int>("IntegerProperty", 0);
- bool b = currentPage.GetPropertyValue<bool>("B", false);

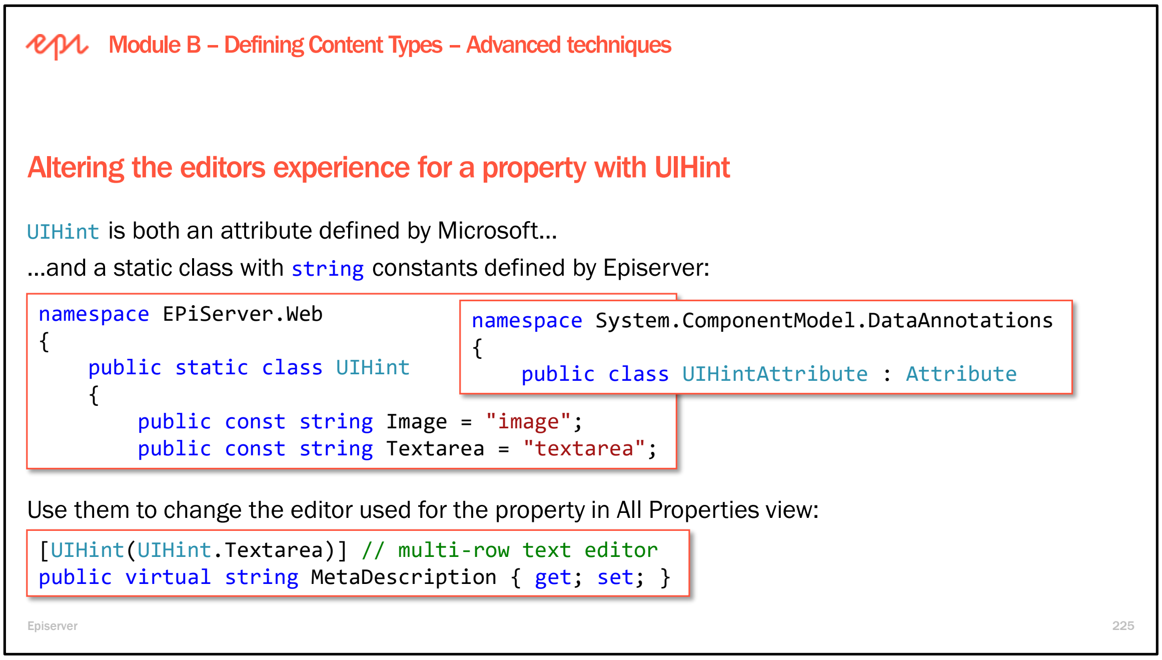
Another fallback in the markup using the ?? operator:

- currentPage["Heading"] ?? currentPage.Page

Using a fallback value in an overload of the GetPropertyValue extension method:

- string pageHeading = currentPage.GetPropertyValue("PageHeading", currentPage.Name);

More information
Magnus Rahl has written a blog post on Episerver World about the property return types and how they have changed and how to work with non-strongly-typed property values: http://world.episerver.com/Blogs/Magnus-Rahl/Dates/2012/10/Upgrading-vs-property-return-type-changes-in-Episerver-7/



Module B – Defining Content Types – Advanced techniques

Altering the editors experience for a property with UIHint

UIHint is both an attribute defined by Microsoft…
…and a static class with string constants defined by Episerver:

```
namespace EPiServer.Web
{
    public static class UIHint
    {
        public const string Image = "image";
        public const string Textarea = "textarea";
```

```
namespace System.ComponentModel.DataAnnotations
{
    public class UIHintAttribute : Attribute
```

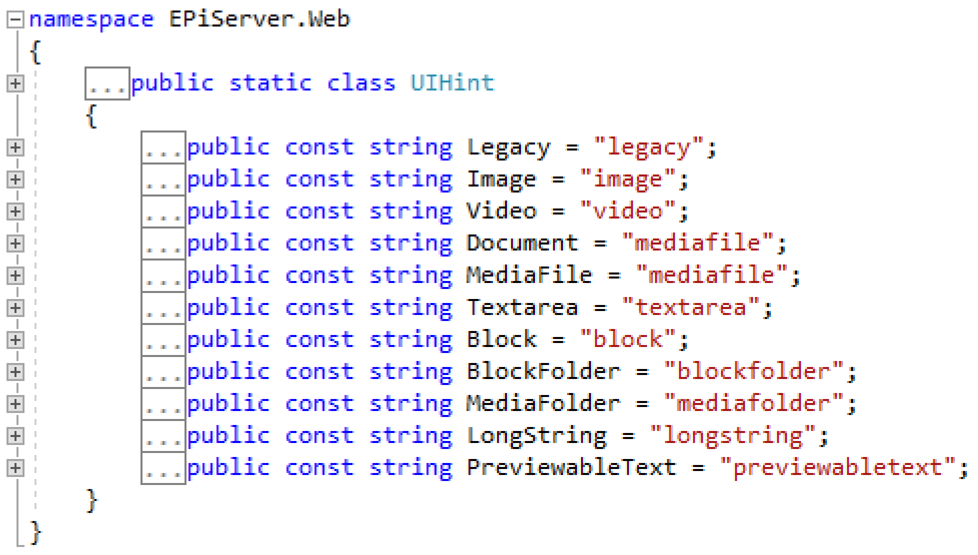
Use them to change the editor used for the property in All Properties view:

```
[UIHint(UIHint.Textarea)] // multi-row text editor
public virtual string MetaDescription { get; set; }
```

Episerver 225

UIHint is used to select either editor/renderer or both by defining a hint string. You can use the enumeration `EPiServer.Web.UIHint` to use hints for known types in the system, for instance `UIHint.Image`.

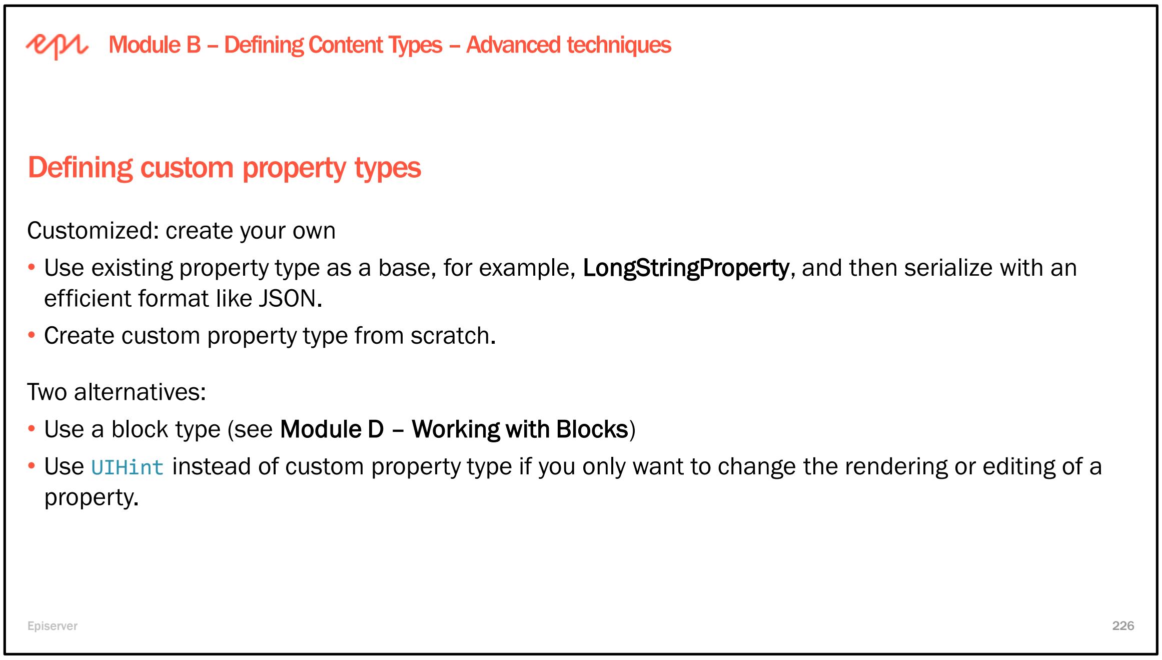
Episerver CMS – Advanced Development training course covers how to define your own `SiteUIHint` enum with custom editors.



```
namespace EPiServer.Web
{
    ...public static class UIHint
    {
        ...public const string Legacy = "legacy";
        ...public const string Image = "image";
        ...public const string Video = "video";
        ...public const string Document = "mediafile";
        ...public const string MediaFile = "mediafile";
        ...public const string Textarea = "textarea";
        ...public const string Block = "block";
        ...public const string BlockFolder = "blockfolder";
        ...public const string MediaFolder = "mediafolder";
        ...public const string LongString = "longstring";
        ...public const string PreviewableText = "previewabletext";
    }
}
```

`UIHint.BlockFolder` and `UIHint.MediaFolder` are deprecated in CMS 11. Use `UIHint.AssetsFolder` instead.

`UIHint.LongString` is deprecated in CMS 11. Use `UIHint.Textarea` instead.



Module B – Defining Content Types – Advanced techniques

Defining custom property types

Customized: create your own

- Use existing property type as a base, for example, **LongStringProperty**, and then serialize with an efficient format like JSON.
- Create custom property type from scratch.

Two alternatives:

- Use a block type (see **Module D – Working with Blocks**)
- Use `UIHint` instead of custom property type if you only want to change the rendering or editing of a property.

Episerver 226

Episerver CMS provides many built-in data types for properties. It is also possible to create your own customized property types.
Customized property types can be implemented in the following ways:

- Use an existing property type as a base and change its behavior
- Create a custom property type from scratch

More information:
Validating property values, change rendering and change editing: http://world.episerver.com/Blogs/Linus-Ekstrom/Dates/2012/12/Changes-for-properties-between-Episerver-6-and-7/
Advanced:

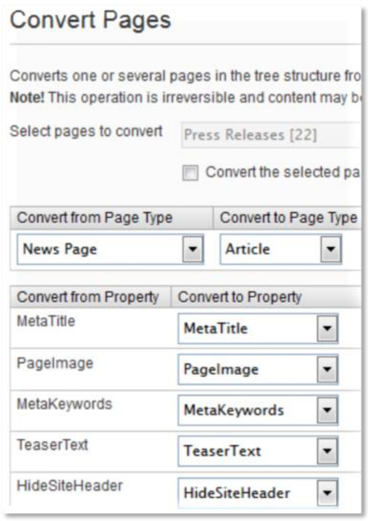
Configuring editors for your properties: http://world.episerver.com/blogs/Linus-Ekstrom/Dates/2013/12/SingleMultiple-selection-in-Episerver-75/

Custom renderers for properties: http://world.episerver.com/Blogs/Linus-Ekstrom/Dates/2012/10/Custom-renderers-for-properties/

Module B – Defining Content Types – Advanced techniques

Converting pages between types

- Converts pages from one page type to another
- Map page properties
  - The system tries to match the page properties when the destination page type is selected.
  - If a Page Property does not exist on the destination Page Type, the option "Remove property permanently" will be selected
- WARNING: You cannot undo a conversion. Content in the database may be removed permanently. Backup your database before performing the conversation. Also run a test conversion to determine whether an undesirable action might occur.



Convert Pages

Converts one or several pages in the tree structure fro
Note! This operation is irreversible and content may b

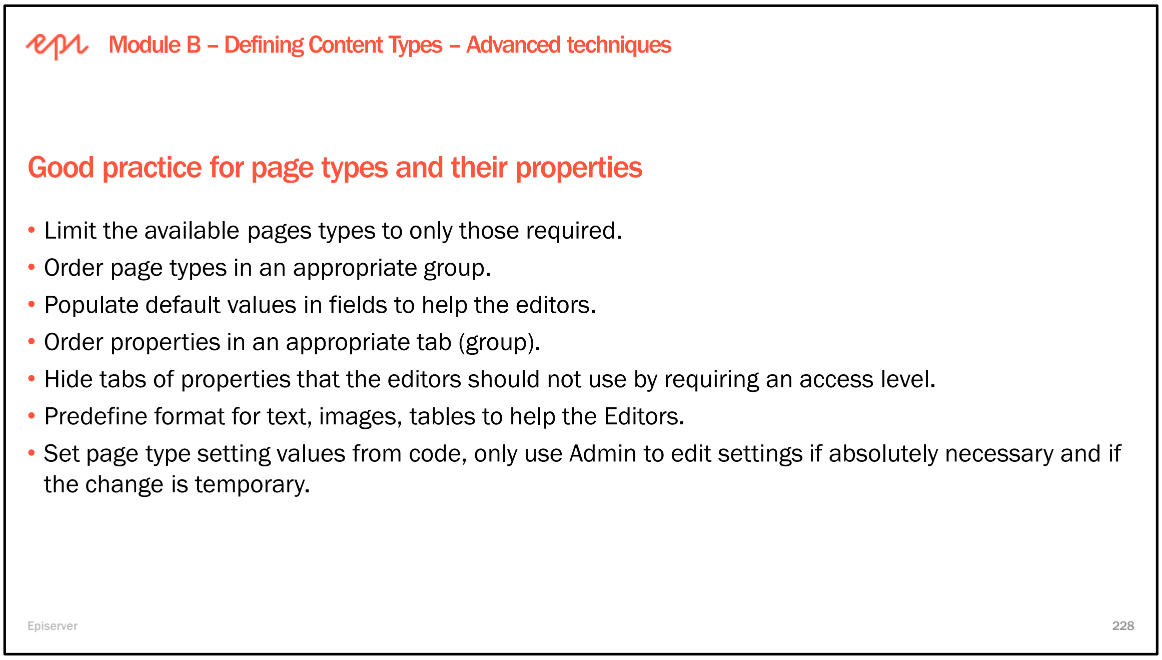
Select pages to convert: Press Releases [22]

☐ Convert the selected pa

Convert from Page Type	Convert to Page Type
News Page	Article

Convert from Property	Convert to Property
MetaTitle	MetaTitle
PageImage	PageImage
MetaKeywords	MetaKeywords
TeaserText	TeaserText
HideSiteHeader	HideSiteHeader

Episerver 227

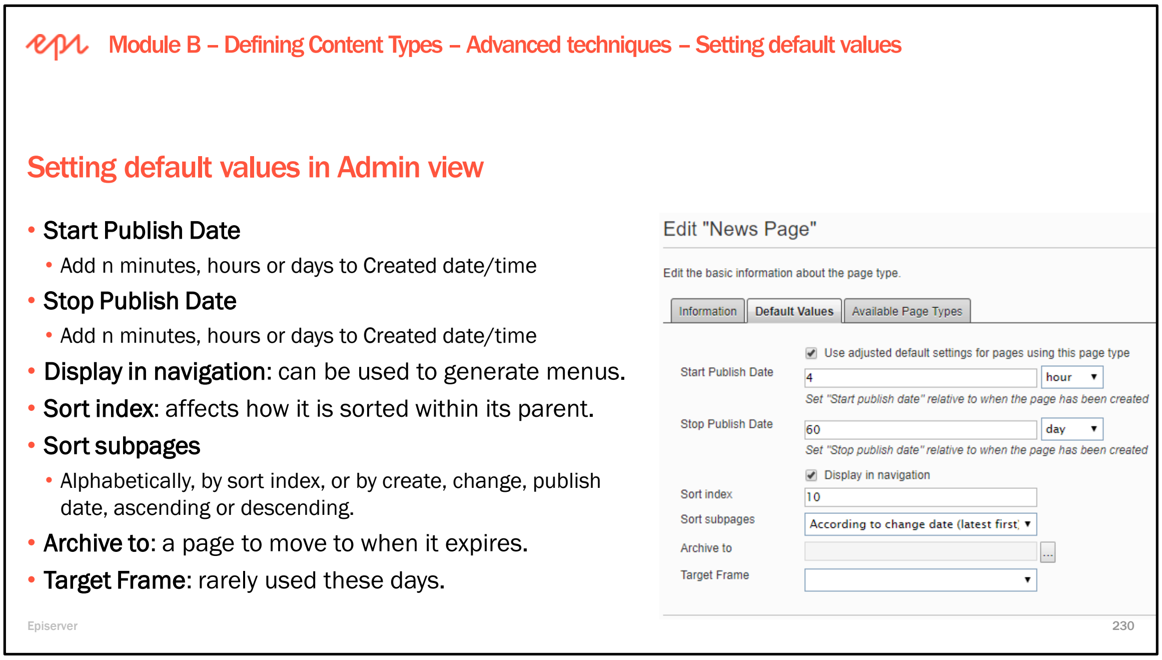


Module B – Defining Content Types – Advanced techniques

Good practice for page types and their properties

- Limit the available pages types to only those required.
- Order page types in an appropriate group.
- Populate default values in fields to help the editors.
- Order properties in an appropriate tab (group).
- Hide tabs of properties that the editors should not use by requiring an access level.
- Predefine format for text, images, tables to help the Editors.
- Set page type setting values from code, only use Admin to edit settings if absolutely necessary and if the change is temporary.

Episerver 228



Module B – Defining Content Types – Advanced techniques – Setting default values

Setting default values in Admin view

- **Start Publish Date**
  - Add n minutes, hours or days to Created date/time
- **Stop Publish Date**
  - Add n minutes, hours or days to Created date/time
- **Display in navigation**: can be used to generate menus.
- **Sort index**: affects how it is sorted within its parent.
- **Sort subpages**
  - Alphabetically, by sort index, or by create, change, publish date, ascending or descending.
- **Archive to**: a page to move to when it expires.
- **Target Frame**: rarely used these days.

Edit "News Page"

Edit the basic information about the page type.

Information | Default Values | Available Page Types

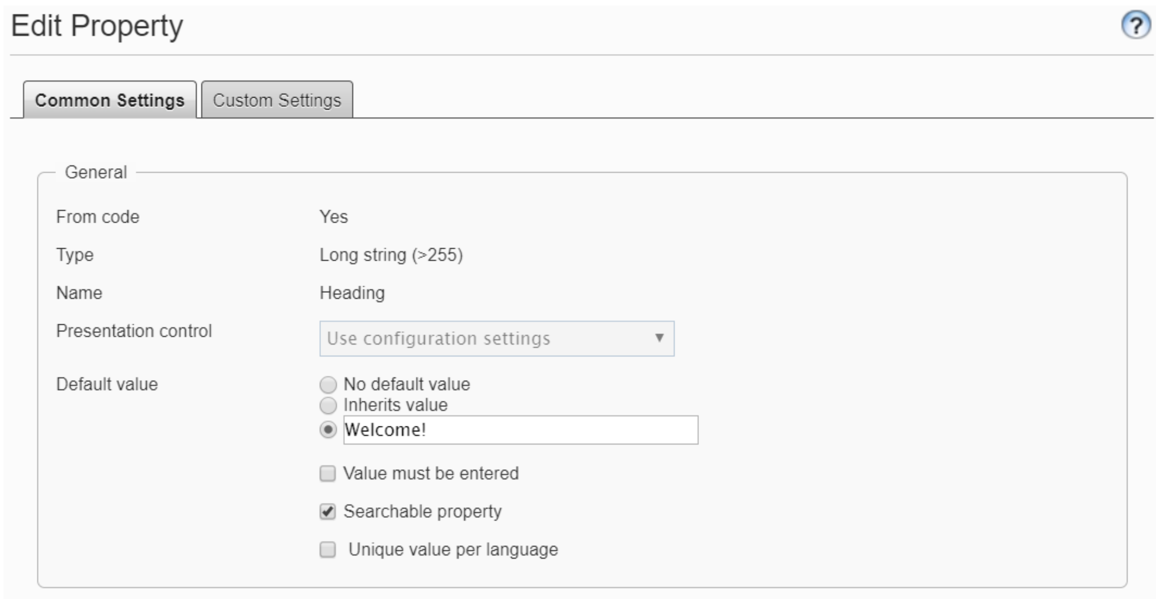
☑ Use adjusted default settings for pages using this page type

Start Publish Date	4 hour
	Set "Start publish date" relative to when the page has been created
Stop Publish Date	60 day
	Set "Stop publish date" relative to when the page has been created
	☑ Display in navigation
Sort index	10
Sort subpages	According to change date (latest first)
Archive to	
Target Frame	

Episerver 230

Setting default values for custom properties

Navigate to **CMS** | **Admin** | **Content Type**, click on a content type, and click one of its properties, for example the **Heading** property:

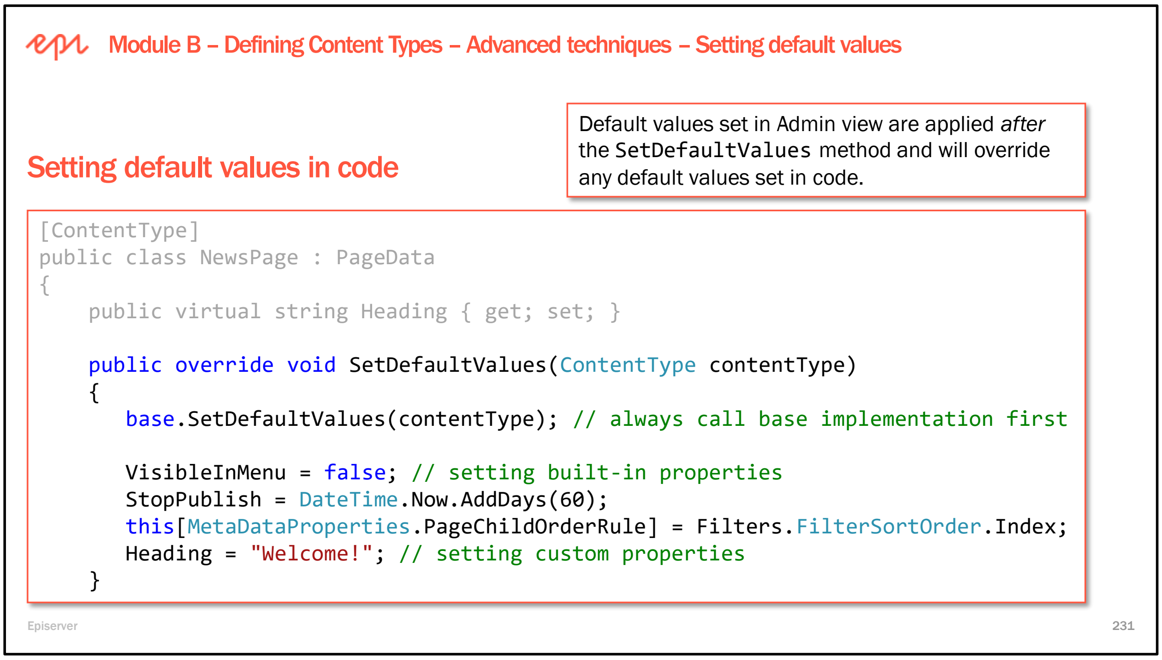


Edit Property

Common Settings | Custom Settings

General

From code	Yes
Type	Long string (>255)
Name	Heading
Presentation control	Use configuration settings
Default value	○ No default value ○ Inherits value ● Welcome!
	☐ Value must be entered
	☑ Searchable property
	☐ Unique value per language



Module B – Defining Content Types – Advanced techniques – Setting default values

Setting default values in code

Default values set in Admin view are applied *after* the `SetDefaultValues` method and will override any default values set in code.

```
[ContentType]
public class NewsPage : PageData
{
    public virtual string Heading { get; set; }

    public override void SetDefaultValues(ContentType contentType)
    {
       base.SetDefaultValues(contentType); // always call base implementation first

       VisibleInMenu = false; // setting built-in properties
       StopPublish = DateTime.Now.AddDays(60);
       this[MetaDataProperties.PageChildOrderRule] = Filters.FilterSortOrder.Index;
       Heading = "Welcome!"; // setting custom properties
    }
```

Episerver 231

The default property values specified in code will be applied to all new pages created from that page type, but defaults for the built-in properties are not visible on the **Default Values** tab in Admin.

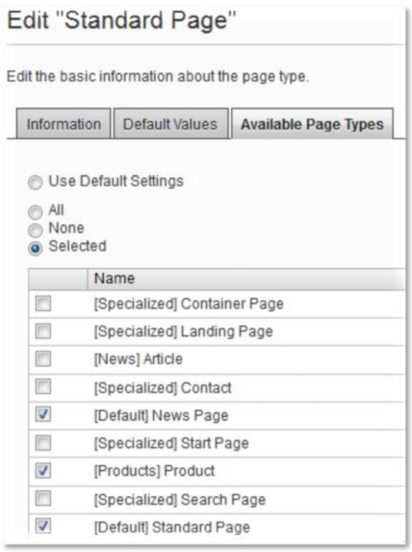
The default value is usually an “empty” value, but not null, for example, zero (0) for a number value type property or an empty string for a string value type property.

A detailed example of setting default values for a content type can be found in Alexander Haneng’s blog on Episerver World: http://world.episerver.com/Blogs/Alexander-Haneng/Dates/2012/9/How-to-define-default-values-for-pages-and-blocks-in-Episerver-CMS-7/

Module B – Defining Content Types – Advanced techniques – Available content types

Available Page Types – Admin view

- Default is **All**, i.e. all page types are available.
  - Means that a page that is created beneath the current page can be based on all possible page types.
- Switching to **Selected** makes it possible to choose which page types may be created beneath the current page type.
  - Simplify the work of editors by limiting the list to pick from.
  - In the screenshot, only News Page, Product, and Standard Page can be created as children of Standard Page.
- Admin view only allows control over children; in code you can control children and parent page types.



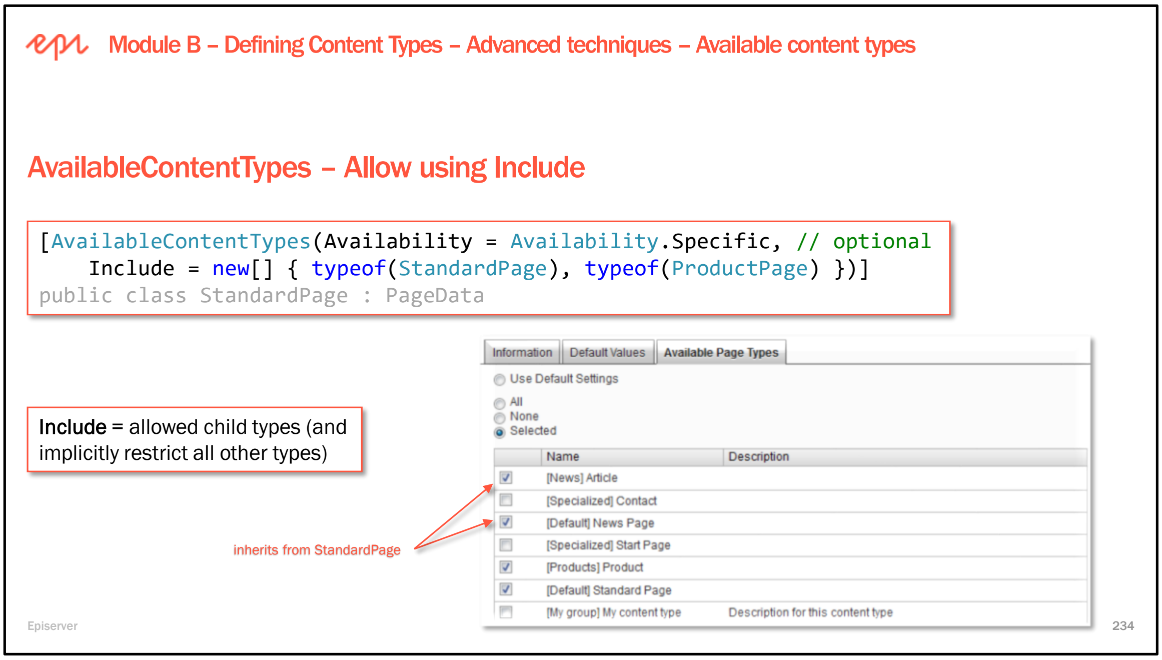
Edit "Standard Page"

Edit the basic information about the page type.

Use Default Settings
All
None
[✓ Selected]

	Name
	[Specialized] Container Page
	[Specialized] Landing Page
	[News] Article
	[Specialized] Contact
✓	[Default] News Page
	[Specialized] Start Page
✓	[Products] Product
	[Specialized] Search Page
✓	[Default] Standard Page

Episerver 233



Module B – Defining Content Types – Advanced techniques – Available content types

AvailableContentTypes – Allow using Include

```
[AvailableContentTypes(Availability = Availability.Specific, // optional
    Include = new[] { typeof(StandardPage), typeof(ProductPage) })]
public class StandardPage : PageData
```

Include = allowed child types (and implicitly restrict all other types)

inherits from StandardPage

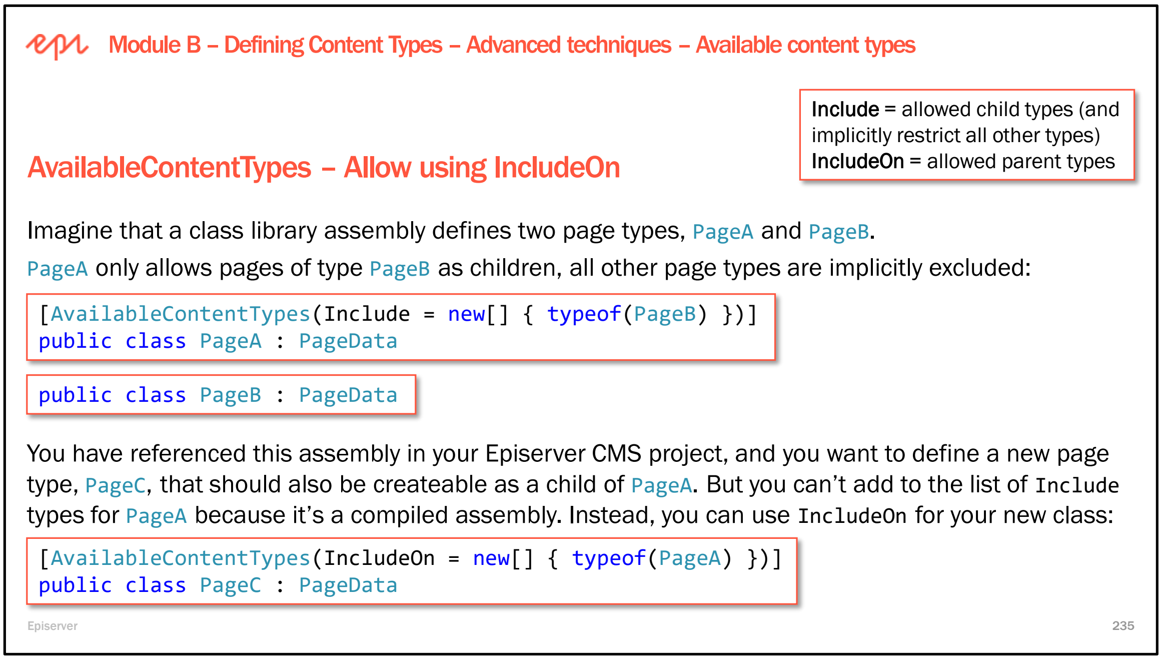
Use Default Settings

All
None
[✓ Selected]

	Name	Description
✓	[News] Article	
	[Specialized] Contact	
✓	[Default] News Page	
	[Specialized] Start Page	
✓	[Products] Product	
✓	[Default] Standard Page	
	[My group] My content type	Description for this content type

Episerver 234

Default behavior: If no AvailableContentTypes attribute is specified for a page type, the default behavior is for the page type to be included on all page types that does not specifically exclude it.
The types given on AvailableContentTypes attribute can either be a typed page (that is a type inheriting PageData) directly or it can be the type of an interface or a base class. At registration all registered types that can be assigned to the specified type will be included. So if for example an interface is specified in the Include list then all typed pages that implement the interface will be included.



Module B – Defining Content Types – Advanced techniques – Available content types

Include = allowed child types (and implicitly restrict all other types)
IncludeOn = allowed parent types

AvailableContentTypes – Allow using IncludeOn

Imagine that a class library assembly defines two page types, `PageA` and `PageB`.
`PageA` only allows pages of type `PageB` as children, all other page types are implicitly excluded:

```
[AvailableContentTypes(Include = new[] { typeof(PageB) })]
public class PageA : PageData
```

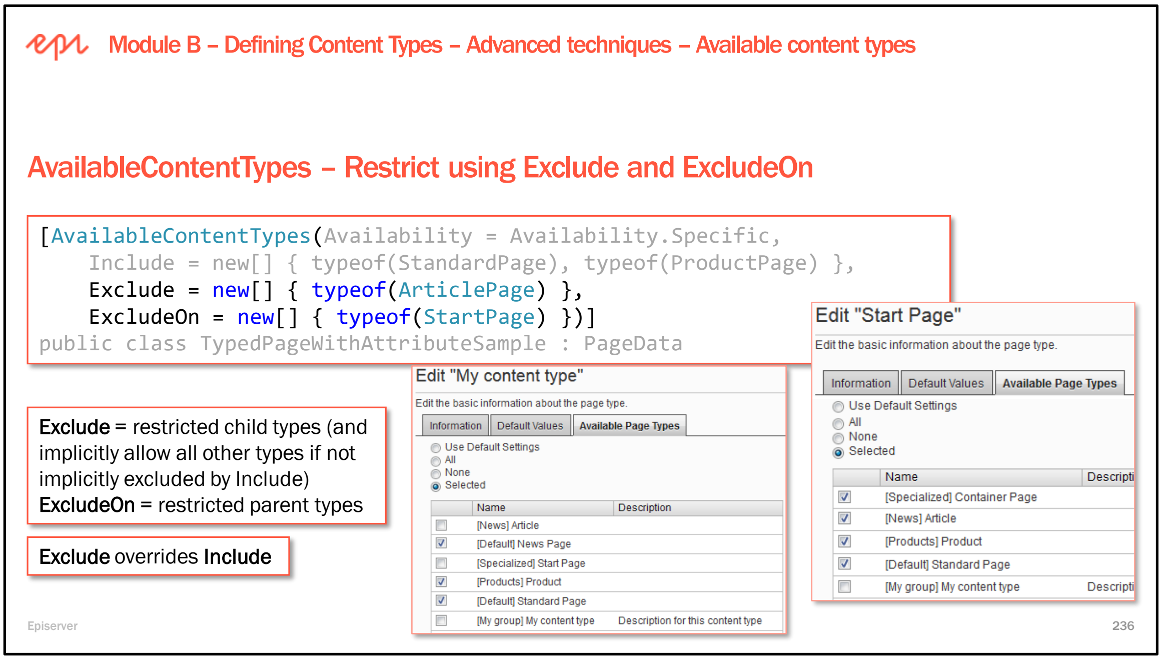
```
public class PageB : PageData
```

You have referenced this assembly in your Episerver CMS project, and you want to define a new page type, `PageC`, that should also be createable as a child of `PageA`. But you can't add to the list of `Include` types for `PageA` because it's a compiled assembly. Instead, you can use `IncludeOn` for your new class:

```
[AvailableContentTypes(IncludeOn = new[] { typeof(PageA) })]
public class PageC : PageData
```

Episerver 235

IncludeOn differs from Include in the way that it is not excluding. That is, for types in IncludeOn that has all page types available no page types will be excluded. Include on the other hand will exclude all typed pages except the ones given in Include.



Module B – Defining Content Types – Advanced techniques – Available content types

AvailableContentTypes – Restrict using Exclude and ExcludeOn

```
[AvailableContentTypes(Availability = Availability.Specific,
    Include = new[] { typeof(StandardPage), typeof(ProductPage) },
    Exclude = new[] { typeof(ArticlePage) },
    ExcludeOn = new[] { typeof(StartPage) })]
public class TypedPageWithAttributeSample : PageData
```

Exclude = restricted child types (and implicitly allow all other types if not implicitly excluded by Include)
ExcludeOn = restricted parent types

Exclude overrides **Include**

Edit "My content type"

Edit the basic information about the page type.

Information | Default Values | **Available Page Types**

Use Default Settings
All
None
[✓ Selected]

	Name	Description
	[News] Article	
✓	[Default] News Page	
	[Specialized] Start Page	
✓	[Products] Product	
✓	[Default] Standard Page	
	[My group] My content type	Description for this content type

Edit "Start Page"

Edit the basic information about the page type.

Information | Default Values | **Available Page Types**

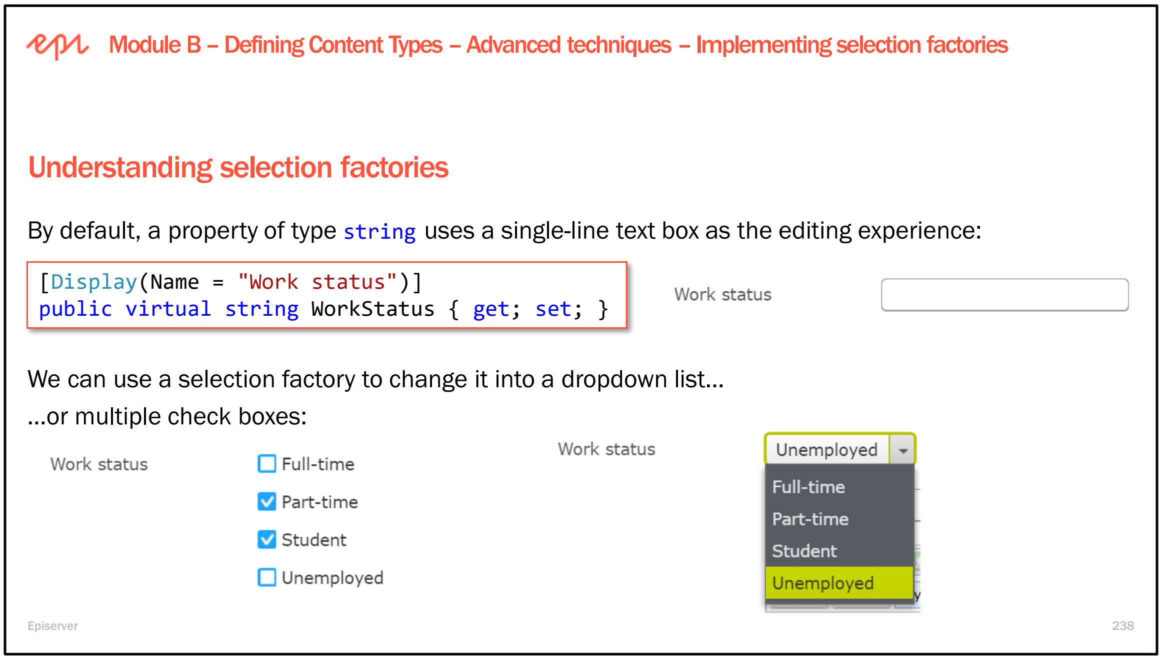
Use Default Settings
All
None
[✓ Selected]

	Name	Descripti
✓	[Specialized] Container Page	
✓	[News] Article	
✓	[Products] Product	
✓	[Default] Standard Page	
	[My group] My content type	Descripti

Episerver

236

Exclude works so that if no types are set on Include then the result will be that all registered page types except the Excluded ones are available. If there are types registered in Include then all types in Include except the ones in Exclude are available.
ExcludeOn states that the page with this attribute should be not available under the any of the typed pages in the type array.



Module B – Defining Content Types – Advanced techniques – Implementing selection factories

Understanding selection factories

By default, a property of type `string` uses a single-line text box as the editing experience:

```
[Display(Name = "Work status")]
public virtual string WorkStatus { get; set; }
```

Work status

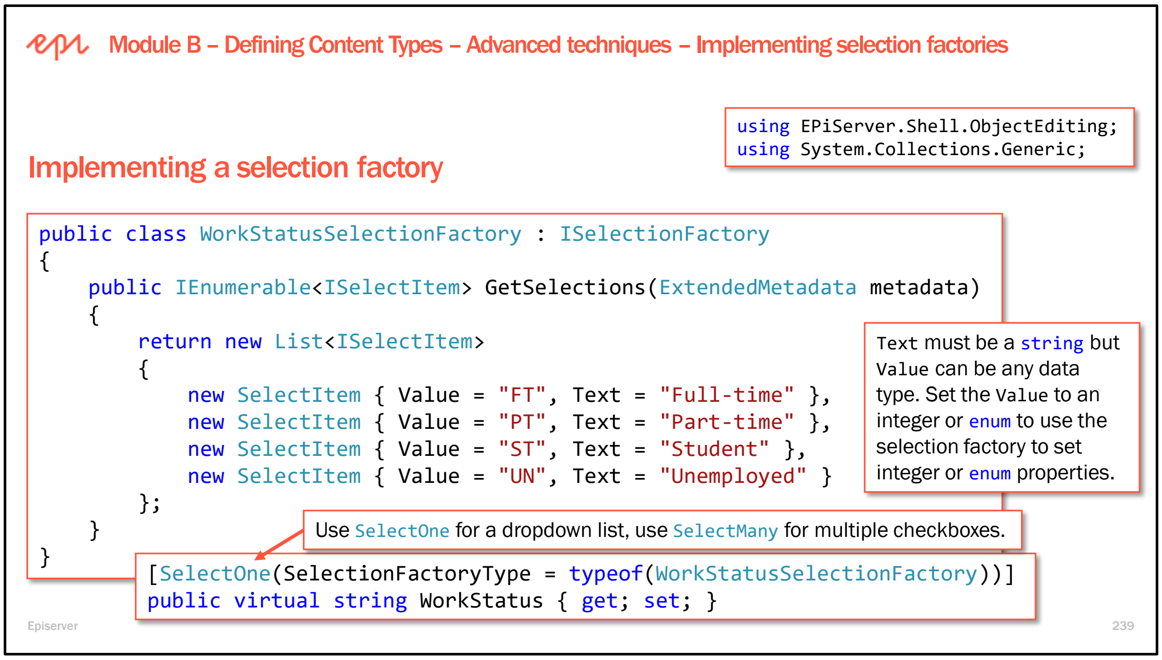
We can use a selection factory to change it into a dropdown list…
…or multiple check boxes:

Work status
- ☐ Full-time
- ☑ Part-time
- ☑ Student
- ☐ Unemployed

Work status

Unemployed
- Full-time
- Part-time
- Student
- Unemployed

Episerver 238



Module B – Defining Content Types – Advanced techniques – Implementing selection factories

Implementing a selection factory

```
using EPiServer.Shell.ObjectEditing;
using System.Collections.Generic;
```

```
public class WorkStatusSelectionFactory : ISelectionFactory
{
    public IEnumerable<ISelectItem> GetSelections(ExtendedMetadata metadata)
    {
        return new List<ISelectItem>
        {
            new SelectItem { Value = "FT", Text = "Full-time" },
            new SelectItem { Value = "PT", Text = "Part-time" },
            new SelectItem { Value = "ST", Text = "Student" },
            new SelectItem { Value = "UN", Text = "Unemployed" }
        };
    }
}
```

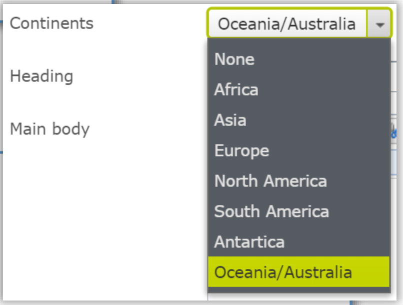
Text must be a `string` but `Value` can be any data type. Set the `Value` to an integer or `enum` to use the selection factory to set integer or `enum` properties.

Use `SelectOne` for a dropdown list, use `SelectMany` for multiple checkboxes.

```
[SelectOne(SelectionFactoryType = typeof(WorkStatusSelectionFactory))]
public virtual string WorkStatus { get; set; }
```

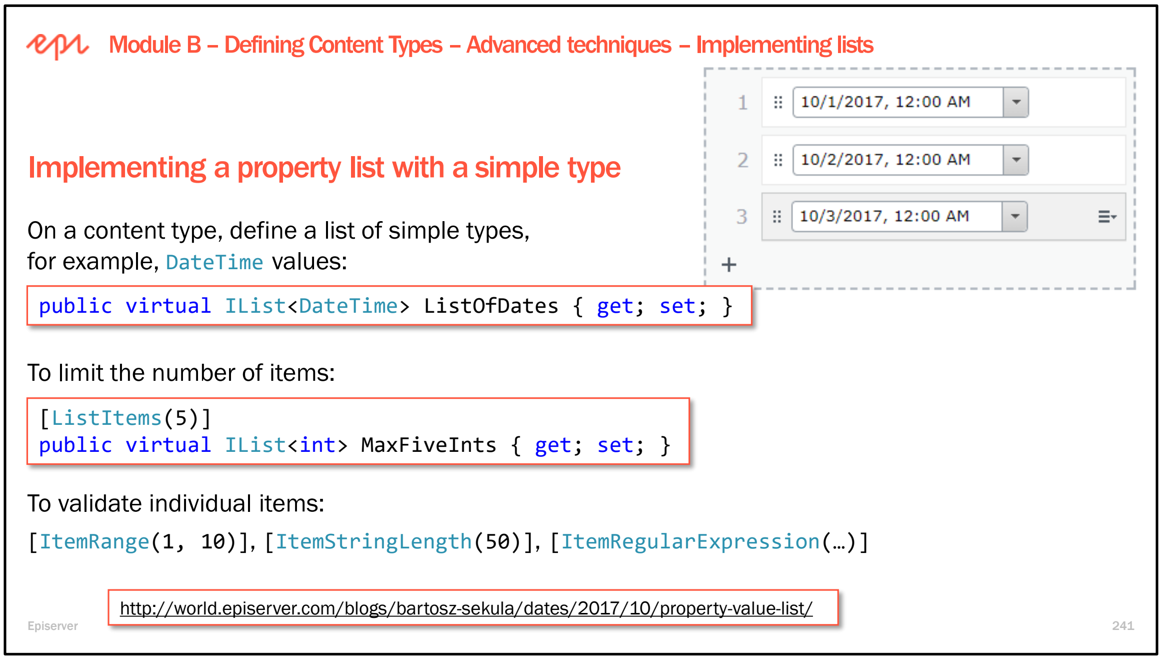
Episerver 239

```
[SelectOne(SelectionFactoryType = typeof(ContinentSelectionFactory))]
public virtual Continent Continent { get; set; }
```



```
using EPiServer.Shell.ObjectEditing;
using System.Collections.Generic;

namespace AlloyTraining.Business.SelectionFactories
{
    public enum Continent
    {
        None, Africa, Asia, Europe, NorthAmerica, SouthAmerica, Antartica, Oceania
    }

    public class ContinentSelectionFactory : ISelectionFactory
    {
        public IEnumerable<ISelectItem> GetSelections(ExtendedMetadata metadata)
        {
            return new List<SelectItem>
            {
                new SelectItem { Value = Continent.None, Text = "None" },
                new SelectItem { Value = Continent.Africa, Text = "Africa" },
                new SelectItem { Value = Continent.Asia, Text = "Asia" },
                new SelectItem { Value = Continent.Europe, Text = "Europe" },
                new SelectItem { Value = Continent.NorthAmerica, Text = "North America" },
                new SelectItem { Value = Continent.SouthAmerica, Text = "South America" },
                new SelectItem { Value = Continent.Antartica, Text = "Antartica" },
                new SelectItem { Value = Continent.Oceania, Text = "Oceania/Australia" }
            };
        }
    }
}
```

```
// or enumerate the enum values but you lose formatted text
return Enum.GetValues(typeof(Continent)).Select(c => new SelectItem {
    Value = c, Text = Enum.GetName(typeof(Continent), c) }).ToList();
```



Module B – Defining Content Types – Advanced techniques – Implementing lists

1 10/1/2017, 12:00 AM
2 10/2/2017, 12:00 AM
3 10/3/2017, 12:00 AM

Implementing a property list with a simple type

On a content type, define a list of simple types,
for example, `DateTime` values:

```
public virtual IList<DateTime> ListOfDates { get; set; }
```

To limit the number of items:

```
[ListItems(5)]
public virtual IList<int> MaxFiveInts { get; set; }
```

To validate individual items:

`[ItemRange(1, 10)], [ItemStringLength(50)], [ItemRegularExpression(…)]`

Episerver

http://world.episerver.com/blogs/bartosz-sekula/dates/2017/10/property-value-list/

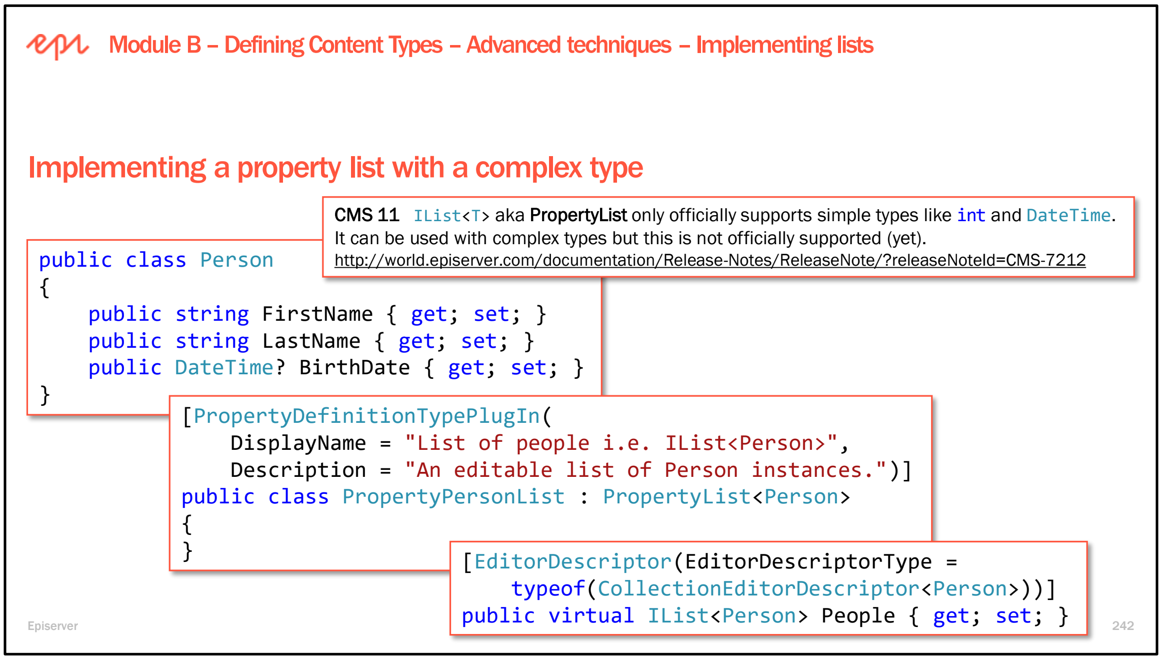
241

If you have defined an `IList<T>` property that you would like editors to be able to edit from on-page edit view, then you must decorate with a `UIHint` to allow `PropertyFor` to render correctly:

```
[UIHint("StringsList")]
public virtual IList<string> Names { get; set; }
```

Add a **StringsList.cshtml** file to **~/Views/Shared/DisplayTemplates** folder:

```
@model IEnumerable<string>
@if (Model != null && Model.Any())
{
    <ul>
        @foreach (var stringValue in Model)
        {
            <li>@stringValue</li>
        }
    </ul>
}
```



Module B – Defining Content Types – Advanced techniques – Implementing lists

Implementing a property list with a complex type

CMS 11 `IList<T>` aka **PropertyList** only officially supports simple types like `int` and `DateTime`. It can be used with complex types but this is not officially supported (yet).
http://world.episerver.com/documentation/Release-Notes/ReleaseNote/?releaseNoteId=CMS-7212

```
public class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime? BirthDate { get; set; }
}
```

```
[PropertyDefinitionTypePlugIn(
    DisplayName = "List of people i.e. IList<Person>",
    Description = "An editable list of Person instances.")]
public class PropertyPersonList : PropertyList<Person>
{
}
```

```
[EditorDescriptor(EditorDescriptorType =
    typeof(CollectionEditorDescriptor<Person>))]
public virtual IList<Person> People { get; set; }
```

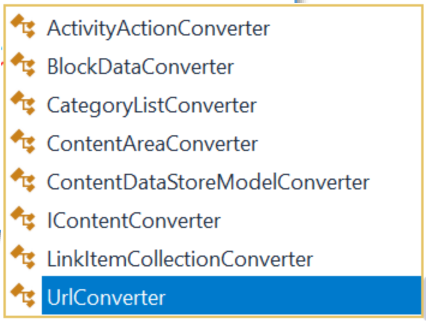
Episerver

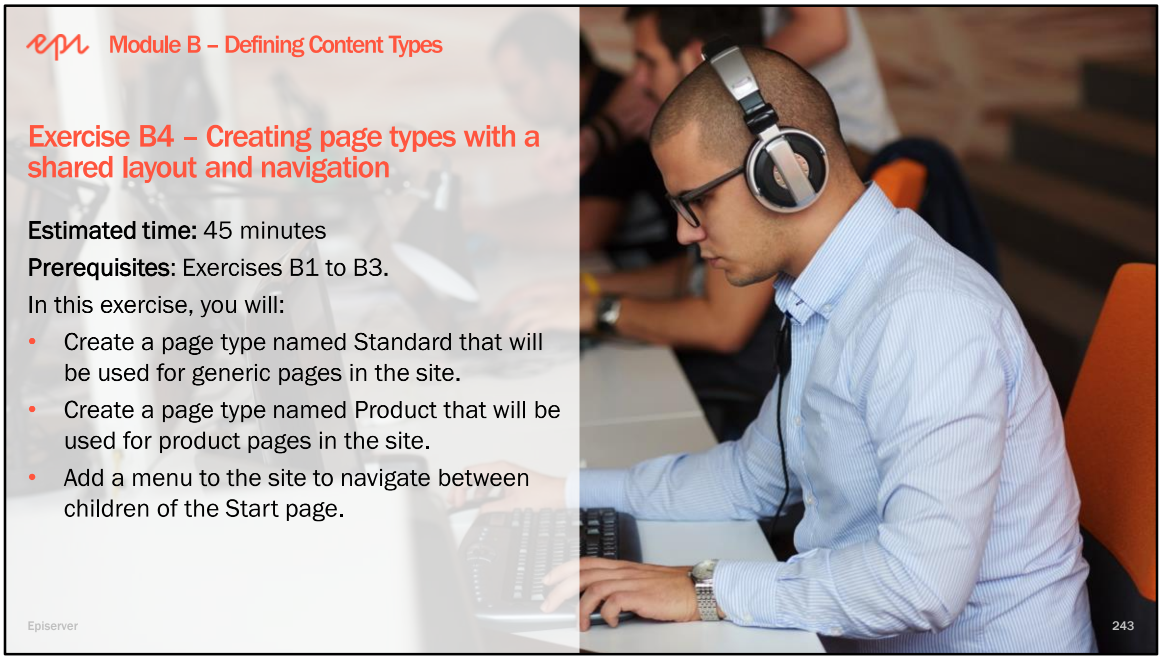
242

The dangers of using pre-release API's
https://www.brianweet.com/2017/02/24/dangers-of-using-pre-release-apis.html

To serialize Episerver property types like `Url`

```
using EPiServer;
using EPiServer.Cms.Shell.Json.Internal; // no breaking changes guarantee
using Newtonsoft.Json;
using System;

public class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime? BirthDate { get; set; }

    [JsonProperty]
    [JsonConverter(typeof(UrlConverter))]
    public Url HomePage { get; set; }
}
```





Module B – Defining Content Types

Exercise B4 – Creating page types with a shared layout and navigation

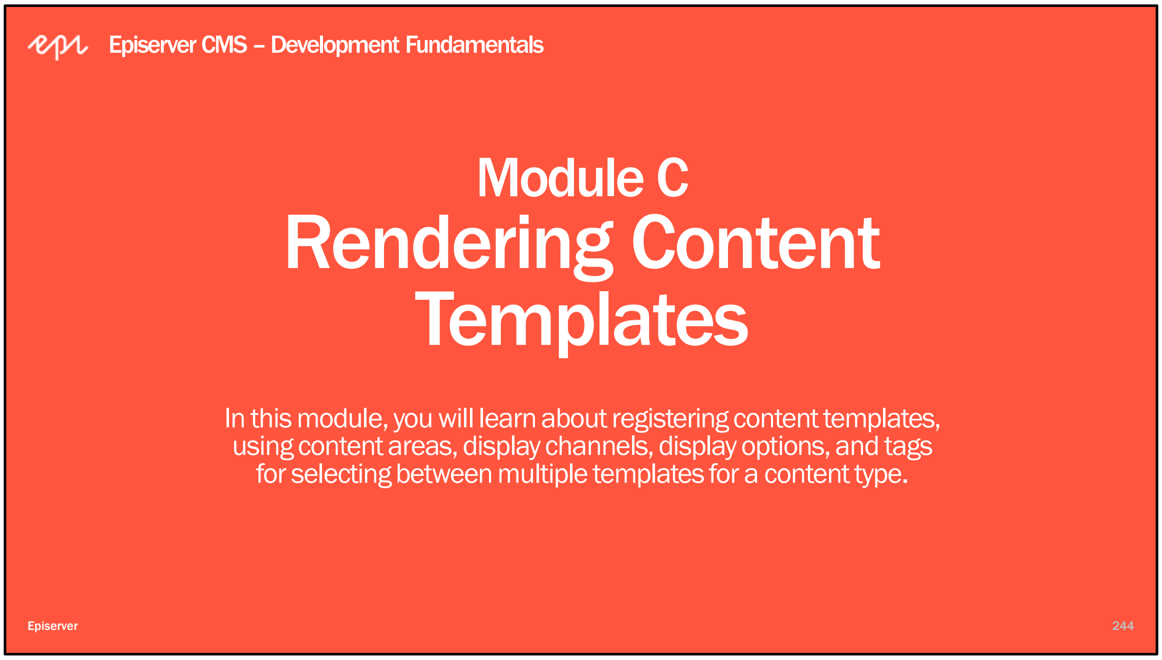
Estimated time: 45 minutes

Prerequisites: Exercises B1 to B3.

In this exercise, you will:

- Create a page type named Standard that will be used for generic pages in the site.
- Create a page type named Product that will be used for product pages in the site.
- Add a menu to the site to navigate between children of the Start page.

Episerver 243

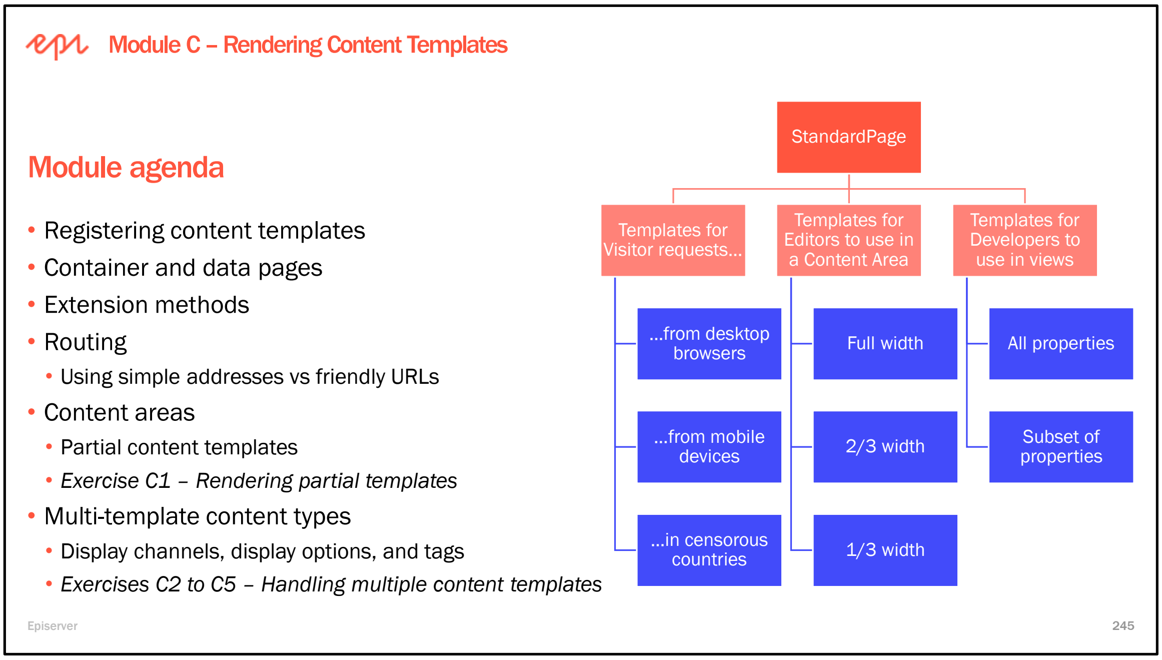


Episerver CMS – Development Fundamentals

Module C
Rendering Content Templates

In this module, you will learn about registering content templates, using content areas, display channels, display options, and tags for selecting between multiple templates for a content type.

Episerver 244

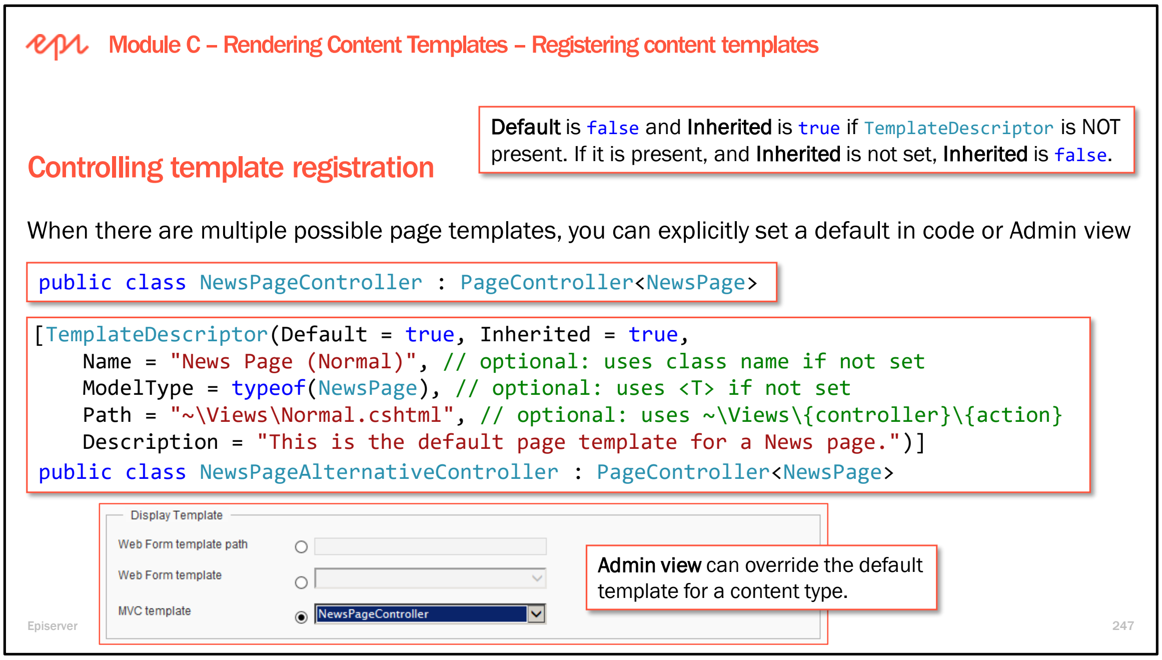


Module C – Rendering Content Templates

Module agenda

- Registering content templates
- Container and data pages
- Extension methods
- Routing
  - Using simple addresses vs friendly URLs
- Content areas
  - Partial content templates
  - *Exercise C1 – Rendering partial templates*
- Multi-template content types
  - Display channels, display options, and tags
  - *Exercises C2 to C5 – Handling multiple content templates*

Episerver 245



Module C – Rendering Content Templates – Registering content templates

Controlling template registration

Default is `false` and **Inherited** is `true` if `TemplateDescriptor` is NOT present. If it is present, and **Inherited** is not set, **Inherited** is `false`.

When there are multiple possible page templates, you can explicitly set a default in code or Admin view

```
public class NewsPageController : PageController<NewsPage>
```

```
[TemplateDescriptor(Default = true, Inherited = true,
    Name = "News Page (Normal)", // optional: uses class name if not set
    ModelType = typeof(NewsPage), // optional: uses <T> if not set
    Path = "~\Views\Normal.cshtml", // optional: uses ~\Views\{controller}\{action}
    Description = "This is the default page template for a News page.")]
public class NewsPageAlternativeController : PageController<NewsPage>
```

Display Template

Web Form template path ( )

Web Form template ( )

MVC template (●) NewsPageController

Admin view can override the default template for a content type.

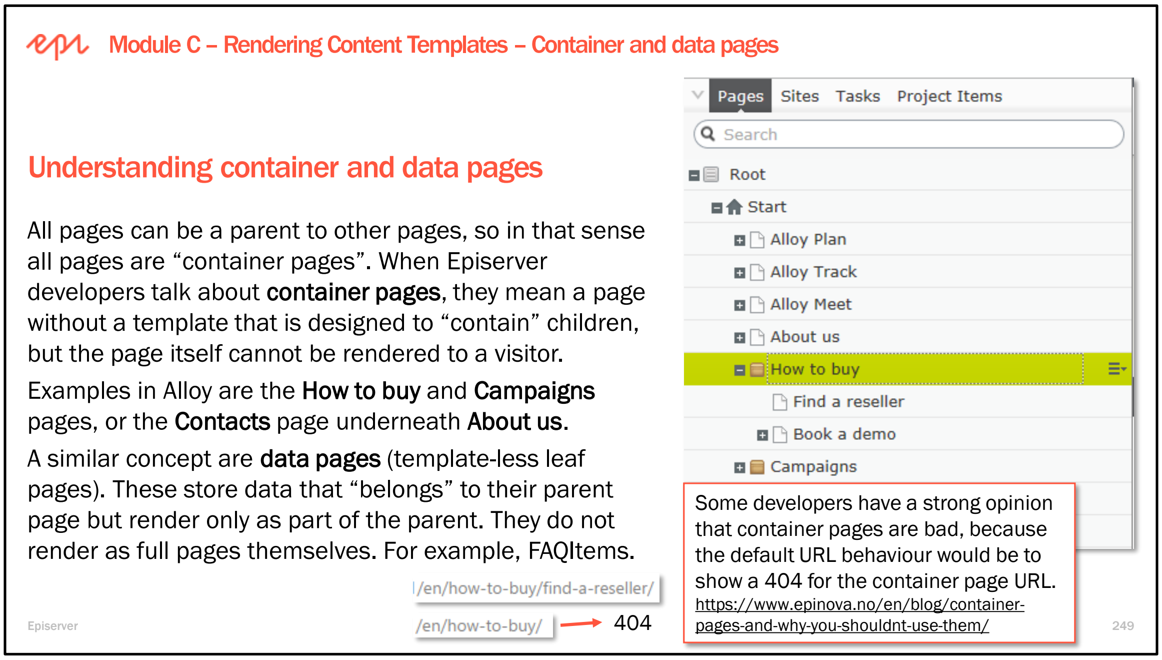
Episerver 247

The [TemplateDescriptor] attribute can be used on templates to add meta data to the template. The attribute can also be used to set the template as the default template for the content data and whether page types that inherit from the one this template supports should also inherit this template.

Behavior if TemplateDescriptor attribute is present but no parameters are specified:

- Default = false
- Inherited = false
- Path = null
  - The path to the template to be rendered only needs to be set if folder structure does not follow namespace structure. There is a namespace convention where the file will be searched for in the path according to the namespace.
- Description = null

Important regarding inheritance of templates:
If the TemplateDescriptor attribute is not present at all, or present with inherited set to true, the template will be available to render all page types that inherit from this one. This can be useful, for example, if you want to have a fallback template for content types that do not have a specific template.
In the case you DON'T want this behaviour, you need to add the TemplateDescriptor attribute and mark it with Inherited=false.



Module C – Rendering Content Templates – Container and data pages

Understanding container and data pages

All pages can be a parent to other pages, so in that sense all pages are “container pages”. When Episerver developers talk about **container pages**, they mean a page without a template that is designed to “contain” children, but the page itself cannot be rendered to a visitor.

Examples in Alloy are the **How to buy** and **Campaigns** pages, or the **Contacts** page underneath **About us**.

A similar concept are **data pages** (template-less leaf pages). These store data that “belongs” to their parent page but render only as part of the parent. They do not render as full pages themselves. For example, FAQItems.

Pages Sites Tasks Project Items

Search

Root
Start
Alloy Plan
Alloy Track
Alloy Meet
About us
How to buy
Find a reseller
Book a demo
Campaigns

/en/how-to-buy/find-a-reseller/

/en/how-to-buy/ → 404

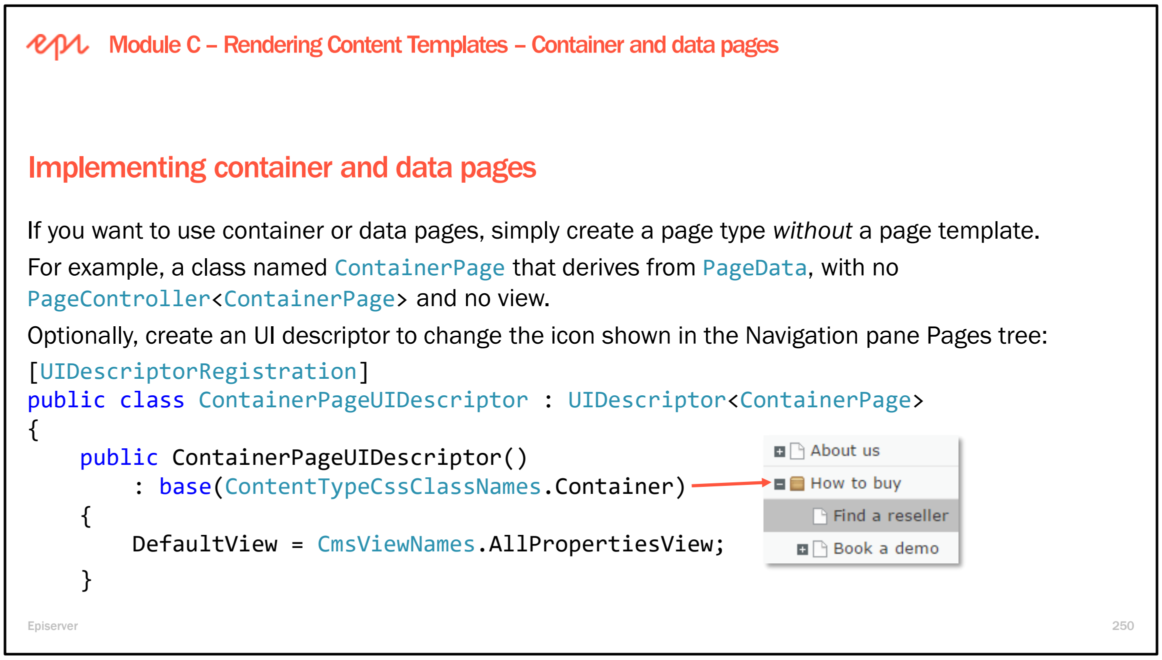
Some developers have a strong opinion that container pages are bad, because the default URL behaviour would be to show a 404 for the container page URL.
https://www.epinova.no/en/blog/container-pages-and-why-you-shouldnt-use-them/

Episerver 249

Key notes for container pages

- Page types without a public template are called “container pages”.
- Container pages have no preview and cannot be linked from other pages.
- Cannot be accessed by a URL (404).
- Examples of use:
  - Often used to logically group pages in the content tree, for example days/months/years in a news archive.
  - Can be used as a settings container
  - Data pages are content items that are never rendered stand-alone but where the content is included in listings, landing pages etc. where they are rendered via another control and where the editor should be able to work with the data.

More details on customizing the look and behavior in the UI for content types can be found in this blog written by Linus Ekström: http://world.episerver.com/Blogs/Linus-Ekstrom/Dates/2013/12/Customizing-the-look-and-behavior-in-the-UI-for-your-content-types/



Module C – Rendering Content Templates – Container and data pages

Implementing container and data pages

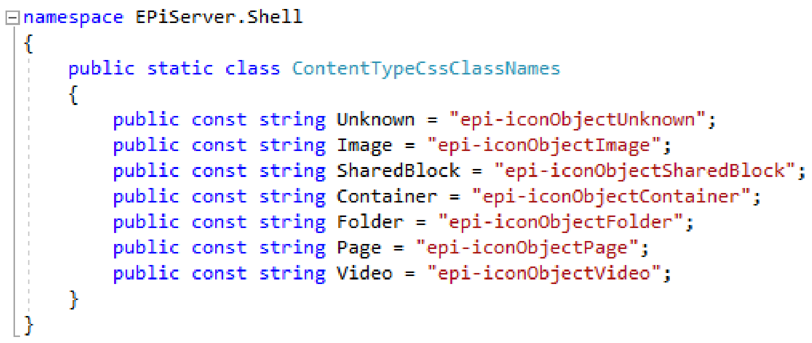
If you want to use container or data pages, simply create a page type *without* a page template.

For example, a class named ContainerPage that derives from PageData, with no PageController<ContainerPage> and no view.

Optionally, create an UI descriptor to change the icon shown in the Navigation pane Pages tree:

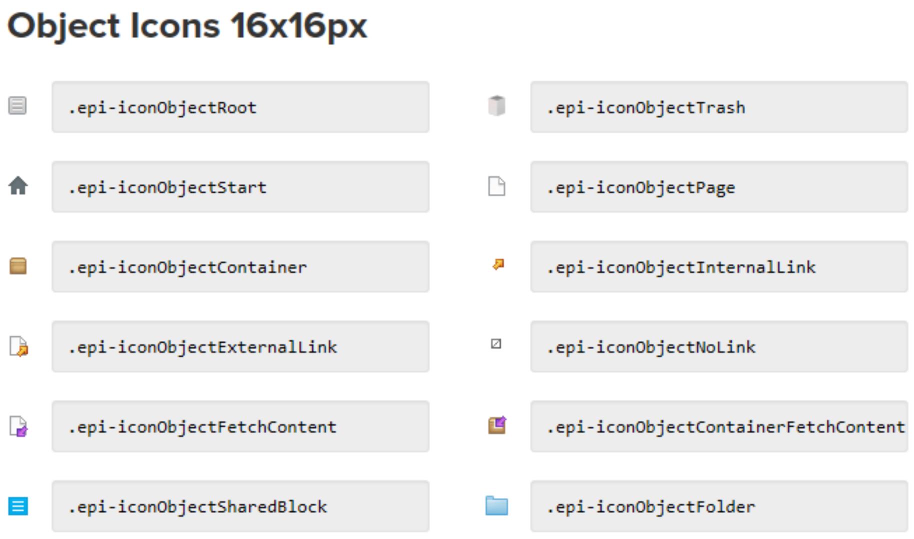
```
[UIDescriptorRegistration]
public class ContainerPageUIDescriptor : UIDescriptor<ContainerPage>
{
    public ContainerPageUIDescriptor()
        : base(ContentTypeCssClassNames.Container)
    {
        DefaultView = CmsViewNames.AllPropertiesView;
    }
```

Episerver 250



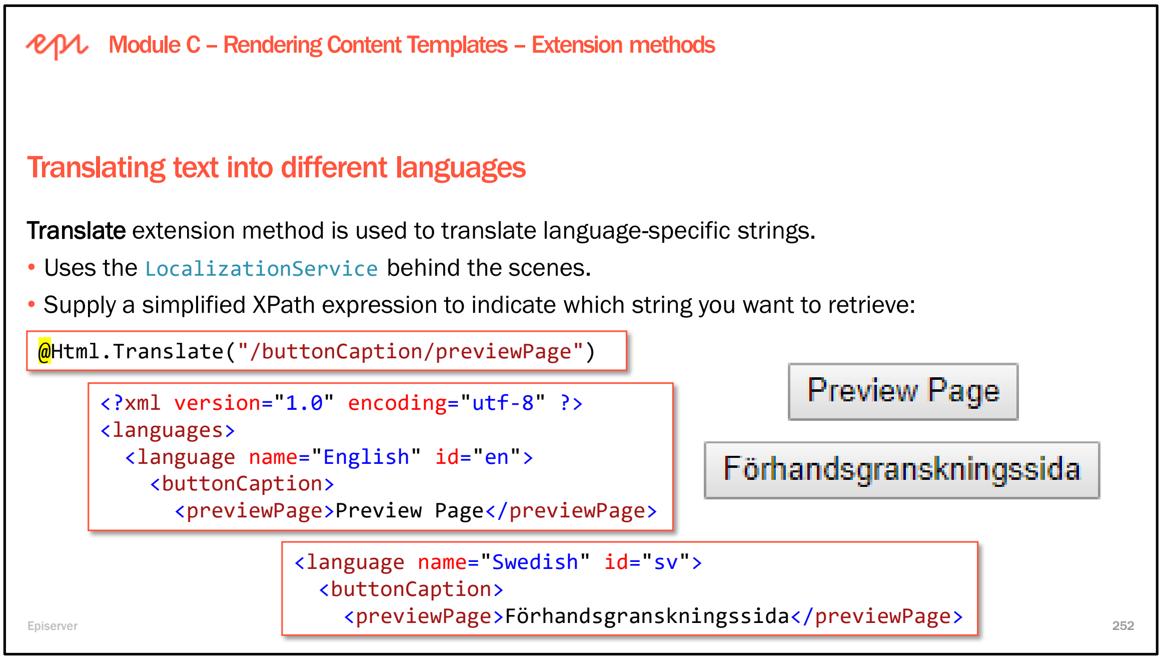
```
namespace EPiServer.Shell
{
    public static class ContentTypeCssClassNames
    {
        public const string Unknown = "epi-iconObjectUnknown";
        public const string Image = "epi-iconObjectImage";
        public const string SharedBlock = "epi-iconObjectSharedBlock";
        public const string Container = "epi-iconObjectContainer";
        public const string Folder = "epi-iconObjectFolder";
        public const string Page = "epi-iconObjectPage";
        public const string Video = "epi-iconObjectVideo";
    }
}
```

View the list of built-in icons at http://ux.episerver.com/



Object Icons 16x16px

.epi-iconObjectRoot
.epi-iconObjectTrash
.epi-iconObjectStart
.epi-iconObjectPage
.epi-iconObjectContainer
.epi-iconObjectInternalLink
.epi-iconObjectExternalLink
.epi-iconObjectNoLink
.epi-iconObjectFetchContent
.epi-iconObjectContainerFetchContent
.epi-iconObjectSharedBlock
.epi-iconObjectFolder



Module C – Rendering Content Templates – Extension methods

Translating text into different languages

Translate extension method is used to translate language-specific strings.

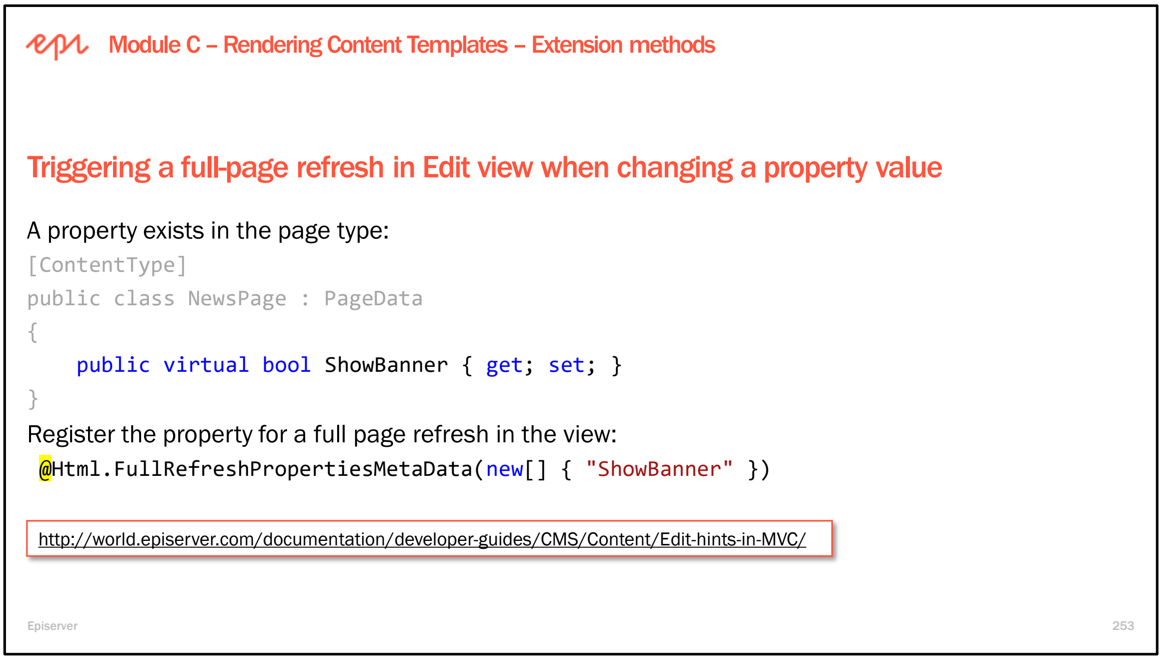
- Uses the `LocalizationService` behind the scenes.
- Supply a simplified XPath expression to indicate which string you want to retrieve:

```
@Html.Translate("/buttonCaption/previewPage")
```

```
<?xml version="1.0" encoding="utf-8" ?>
<languages>
  <language name="English" id="en">
    <buttonCaption>
      <previewPage>Preview Page</previewPage>
```

```
<language name="Swedish" id="sv">
  <buttonCaption>
    <previewPage>Förhandsgranskningssida</previewPage>
```

Episerver 252



Module C – Rendering Content Templates – Extension methods

Triggering a full-page refresh in Edit view when changing a property value

A property exists in the page type:

```
[ContentType]
public class NewsPage : PageData
{
    public virtual bool ShowBanner { get; set; }
}
```

Register the property for a full page refresh in the view:

```
 @Html.FullRefreshPropertiesMetaData(new[] { "ShowBanner" })
```

http://world.episerver.com/documentation/developer-guides/CMS/Content/Edit-hints-in-MVC/

Episerver 253

Trigger full page refresh when changing the value of a property

Some properties can affect the rendering of several parts of the page. For instance, you may have a boolean value on you page type that enables/disables several panels. To get a correct preview of such a property you would need to do a full refresh of the page. The following is needed to achieve this:
1: The names of properties that trigger a full refresh is retrieved from and stored as a comma separated list in PageBase.EditHints, so in order to register a property for full refresh preview you will need to add it to this list. In any page that inherits from PageBase this is done by calling EditHints.AddFullRefreshFor(p => p.<yourproperty>).
2: You also have to make sure that the Html Helper Html.FullRefreshPropertiesMetaData() is used somewhere in the markup if you have registered any properties for full refresh.
Example:

- In the View:

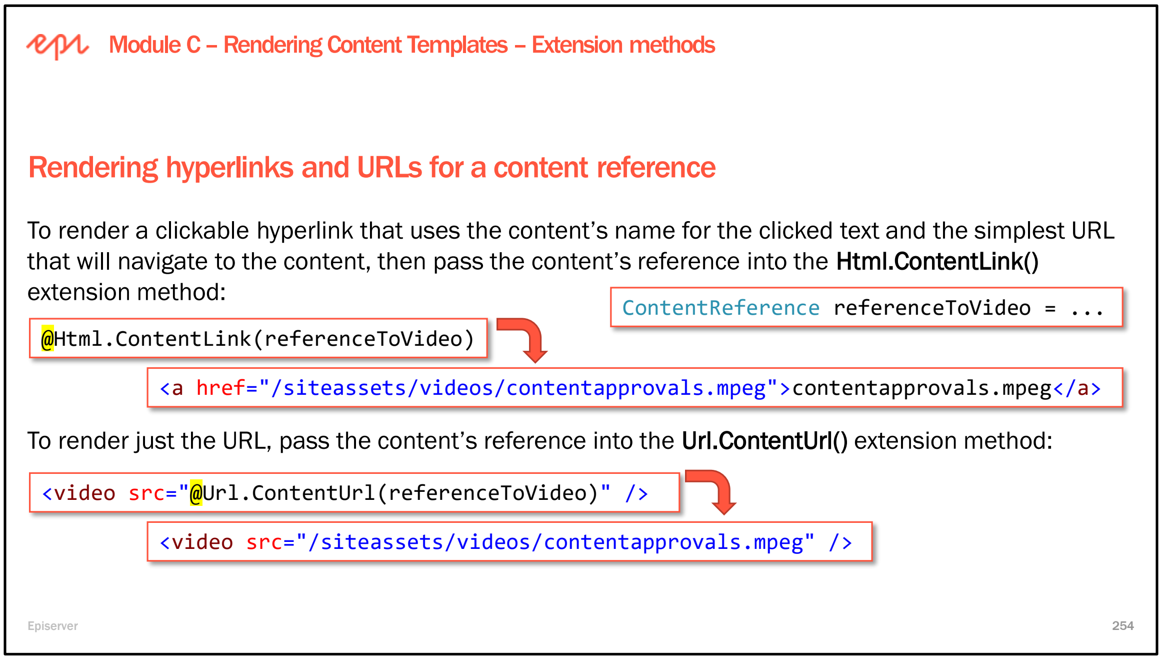
```
@Html.FullRefreshPropertiesMetaData()

@* Alternative to registering the AddFullRefreshFor in your Controller *@
@*Html.FullRefreshPropertiesMetaData(new [ ]{"ShowBanner"}) *@
```

- In the Controller code:

```
    EditHints.AddFullRefreshFor(p => p.ShowBanner);
```

- In the Model:

```
[ContentType]
public class MyPageType : PageData
{
    public virtual string Heading { get; set; }
    public virtual XhtmlString MainBody { get; set; }
    public virtual bool ShowBanner { get; set; }
}
```



Module C – Rendering Content Templates – Extension methods

Rendering hyperlinks and URLs for a content reference

To render a clickable hyperlink that uses the content's name for the clicked text and the simplest URL that will navigate to the content, then pass the content's reference into the **Html.ContentLink()** extension method:

```
ContentReference referenceToVideo = ...
```

```
@Html.ContentLink(referenceToVideo)
```

```
<a href="/siteassets/videos/contentapprovals.mpeg">contentapprovals.mpeg</a>
```

To render just the URL, pass the content's reference into the **Url.ContentUrl()** extension method:

```
<video src="@Url.ContentUrl(referenceToVideo)" />
```

```
<video src="/siteassets/videos/contentapprovals.mpeg" />
```

Episerver 254

ContentLink and ContentUrl extension methods only work on content that has URLs: pages and media.

To render blocks, write an extension method to fetch the block object using the content reference, then use the RenderContentData extension method.

Rendering content with a partial template

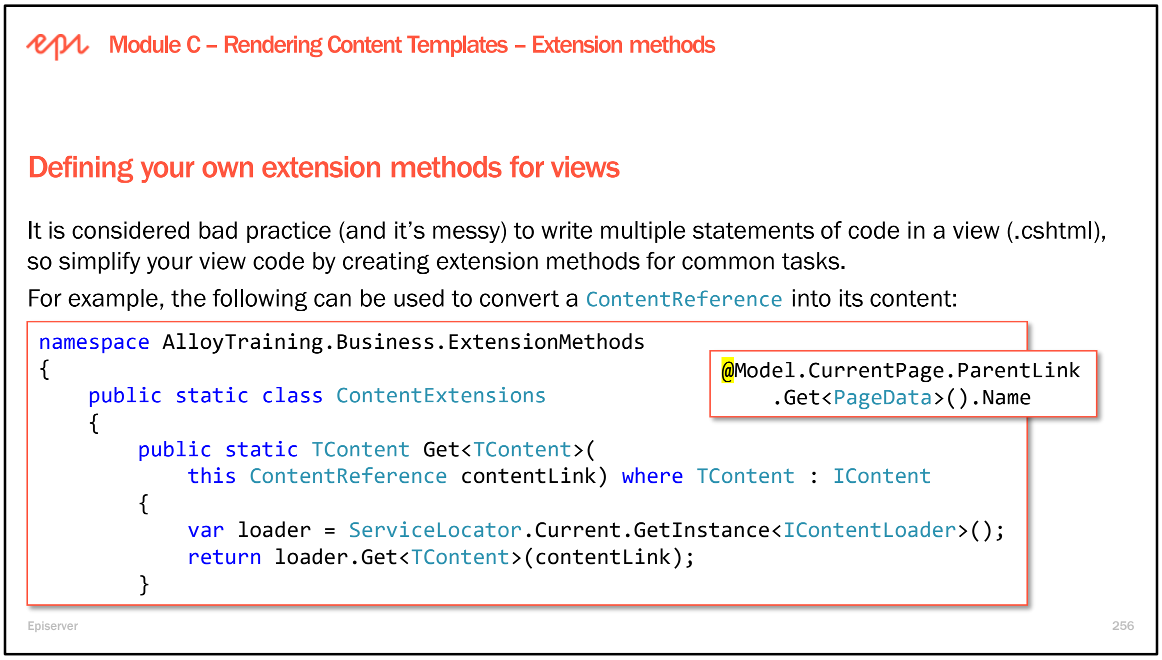
To render the references to content items in a content area, call the `Html.PropertyFor()` extension method on the content area property. This method will enumerate each content reference, load the content item, and render it using its partial template (if it has one).

To render a content item using its partial template (if it has one), then pass the content into the `Html.RenderContentData()` extension method:

```
IContent someContent = ...
```

```
@Html.RenderContentData(someContent, isContentInContentArea: false)
```

...but if we have a content reference instead of the content item, we need a way to manually load the content.



Module C – Rendering Content Templates – Extension methods

Defining your own extension methods for views

It is considered bad practice (and it's messy) to write multiple statements of code in a view (.cshtml), so simplify your view code by creating extension methods for common tasks.

For example, the following can be used to convert a `ContentReference` into its content:

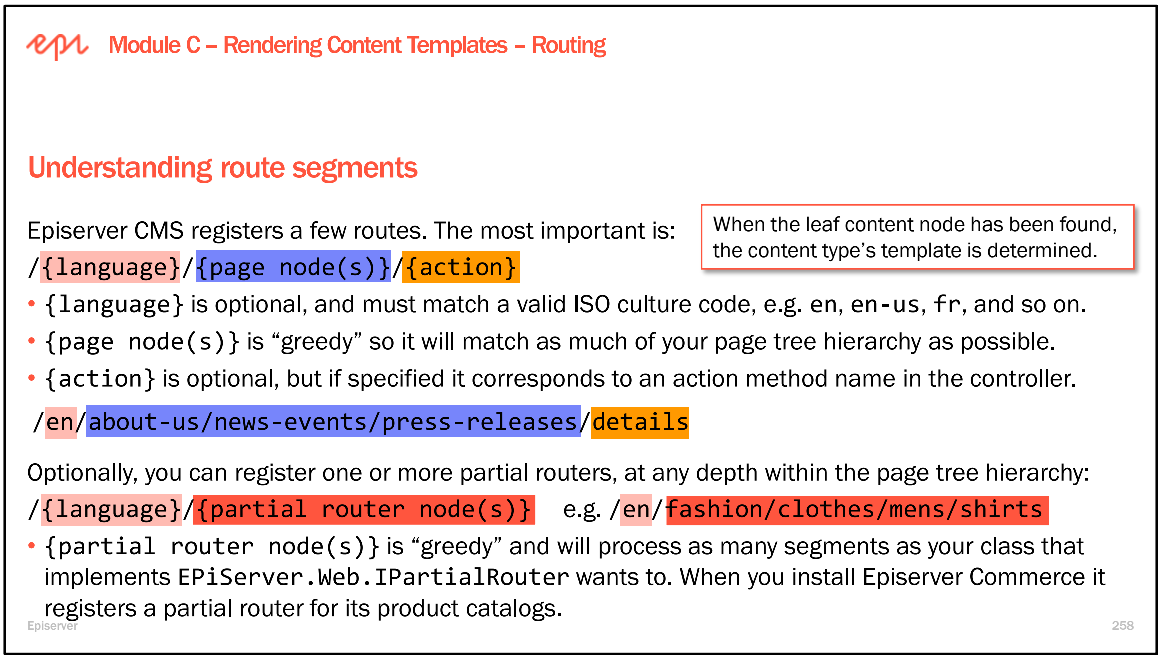
```
namespace AlloyTraining.Business.ExtensionMethods
{
    public static class ContentExtensions
    {
        public static TContent Get<TContent>(
            this ContentReference contentLink) where TContent : IContent
        {
            var loader = ServiceLocator.Current.GetInstance<IContentLoader>();
            return loader.Get<TContent>(contentLink);
        }
```

```
@Model.CurrentPage.ParentLink
    .Get<PageData>().Name
```

Episerver 256

Another example extension method, that returns an absolute URL from a page reference, and you will use it in an exercise in this module:

```
public static string ExternalURLFromReference(this PageReference p)
{
    var loader = ServiceLocator.Current.GetInstance<IContentLoader>();

    PageData page = loader.Get<PageData>(p);

    UrlBuilder pageURLBuilder = new UrlBuilder(page.LinkURL);

    Global.UrlRewriteProvider.ConvertToExternal(pageURLBuilder,
        page.PageLink, UTF8Encoding.UTF8);

    string pageURL = pageURLBuilder.ToString();

    UriBuilder uriBuilder = new UriBuilder(EPiServer.Web.SiteDefinition.Current.SiteUrl);

    uriBuilder.Path = pageURL;

    return uriBuilder.Uri.AbsoluteUri;
}
```



Module C – Rendering Content Templates – Routing

Understanding route segments

Episerver CMS registers a few routes. The most important is:

`/{language}/{page node(s)}/{action}`

When the leaf content node has been found, the content type's template is determined.

- `{language}` is optional, and must match a valid ISO culture code, e.g. `en`, `en-us`, `fr`, and so on.
- `{page node(s)}` is "greedy" so it will match as much of your page tree hierarchy as possible.
- `{action}` is optional, but if specified it corresponds to an action method name in the controller.

`/en/about-us/news-events/press-releases/details`

Optionally, you can register one or more partial routers, at any depth within the page tree hierarchy:

`/{language}/{partial router node(s)}` e.g. `/en/fashion/clothes/mens/shirts`

- `{partial router node(s)}` is "greedy" and will process as many segments as your class that implements `EPiServer.Web.IPartialRouter` wants to. When you install Episerver Commerce it registers a partial router for its product catalogs.

Episerver 258

Routing

By default, the routing system in Episerver uses `System.Web.Routing`, with specific segments added for language, node, and a partial route. Routing is automatically handled based on content type.

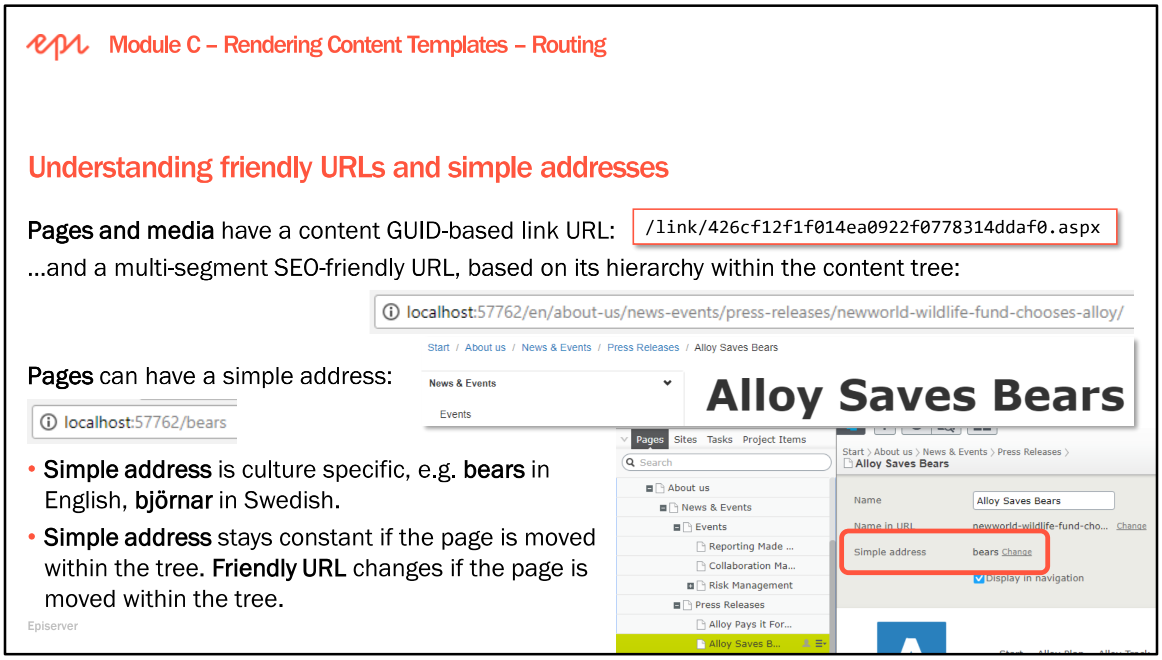
There are several routes registered by default:

- Shell modules have routes registered to support routing to gadgets.
- CMS registers a number of routes by default:
    - routing a simple address.
    - routing for sites (can be several sites in a multi-site environment).
    - routing pages/content from the root (that is, pages/content not under any start page).
- The "ordinary" MVC route "{controller}/{action}" also is registered to support partial requests through Html.RenderAction. However, direct browsing to those routes are prevented.

Events

The `EPiServer.Web.Routing.IContentRouteEvents` interface exposes the events `RoutingContent` and `RoutedContent`, which are raised during incoming routing. `RoutingContent` events are raised before executing the default routing implementation, and the content that matches the request is set in an event handler. `RoutedContent` events are raised after executing the default routing, and the routed content is replaced in an event handler.

https://world.episerver.com/documentation/developer-guides/CMS/routing/



Module C – Rendering Content Templates – Routing

Understanding friendly URLs and simple addresses

Pages and media have a content GUID-based link URL: `/link/426cf12f1f014ea0922f0778314ddaf0.aspx`

...and a multi-segment SEO-friendly URL, based on its hierarchy within the content tree:

localhost:57762/en/about-us/news-events/press-releases/newworld-wildlife-fund-chooses-alloy/

Pages can have a simple address:

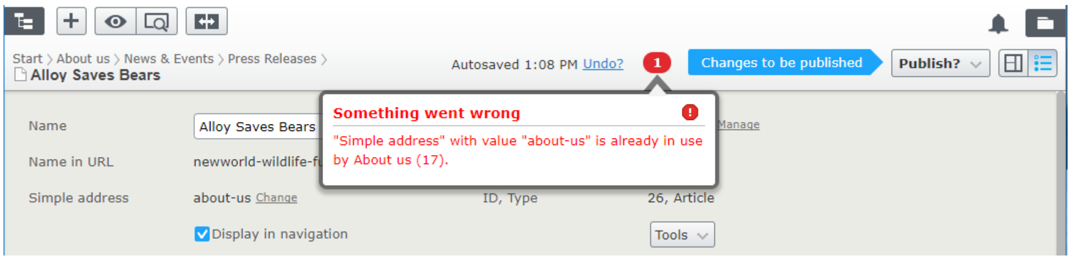
localhost:57762/bears

- **Simple address** is culture specific, e.g. **bears** in English, **björnar** in Swedish.
- **Simple address** stays constant if the page is moved within the tree. **Friendly URL** changes if the page is moved within the tree.

Episerver

Name: Alloy Saves Bears
Name in URL: newworld-wildlife-fund-cho... Change
Simple address: bears Change
Display in navigation

An error is shown if you use a simple address that conflicts with an existing page:



Start › About us › News & Events › Press Releases ›
Alloy Saves Bears

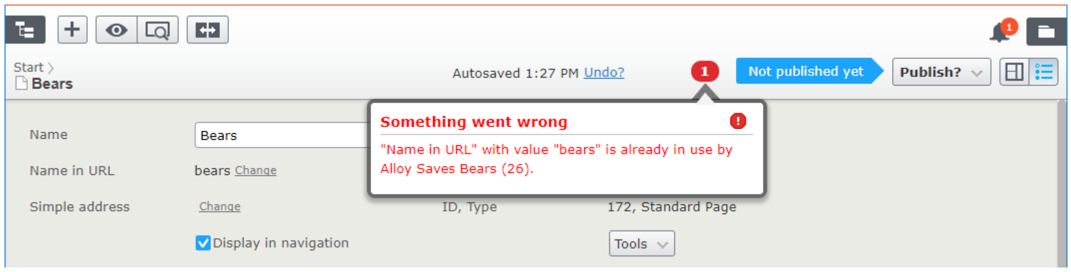
Autosaved 1:08 PM Undo? | Changes to be published | Publish?

Something went wrong

"Simple address" with value "about-us" is already in use by About us (17).

Name: Alloy Saves Bears
Name in URL: newworld-wildlife-f
Simple address: about-us Change
ID, Type: 26, Article
Display in navigation
Tools

An error is shown if you use a segment name that conflicts with a simple address of an existing page:



Start ›
Bears

Autosaved 1:27 PM Undo? | Not published yet | Publish?

Something went wrong

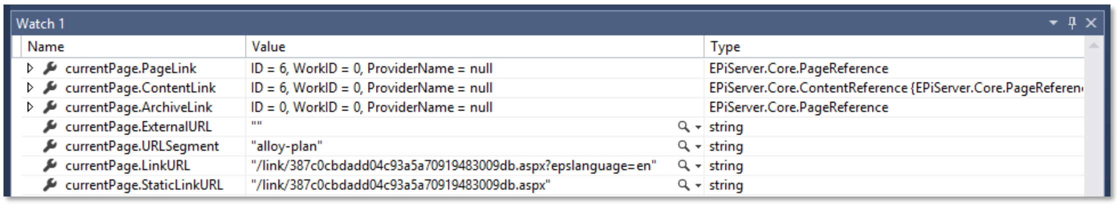
"Name in URL" with value "bears" is already in use by Alloy Saves Bears (26).

Name: Bears
Name in URL: bears Change
Simple address: Change
ID, Type: 172, Standard Page
Display in navigation
Tools

Understanding links and URLs

What is the difference between the following properties of `PageData`?
Note that some use `Link` as a suffix and some use the acronym `URL` in their names:

- `ArchiveLink`, `PageLink`, `ContentLink`, `ParentLink`
- `ExternalURL`, `LinkURL`, `StaticLinkURL`, `URLSegment`

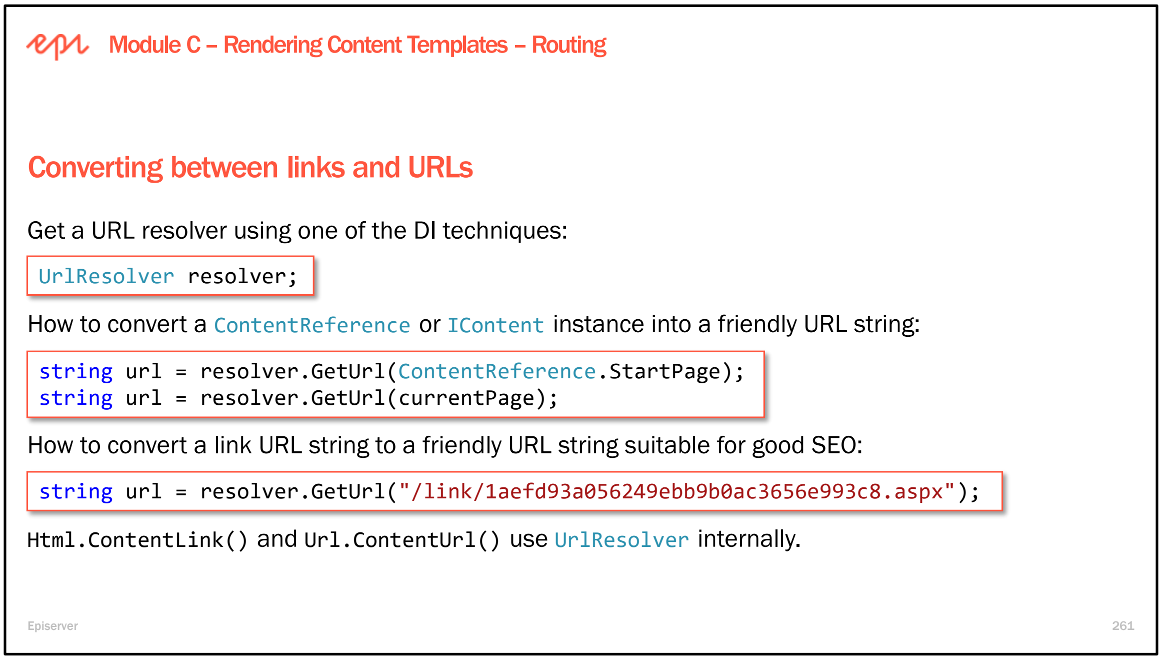

Watch 1

Name	Value	Type
currentPage.PageLink	ID = 6, WorkID = 0, ProviderName = null	EPiServer.Core.PageReference
currentPage.ContentLink	ID = 6, WorkID = 0, ProviderName = null	EPiServer.Core.ContentReference {EPiServer.Core.PageReferen
currentPage.ArchiveLink	ID = 0, WorkID = 0, ProviderName = null	EPiServer.Core.PageReference
currentPage.ExternalURL	""	string
currentPage.URLSegment	"alloy-plan"	string
currentPage.LinkURL	"/link/387c0cbdadd04c93a5a70919483009db.aspx?epslanguage=en"	string
currentPage.StaticLinkURL	"/link/387c0cbdadd04c93a5a70919483009db.aspx"	string

`ExternalURL` is the optional **simple address**; `URLSegment` is one part of the **friendly URL**.

Episerver

260



Converting between links and URLs

Get a URL resolver using one of the DI techniques:

```
UrlResolver resolver;
```

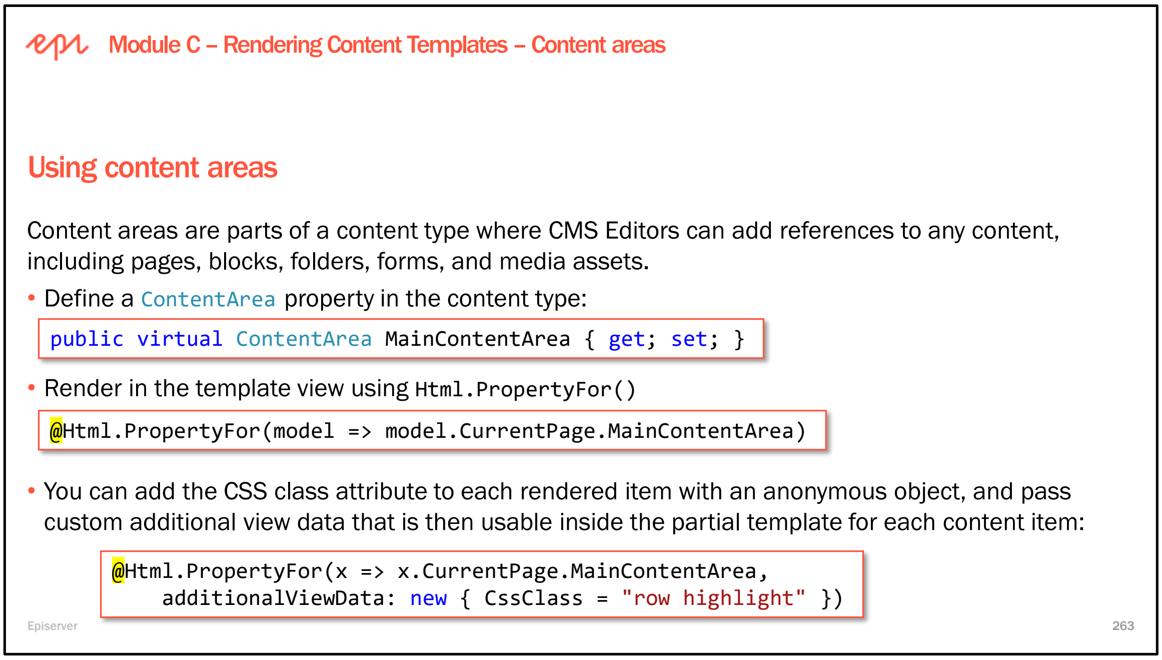
How to convert a `ContentReference` or `IContent` instance into a friendly URL string:

```
string url = resolver.GetUrl(ContentReference.StartPage);
string url = resolver.GetUrl(currentPage);
```

How to convert a link URL string to a friendly URL string suitable for good SEO:

```
string url = resolver.GetUrl("/link/1aefd93a056249ebb9b0ac3656e993c8.aspx");
```

`Html.ContentLink()` and `Url.ContentUrl()` use `UrlResolver` internally.



Module C – Rendering Content Templates – Content areas

Using content areas

Content areas are parts of a content type where CMS Editors can add references to any content, including pages, blocks, folders, forms, and media assets.

- Define a `ContentArea` property in the content type:

```
public virtual ContentArea MainContentArea { get; set; }
```

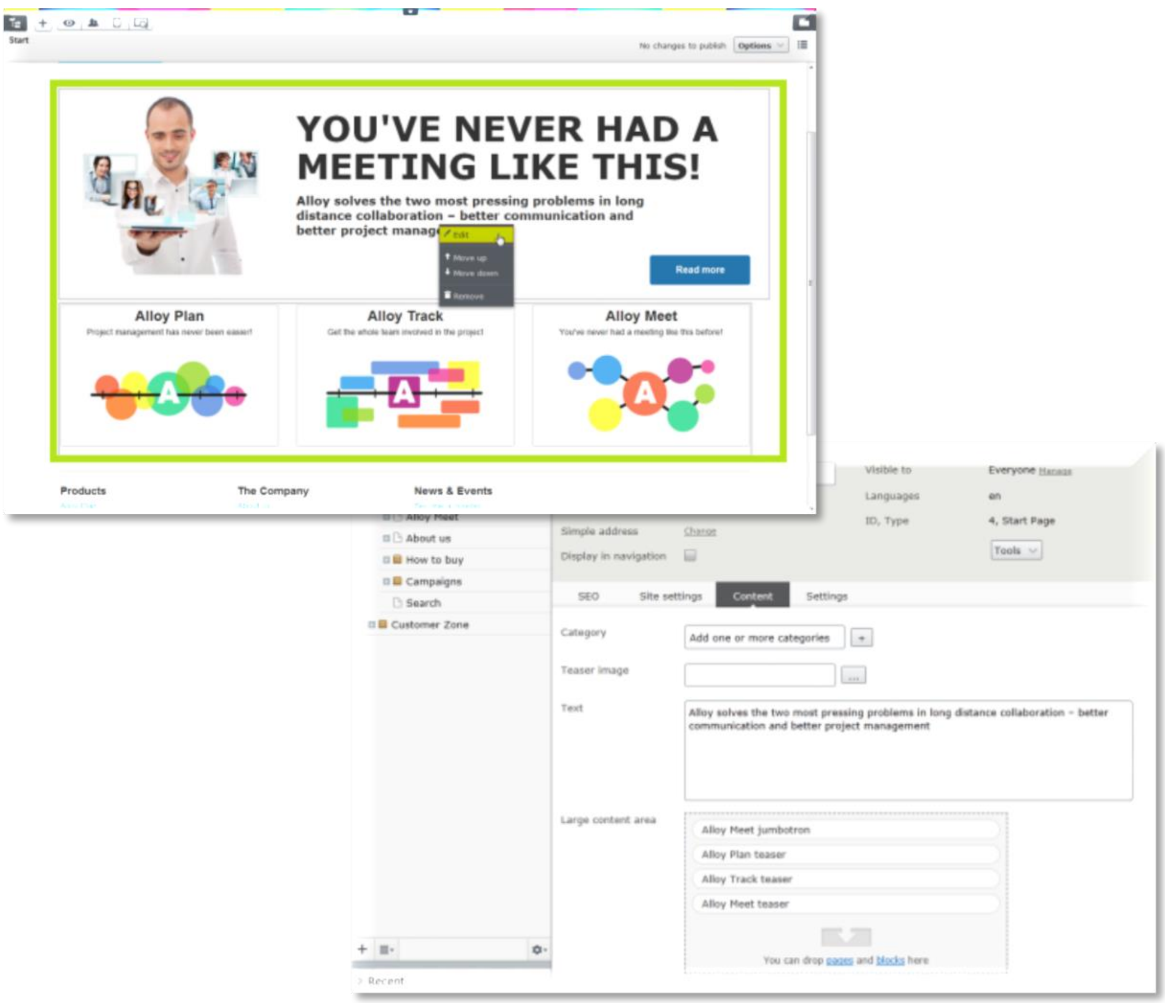
- Render in the template view using `Html.PropertyFor()`

```
@Html.PropertyFor(model => model.CurrentPage.MainContentArea)
```

- You can add the CSS class attribute to each rendered item with an anonymous object, and pass custom additional view data that is then usable inside the partial template for each content item:

```
@Html.PropertyFor(x => x.CurrentPage.MainContentArea,
    additionalViewData: new { CssClass = "row highlight" })
```

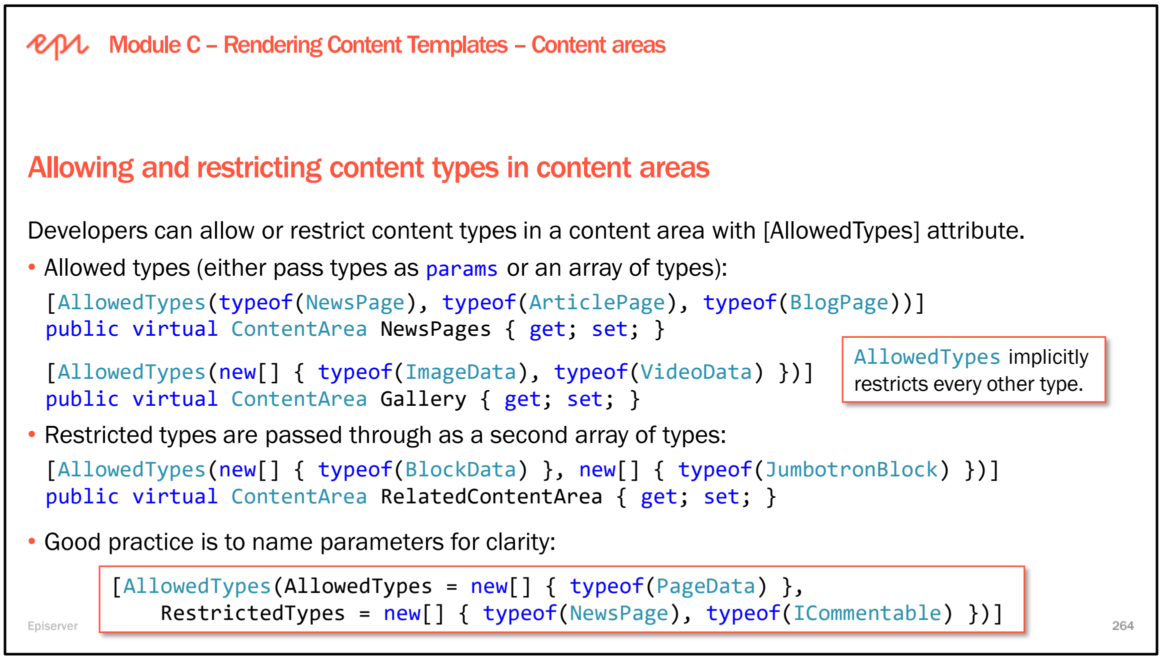
Episerver 263


YOU'VE NEVER HAD A MEETING LIKE THIS!
Alloy solves the two most pressing problems in long distance collaboration – better communication and better project manag
Edit
Move up
Move down
Remove
Read more
Alloy Plan
Project management has never been easier!
Alloy Track
Get the whole team involved in the project
Alloy Meet
You've never had a meeting like this before!
Products
The Company
News & Events
No changes to publish
Options
Start
About us
How to buy
Campaigns
Search
Customer Zone
Recent
SEO
Site settings
Content
Settings
Tools

Visible to: Everyone Manage
Languages: en
ID, Type: 4, Start Page
Simple address: Change
Display in navigation

Category: Add one or more categories
Teaser image
Text: Alloy solves the two most pressing problems in long distance collaboration – better communication and better project management
Large content area:
- Alloy Meet jumbotron
- Alloy Plan teaser
- Alloy Track teaser
- Alloy Meet teaser

You can drop pages and blocks here



Module C – Rendering Content Templates – Content areas

Allowing and restricting content types in content areas

Developers can allow or restrict content types in a content area with [AllowedTypes] attribute.

- Allowed types (either pass types as `params` or an array of types):

```
[AllowedTypes(typeof(NewsPage), typeof(ArticlePage), typeof(BlogPage))]
public virtual ContentArea NewsPages { get; set; }

[AllowedTypes(new[] { typeof(ImageData), typeof(VideoData) })]
public virtual ContentArea Gallery { get; set; }
```

`AllowedTypes` implicitly restricts every other type.

- Restricted types are passed through as a second array of types:

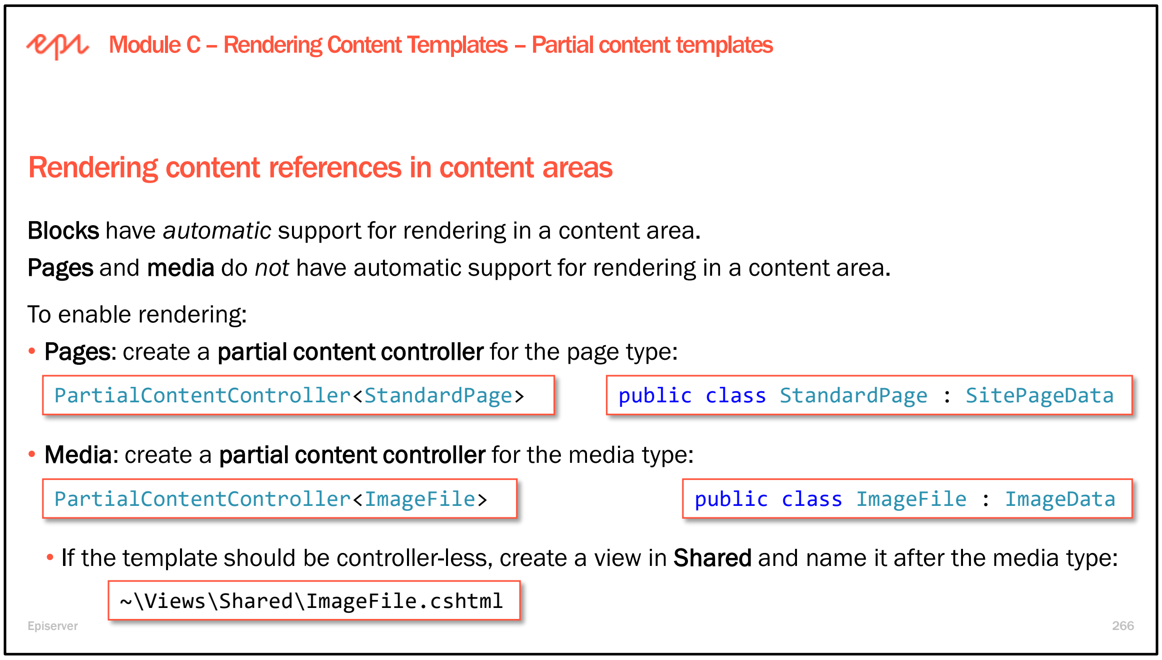
```
[AllowedTypes(new[] { typeof(BlockData) }, new[] { typeof(JumbotronBlock) })]
public virtual ContentArea RelatedContentArea { get; set; }
```

- Good practice is to name parameters for clarity:

```
[AllowedTypes(AllowedTypes = new[] { typeof(PageData) },
    RestrictedTypes = new[] { typeof(NewsPage), typeof(ICommentable) })]
```

Episerver

264



Module C – Rendering Content Templates – Partial content templates

Rendering content references in content areas

Blocks have *automatic* support for rendering in a content area.
Pages and **media** do *not* have automatic support for rendering in a content area.

To enable rendering:

- **Pages**: create a **partial content controller** for the page type:

```
PartialContentController<StandardPage>
```

```
public class StandardPage : SitePageData
```

- **Media**: create a **partial content controller** for the media type:

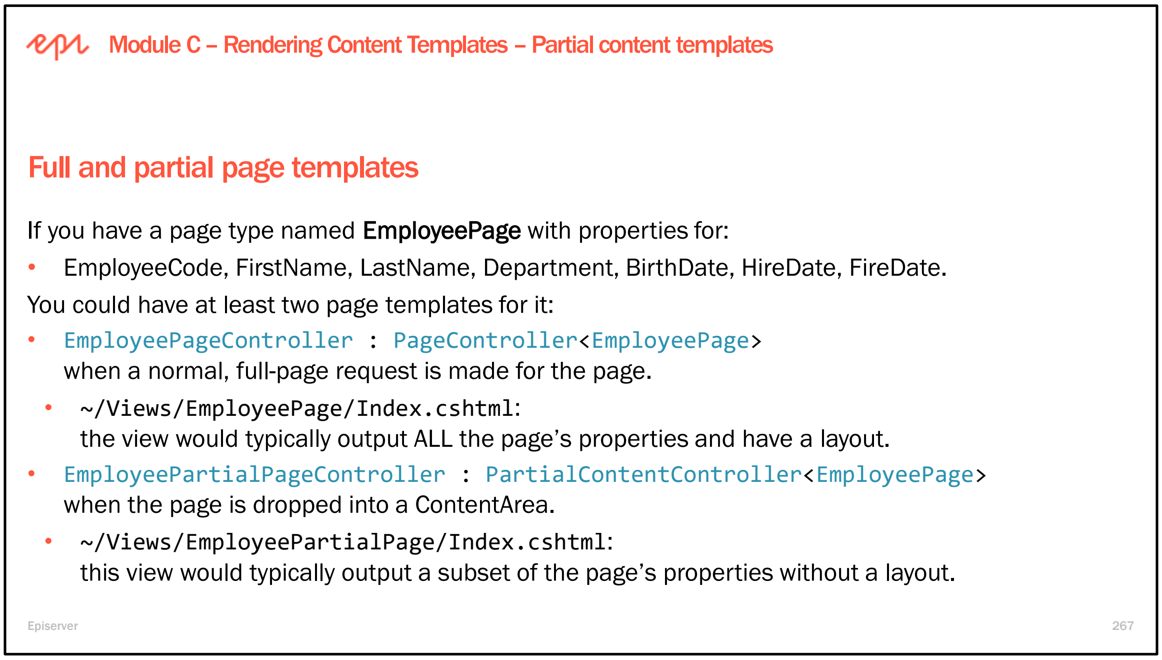
```
PartialContentController<ImageFile>
```

```
public class ImageFile : ImageData
```

  - If the template should be controller-less, create a view in **Shared** and name it after the media type:

```
~\Views\Shared\ImageFile.cshtml
```

Episerver 266



Module C – Rendering Content Templates – Partial content templates

Full and partial page templates

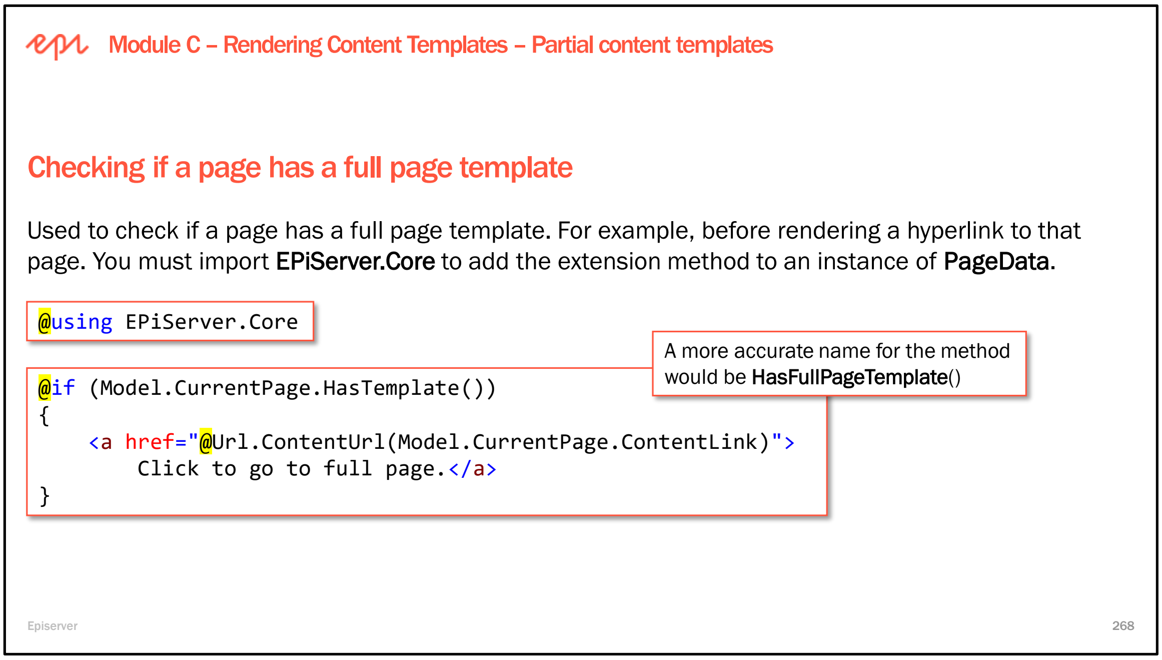
If you have a page type named **EmployeePage** with properties for:

- EmployeeCode, FirstName, LastName, Department, BirthDate, HireDate, FireDate.

You could have at least two page templates for it:

- `EmployeePageController : PageController<EmployeePage>`
  when a normal, full-page request is made for the page.
  - `~/Views/EmployeePage/Index.cshtml`:
    the view would typically output ALL the page's properties and have a layout.
- `EmployeePartialPageController : PartialContentController<EmployeePage>`
  when the page is dropped into a ContentArea.
  - `~/Views/EmployeePartialPage/Index.cshtml`:
    this view would typically output a subset of the page's properties without a layout.

Episerver 267



Module C – Rendering Content Templates – Partial content templates

Checking if a page has a full page template

Used to check if a page has a full page template. For example, before rendering a hyperlink to that page. You must import **EPiServer.Core** to add the extension method to an instance of **PageData.**

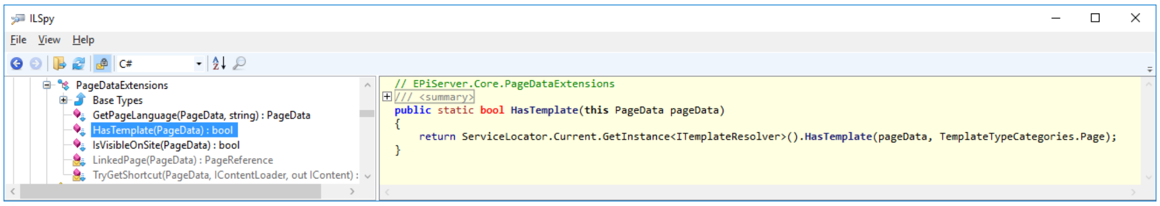
```
@using EPiServer.Core
```

```
@if (Model.CurrentPage.HasTemplate())
{
    <a href="@Url.ContentUrl(Model.CurrentPage.ContentLink)">
        Click to go to full page.</a>
}
```

A more accurate name for the method would be **HasFullPageTemplate**()

Episerver 268

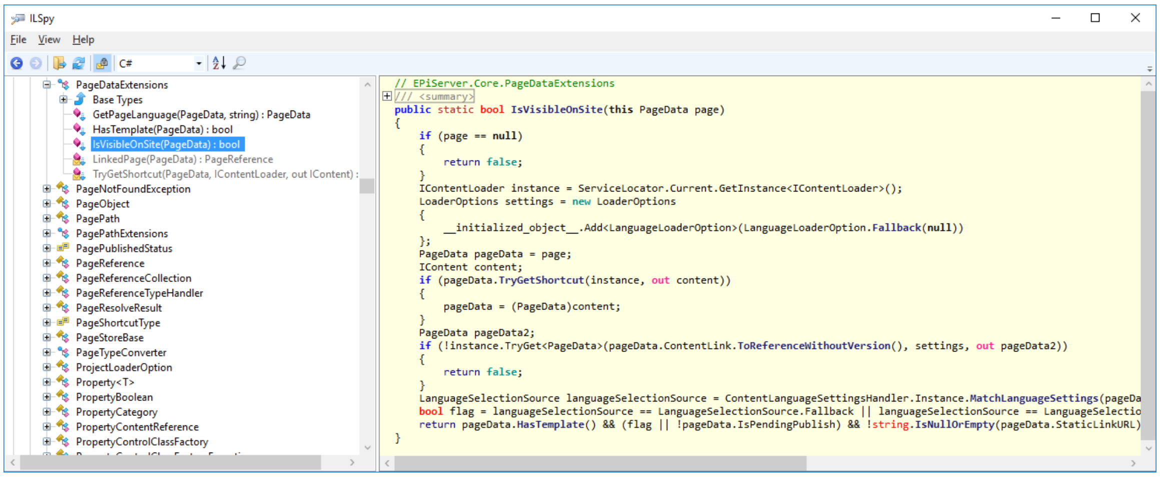
Using ILSpy to reveal how the HasTemplate() extension method for PageData works:



PageDataExtensions
Base Types
GetPageLanguage(PageData, string) : PageData
HasTemplate(PageData) : bool
IsVisibleOnSite(PageData) : bool
LinkedPage(PageData) : PageReference
TryGetShortcut(PageData, IContentLoader, out IContent) :

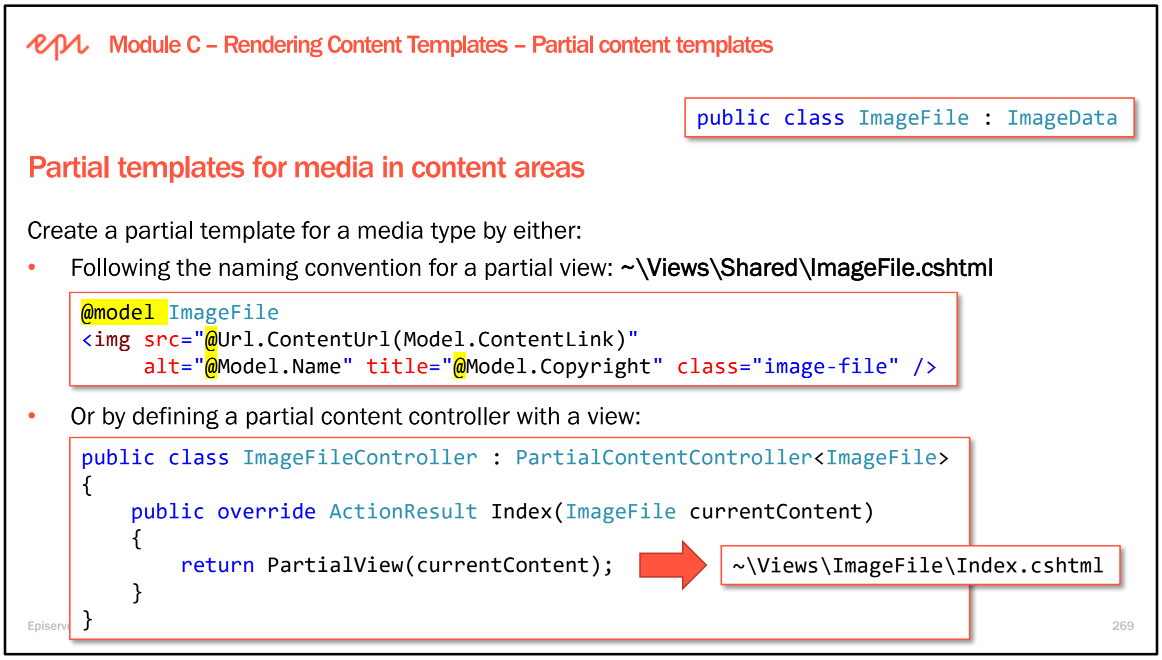
```
// EPiServer.Core.PageDataExtensions
/// <summary>
public static bool HasTemplate(this PageData pageData)
{
    return ServiceLocator.Current.GetInstance<ITemplateResolver>().HasTemplate(pageData, TemplateTypeCategories.Page);
}
```

Using ILSpy to reveal how the IsVisibleOnSite() extension method for PageData works:



PageDataExtensions
Base Types
GetPageLanguage(PageData, string) : PageData
HasTemplate(PageData) : bool
IsVisibleOnSite(PageData) : bool
LinkedPage(PageData) : PageReference
TryGetShortcut(PageData, IContentLoader, out IContent) :
PageNotFoundException
PageObject
PagePath
PagePathExtensions
PagePublishedStatus
PageReference
PageReferenceCollection
PageReferenceTypeHandler
PageResolveResult
PageShortcutType
PageStoreBase
PageTypeConverter
ProjectLoaderOption
Property<T>
PropertyBoolean
PropertyCategory
PropertyContentReference
PropertyControlClassFactory

```
// EPiServer.Core.PageDataExtensions
/// <summary>
public static bool IsVisibleOnSite(this PageData page)
{
    if (page == null)
    {
        return false;
    }
    IContentLoader instance = ServiceLocator.Current.GetInstance<IContentLoader>();
    LoaderOptions settings = new LoaderOptions
    {
        __initialized_object__.Add<LanguageLoaderOption>(LanguageLoaderOption.Fallback(null))
    };
    PageData pageData = page;
    IContent content;
    if (pageData.TryGetShortcut(instance, out content))
    {
        pageData = (PageData)content;
    }
    PageData pageData2;
    if (!instance.TryGet<PageData>(pageData.ContentLink.ToReferenceWithoutVersion(), settings, out pageData2))
    {
        return false;
    }
    LanguageSelectionSource languageSelectionSource = ContentLanguageSettingsHandler.Instance.MatchLanguageSettings(pageDa
    bool flag = languageSelectionSource == LanguageSelectionSource.Fallback || languageSelectionSource == LanguageSelectio
    return pageData.HasTemplate() && (flag || !pageData.IsPendingPublish) && !string.IsNullOrEmpty(pageData.StaticLinkURL)
}
```



Module C – Rendering Content Templates – Partial content templates

```
public class ImageFile : ImageData
```

Partial templates for media in content areas

Create a partial template for a media type by either:

- Following the naming convention for a partial view: **~\Views\Shared\ImageFile.cshtml**

```
@model ImageFile
<img src="@Url.ContentUrl(Model.ContentLink)"
     alt="@Model.Name" title="@Model.Copyright" class="image-file" />
```

- Or by defining a partial content controller with a view:

```
public class ImageFileController : PartialContentController<ImageFile>
{
    public override ActionResult Index(ImageFile currentContent)
    {
        return PartialView(currentContent);
    }
}
```

```
~\Views\ImageFile\Index.cshtml
```

Episerv

269

Example media asset controller with a view model

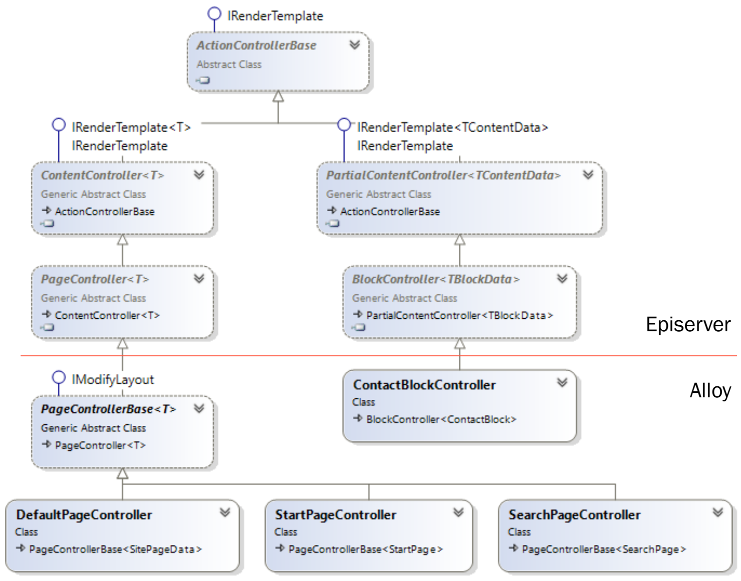
```
using System.Web.Mvc;
using AlloyDemo.Models.Media;
using AlloyDemo.Models.ViewModels;
using EPiServer.Web.Mvc;
using EPiServer.Web.Routing;

namespace AlloyDemo.Controllers
{
    public class ImageFileController : PartialContentController<ImageFile>
    {
        private readonly UrlResolver urlResolver;

        public ImageFileController(UrlResolver urlResolver)
        {
            this.urlResolver = urlResolver;
        }

        public override ActionResult Index(ImageFile currentContent)
        {
            var model = new ImageViewModel
            {
                Url = urlResolver.GetUrl(currentContent.ContentLink),
                Name = currentContent.Name,
                Copyright = currentContent.Copyright
            };

            return PartialView(model);
        }
    }
}
```

Module C – Rendering Content Templates – Partial content templates

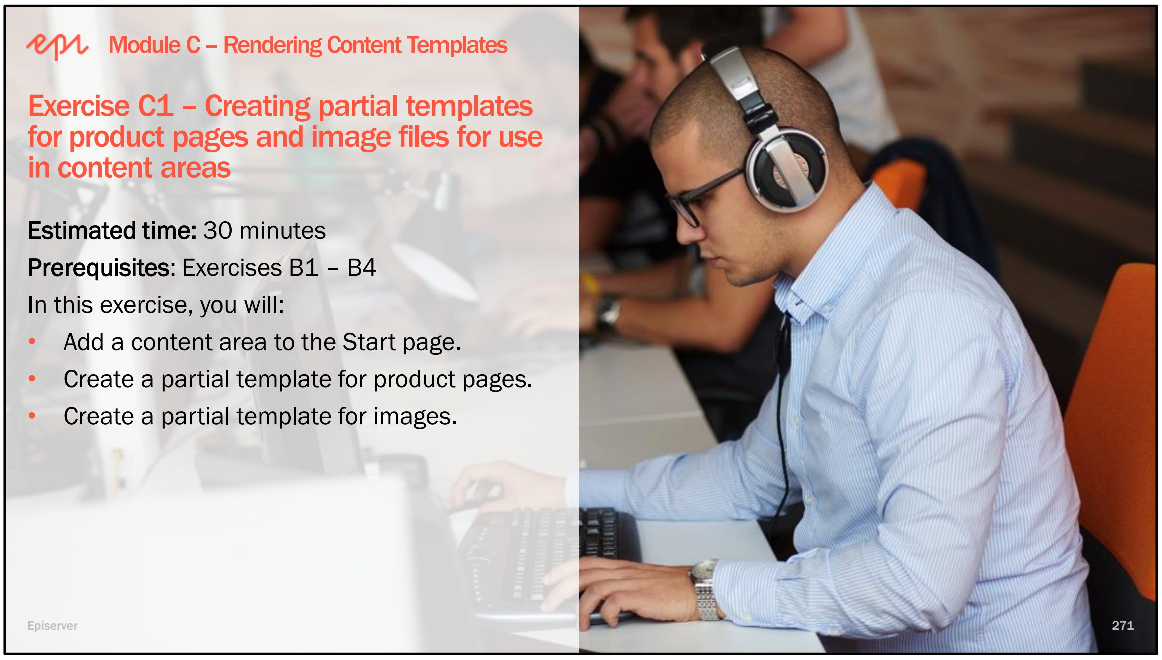
Rendering templates

Episerver CMS's template resolver looks for any class that implements **IRenderTemplate<T>** where **T** is a content type.

In practice, you will usually create a class that derives from **PageController<T>** , **PartialContentController<T>**, or **BlockController<T>** as in this example hierarchy in Alloy.



Episerver



Module C – Rendering Content Templates

Exercise C1 – Creating partial templates for product pages and image files for use in content areas

Estimated time: 30 minutes

Prerequisites: Exercises B1 – B4

In this exercise, you will:

- Add a content area to the Start page.
- Create a partial template for product pages.
- Create a partial template for images.

Episerver 271

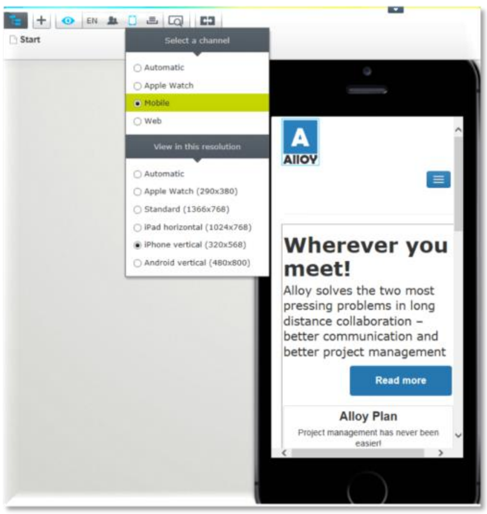
Module C – Rendering Content Templates – Multi-template content types

Multiple templates for a single content type

A content type can have multiple templates.

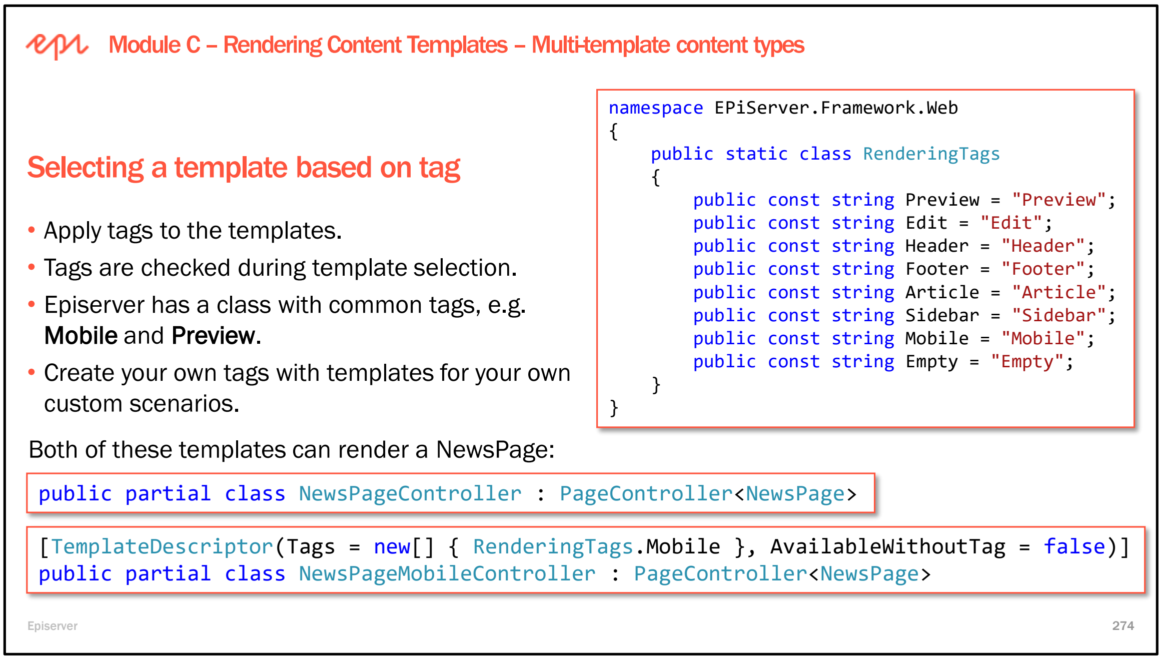
For example:

- A template for desktop browsers, and another for mobile browsers.
- A template for countries with high bandwidth, and another for countries with low bandwidth.
- Templates for different column widths in a layout: Full, Wide (2/3 width), and Narrow (1/3 width).
- A template the returns HTML, and templates that return PDF, Excel, and Word formats.
- Templates that return data formats like JSON and XML and RSS feed formats.


Start
Select a channel
Automatic
Apple Watch
Mobile
Web
View in this resolution
Automatic
Apple Watch (290x380)
Standard (1366x768)
iPad horizontal (1024x768)
iPhone vertical (320x568)
Android vertical (480x800)
Wherever you meet!
Alloy solves the two most pressing problems in long distance collaboration – better communication and better project management
Read more
Alloy Plan
Project management has never been easier!

Episerver 273

Several templates (implemented in either ASP.NET Web Forms or ASP.NET MVC) can be registered for any content type (typically pages or blocks even if the underlying templating system supports any .NET type). By default, there are no channels in an installation and you need to add the channels you desire in your templates. If no matching channel is found no tag will be added when trying to find templates.



Module C – Rendering Content Templates – Multi-template content types

Selecting a template based on tag

- Apply tags to the templates.
- Tags are checked during template selection.
- Episerver has a class with common tags, e.g. **Mobile** and **Preview**.
- Create your own tags with templates for your own custom scenarios.

```
namespace EPiServer.Framework.Web
{
    public static class RenderingTags
    {
        public const string Preview = "Preview";
        public const string Edit = "Edit";
        public const string Header = "Header";
        public const string Footer = "Footer";
        public const string Article = "Article";
        public const string Sidebar = "Sidebar";
        public const string Mobile = "Mobile";
        public const string Empty = "Empty";
    }
}
```

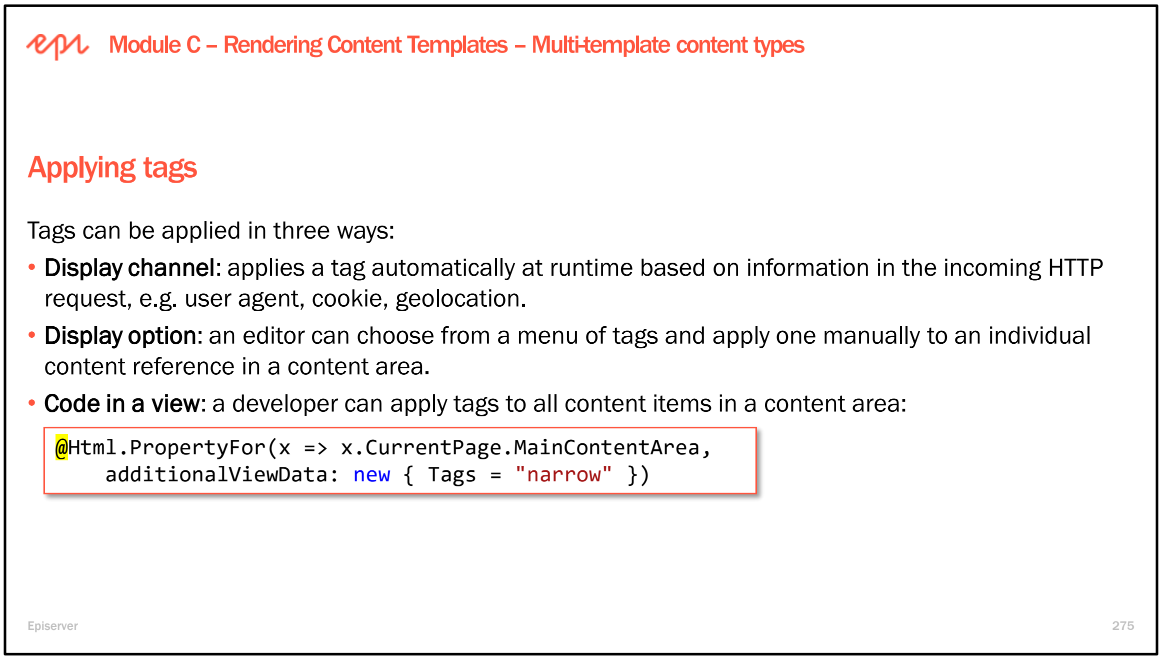
Both of these templates can render a NewsPage:

```
public partial class NewsPageController : PageController<NewsPage>
```

```
[TemplateDescriptor(Tags = new[] { RenderingTags.Mobile }, AvailableWithoutTag = false)]
public partial class NewsPageMobileController : PageController<NewsPage>
```

Episerver 274

By default when a template renderer is associated with a Tag then that renderer will only be available when the calling context (given by for example Property, ContentControl, PropertyFor) has a matching tag. By setting the attribute AvailableWithoutTag to true on your template the template will be available also when calling context has no tag specified.

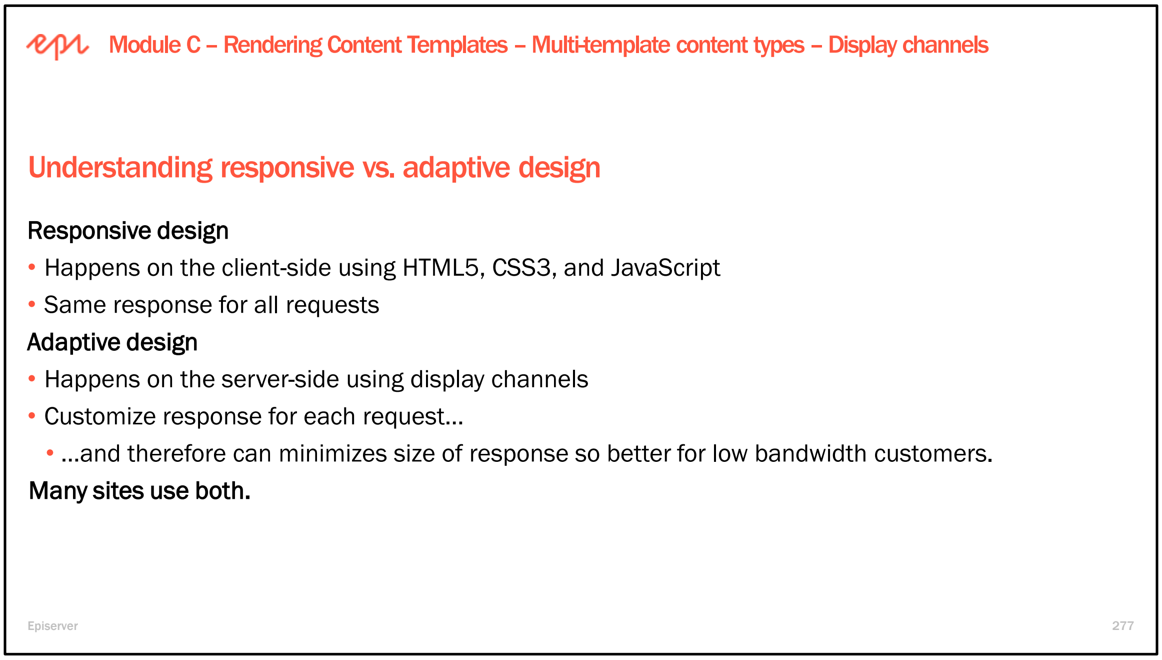


Applying tags

Tags can be applied in three ways:

- **Display channel**: applies a tag automatically at runtime based on information in the incoming HTTP request, e.g. user agent, cookie, geolocation.
- **Display option**: an editor can choose from a menu of tags and apply one manually to an individual content reference in a content area.
- **Code in a view**: a developer can apply tags to all content items in a content area:

```
@Html.PropertyFor(x => x.CurrentPage.MainContentArea,
    additionalViewData: new { Tags = "narrow" })
```



Module C – Rendering Content Templates – Multi-template content types – Display channels

Understanding responsive vs. adaptive design

Responsive design

- Happens on the client-side using HTML5, CSS3, and JavaScript
- Same response for all requests

Adaptive design

- Happens on the server-side using display channels
- Customize response for each request…
  - …and therefore can minimizes size of response so better for low bandwidth customers.

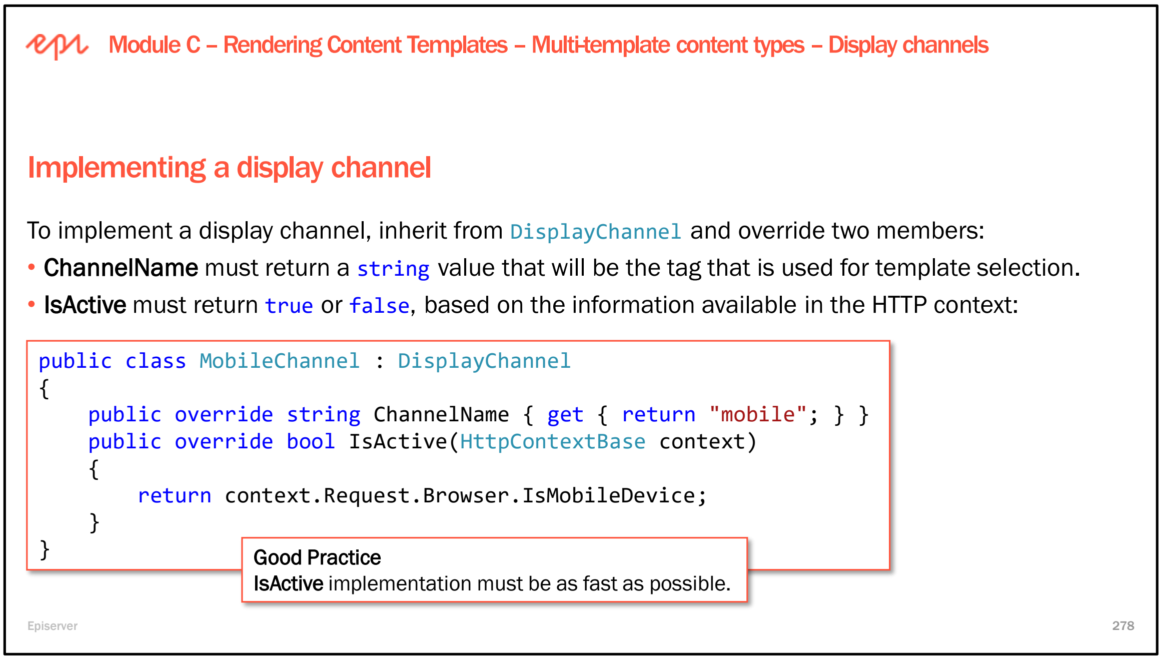
Many sites use both.

Episerver 277

Display channels is a way to control the rendering of content depending on the request:

- Several templates, control which template to use
- Single template, control output depending on channel

Editors can preview display channels when editing content.



Module C – Rendering Content Templates – Multi-template content types – Display channels

Implementing a display channel

To implement a display channel, inherit from `DisplayChannel` and override two members:

- **ChannelName** must return a `string` value that will be the tag that is used for template selection.
- **IsActive** must return `true` or `false`, based on the information available in the HTTP context:

```
public class MobileChannel : DisplayChannel
{
    public override string ChannelName { get { return "mobile"; } }
    public override bool IsActive(HttpContextBase context)
    {
        return context.Request.Browser.IsMobileDevice;
    }
}
```

Good Practice
IsActive implementation must be as fast as possible.

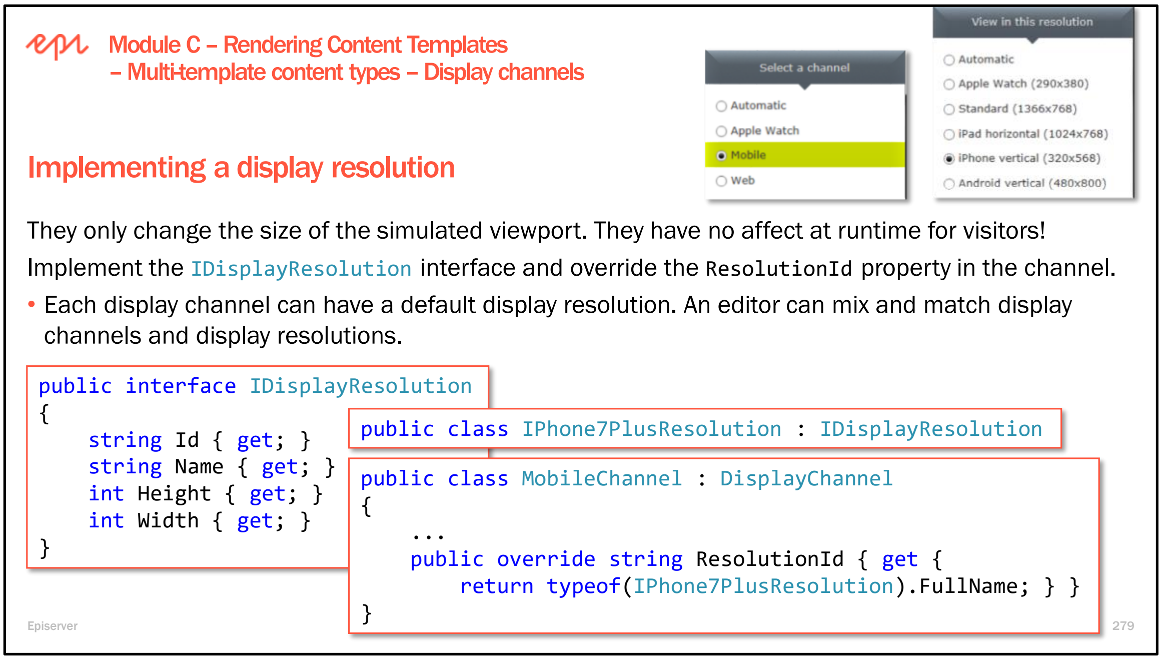
Episerver 278

To create a Display Channel, create a class that inherits from EPiServer.Web.DisplayChannel.

The system will scan and register all found channel instances during initialization. There is no need to explicitly register the channel.

Both of the below templates will be registered as templates for a page type called NewsPage.
If the Mobile channel (above example) is active, templates with tag “Mobile” will be preferred.

```
public class NewsPageController : PageController<NewsPage> {}

[TemplateDescriptor(Tags = new[] { “Mobile” })]
public class NewsMobileController : PageController<NewsPage> {}
```



Module C – Rendering Content Templates
– Multi-template content types – Display channels

Select a channel

- Automatic
- Apple Watch
- [✓ Mobile]
- Web

View in this resolution

- Automatic
- Apple Watch (290x380)
- Standard (1366x768)
- iPad horizontal (1024x768)
- [✓ iPhone vertical (320x568)]
- Android vertical (480x800)

Implementing a display resolution

They only change the size of the simulated viewport. They have no affect at runtime for visitors!

Implement the `IDisplayResolution` interface and override the `ResolutionId` property in the channel.

- Each display channel can have a default display resolution. An editor can mix and match display channels and display resolutions.

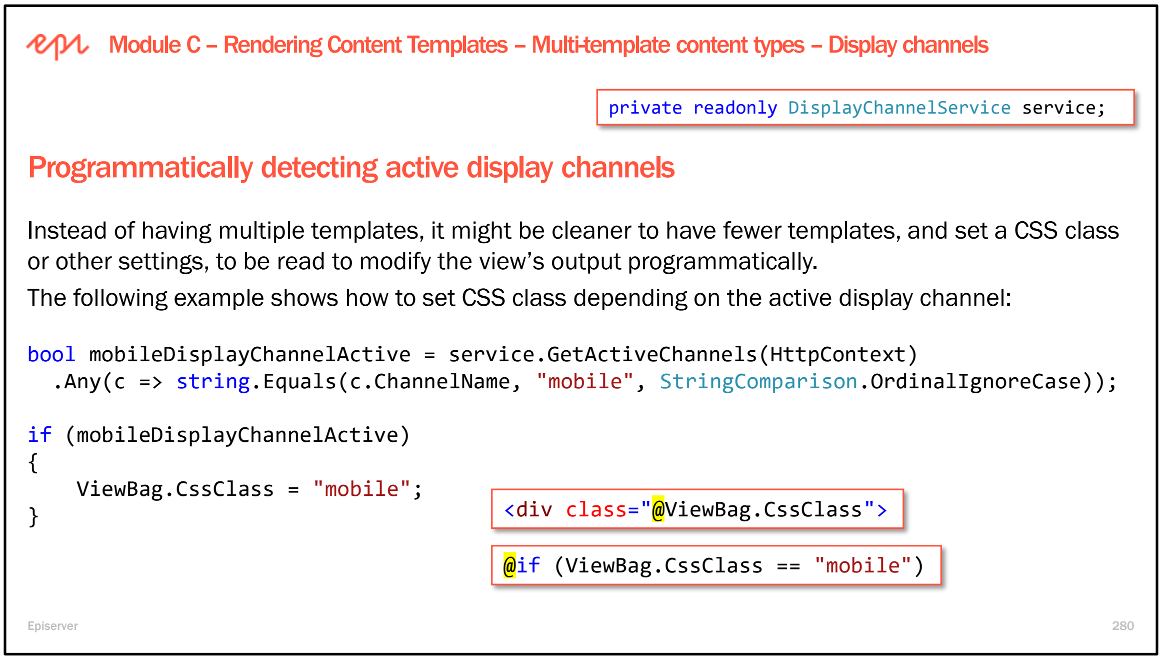
```
public interface IDisplayResolution
{
    string Id { get; }
    string Name { get; }
    int Height { get; }
    int Width { get; }
}
```

```
public class IPhone7PlusResolution : IDisplayResolution
```

```
public class MobileChannel : DisplayChannel
{
    ...
    public override string ResolutionId { get {
        return typeof(IPhone7PlusResolution).FullName; } }
}
```

Episerver 279

http://www.quirksmode.org/m/tests/widthtest_vpdevice.html



Module C – Rendering Content Templates – Multi-template content types – Display channels

```
private readonly DisplayChannelService service;
```

Programmatically detecting active display channels

Instead of having multiple templates, it might be cleaner to have fewer templates, and set a CSS class or other settings, to be read to modify the view's output programmatically.

The following example shows how to set CSS class depending on the active display channel:

```
bool mobileDisplayChannelActive = service.GetActiveChannels(HttpContext)
  .Any(c => string.Equals(c.ChannelName, "mobile", StringComparison.OrdinalIgnoreCase));

if (mobileDisplayChannelActive)
{
    ViewBag.CssClass = "mobile";
}
```

```
<div class="@ViewBag.CssClass">
```

```
@if (ViewBag.CssClass == "mobile")
```

Episerver 280

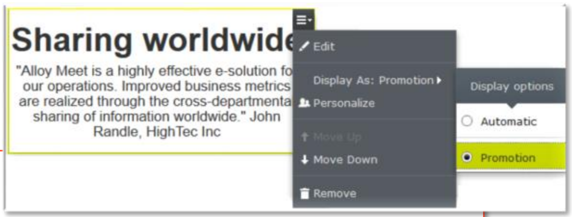
Module C – Rendering Content Templates – Multi-template content types – Display options and tags

Applying tags to content areas using display options

In an initialization module, get the DisplayOptions service using a DI technique, and then call Add for each **Display as:** menu option:

```
DisplayOptions displayOptions;
```

```
displayOptions.Add(
    id: "promo",
    tag: "promo",
    // The name and description properties can also be reference keys to
    // a language resource to allow localization.
    name: "Promotion",
    description: "Promotional content is displayed full width and highlighted.",
    iconClass: "icon-promo");
```



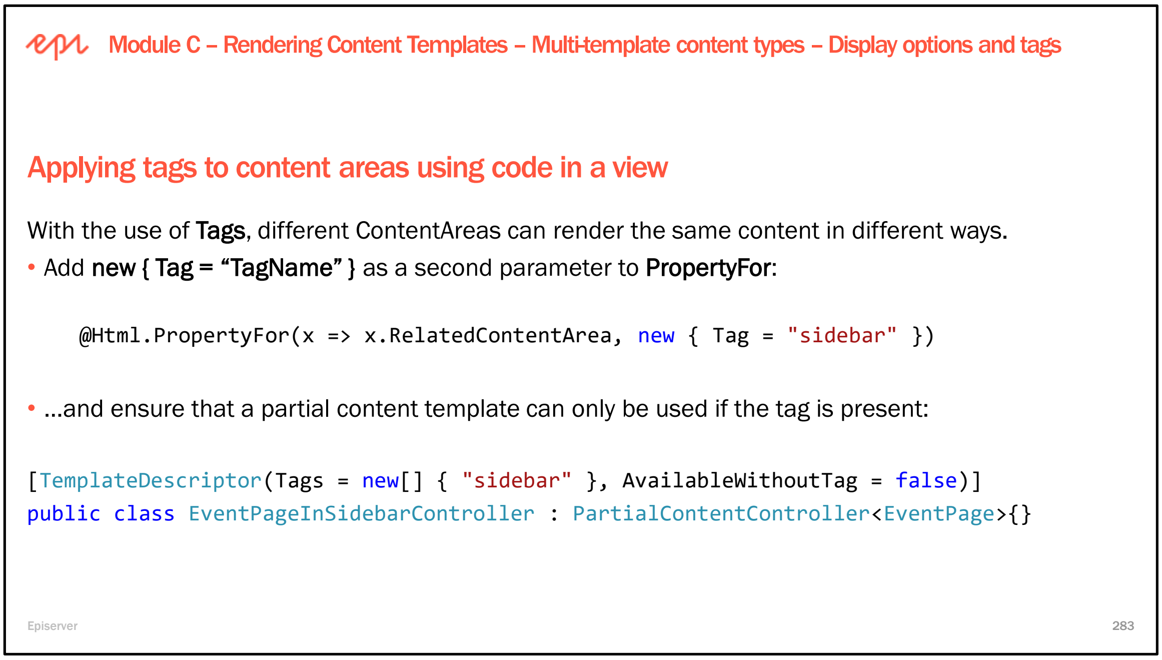
Sharing worldwide

"Alloy Meet is a highly effective e-solution fo our operations. Improved business metrics are realized through the cross-departmenta sharing of information worldwide." John Randle, HighTec Inc

Episerver 282

See the EPiServer.Web.DisplayOptions class for more information and method overloads.

Note: An implementation example is available in the Alloy Sample site.



Module C – Rendering Content Templates – Multi-template content types – Display options and tags

Applying tags to content areas using code in a view

With the use of **Tags**, different ContentAreas can render the same content in different ways.

- Add **new { Tag = "TagName" }** as a second parameter to **PropertyFor**:

```
    @Html.PropertyFor(x => x.RelatedContentArea, new { Tag = "sidebar" })
```

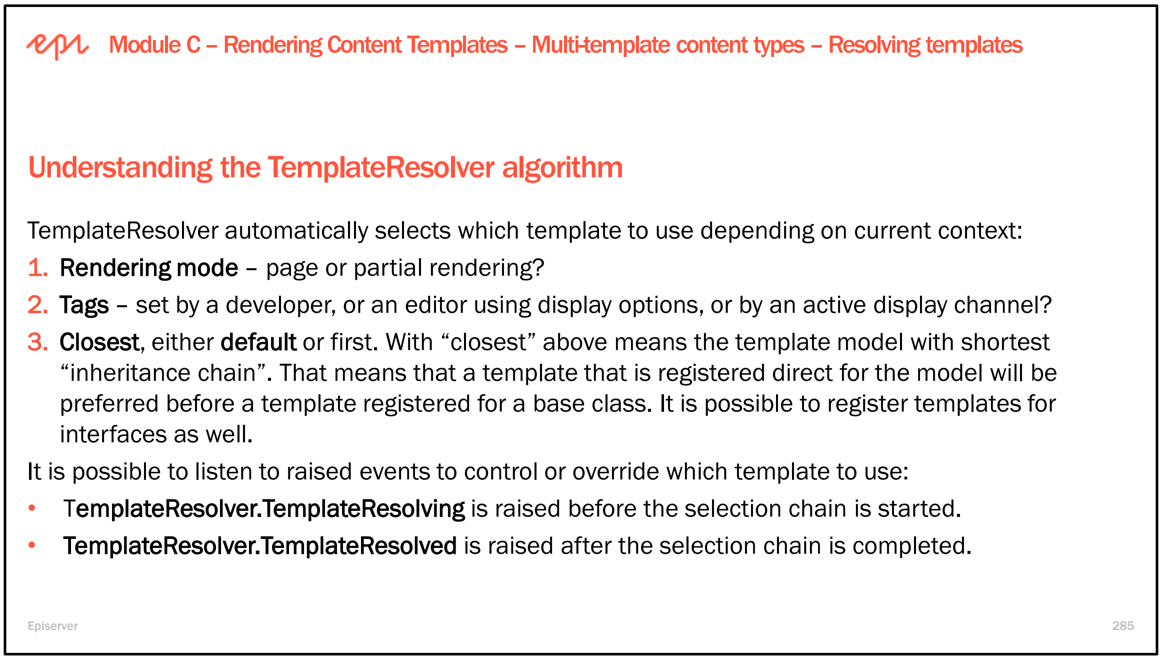
- ...and ensure that a partial content template can only be used if the tag is present:

```
[TemplateDescriptor(Tags = new[] { "sidebar" }, AvailableWithoutTag = false)]
public class EventPageInSidebarController : PartialContentController<EventPage>{}
```

Episerver 283

To see this in action: Look in Alloy templates and add the same block to some different content areas.
You can access the content in a content area from code:

```
foreach (IContent c in myPage.MyContentArea.Contents)
{
    Response.Write("Hello " + c.Name);
}
```



Module C – Rendering Content Templates – Multi-template content types – Resolving templates

Understanding the TemplateResolver algorithm

TemplateResolver automatically selects which template to use depending on current context:

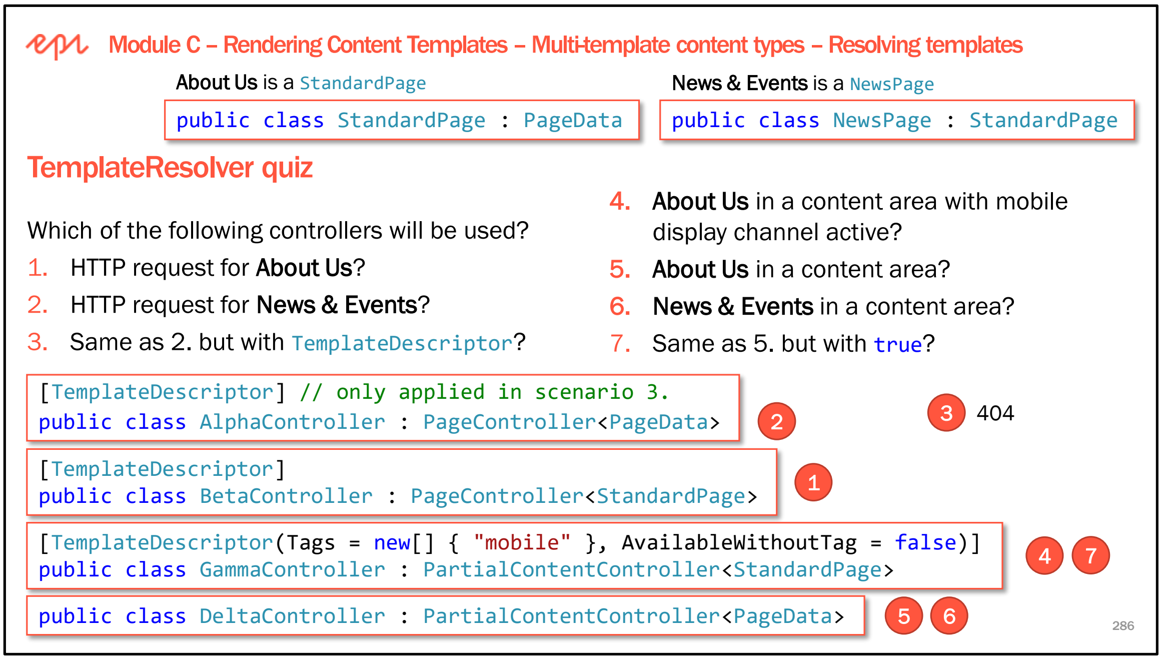
1. **Rendering mode** – page or partial rendering?
2. **Tags** – set by a developer, or an editor using display options, or by an active display channel?
3. **Closest**, either **default** or first. With "closest" above means the template model with shortest "inheritance chain". That means that a template that is registered direct for the model will be preferred before a template registered for a base class. It is possible to register templates for interfaces as well.

It is possible to listen to raised events to control or override which template to use:

- **TemplateResolver.TemplateResolving** is raised before the selection chain is started.
- **TemplateResolver.TemplateResolved** is raised after the selection chain is completed.

Episerver 285

Why would you want to override the template resolver either before or after it has been actioned?
In an Enterprise site, where you can have multiple start pages, same codebase, same page types, but want to have different rendering for _some_ of the page types.
Example: The Start page template is used for 3 of 4 sites, but for one site you need a completely different rendering. It is the same page type though. You can tailor the selector to do this with some custom code, listening to the TemplateResolver event(s).
Sample code: A code example that demonstrates how to exchange the template for mobile requests is available on the TemplateResolver class in the Episerver CMS SDK (look at EPiServer.Web.TemplateResolver).



Module C – Rendering Content Templates – Multi-template content types – Resolving templates

About Us is a `StandardPage`

```
public class StandardPage : PageData
```

News & Events is a `NewsPage`

```
public class NewsPage : StandardPage
```

TemplateResolver quiz

Which of the following controllers will be used?

1. HTTP request for **About Us**?
2. HTTP request for **News & Events**?
3. Same as 2. but with `TemplateDescriptor`?
4. **About Us** in a content area with mobile display channel active?
5. **About Us** in a content area?
6. **News & Events** in a content area?
7. Same as 5. but with `true`?

```
[TemplateDescriptor] // only applied in scenario 3.
public class AlphaController : PageController<PageData>
```

2

3 404

```
[TemplateDescriptor]
public class BetaController : PageController<StandardPage>
```

1

```
[TemplateDescriptor(Tags = new[] { "mobile" }, AvailableWithoutTag = false)]
public class GammaController : PartialContentController<StandardPage>
```

4 7

```
public class DeltaController : PartialContentController<PageData>
```

5 6

286

WITHOUT [TemplateDescriptor], Inherited = true, so AlphaController can be used for NewsPages.

WITH [TemplateDescriptor], Inherited = false, so BetaController cannot be used for NewsPages, even if NewsPage inherits from StandardPage.

GammaController can only be used when mobile tag is applied, for example, when mobile display channel is active, or when mobile display option is applied by editor, or when developer sets tag in view.

Module C – Rendering Content Templates

Exercises C2 to C5 – Handling multiple content templates

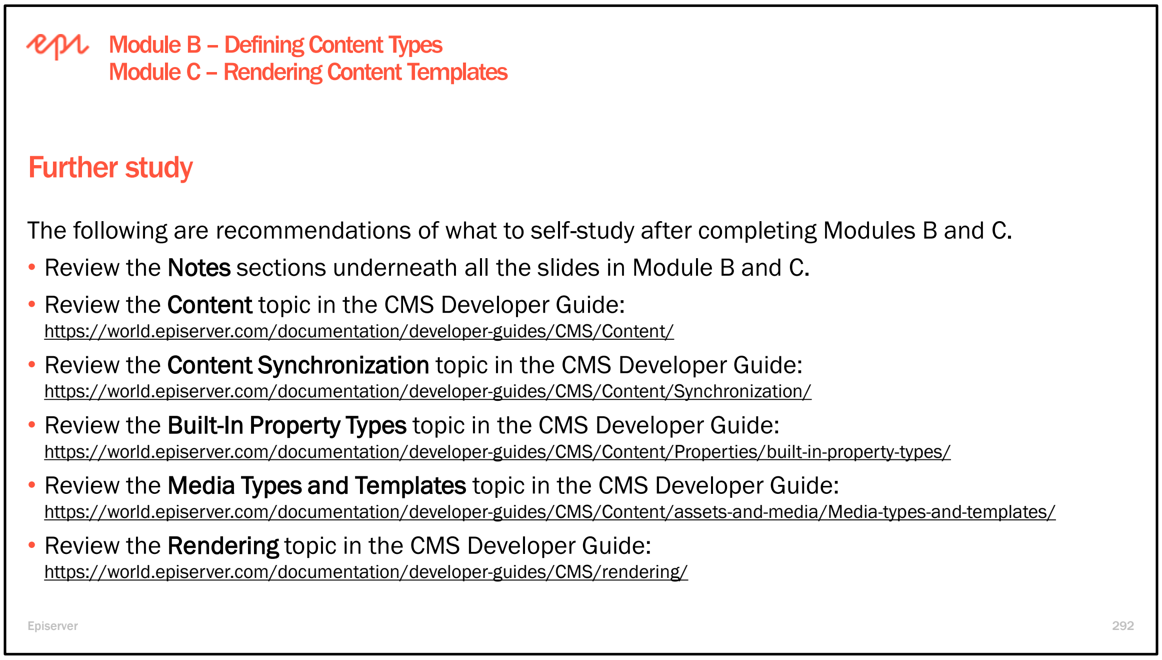
Estimated time: 60 minutes

Prerequisites: Exercises B1 – B4, C1

2. Creating a partial template for all pages
3. Adding display options for content editors
4. Adding tags to content areas programmatically
5. Setting tags using a display channel

Episerver

287



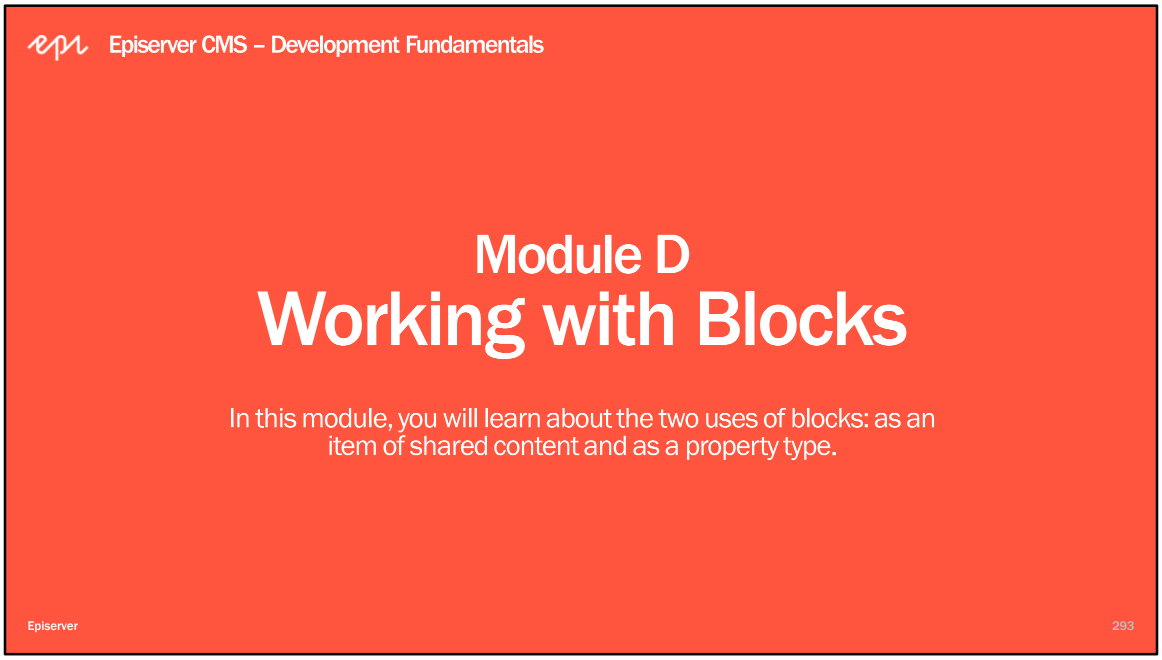
Module B – Defining Content Types
Module C – Rendering Content Templates

Further study

The following are recommendations of what to self-study after completing Modules B and C.

- Review the **Notes** sections underneath all the slides in Module B and C.
- Review the **Content** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/Content/
- Review the **Content Synchronization** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/Content/Synchronization/
- Review the **Built-In Property Types** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/Content/Properties/built-in-property-types/
- Review the **Media Types and Templates** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/Content/assets-and-media/Media-types-and-templates/
- Review the **Rendering** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/rendering/

Episerver 292



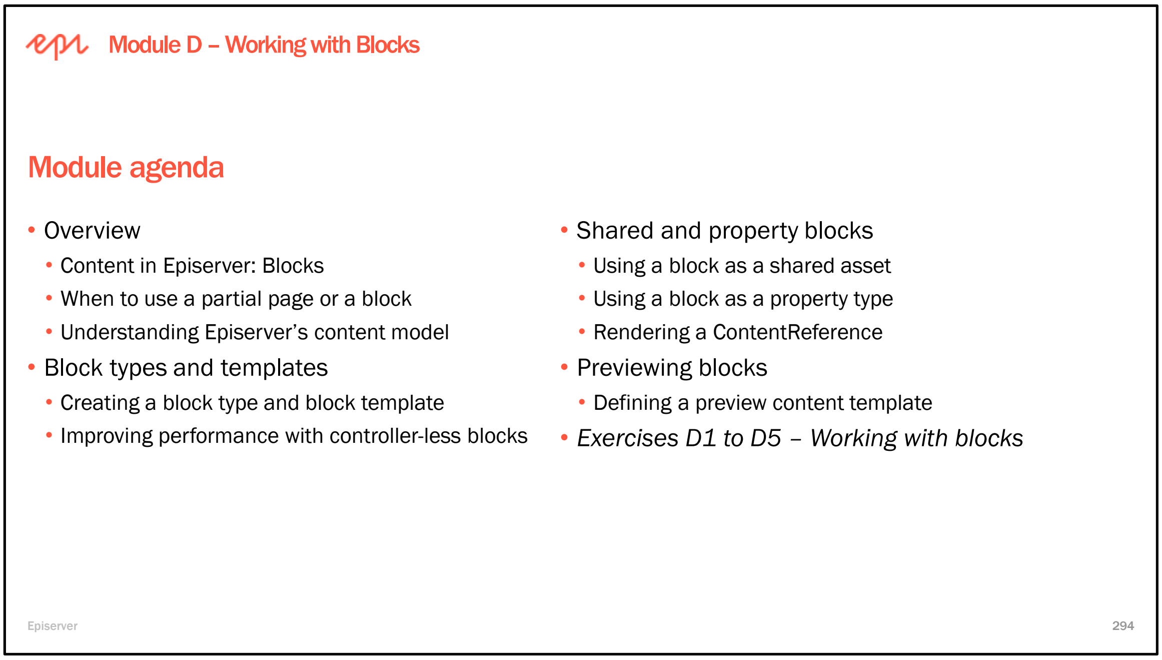
Episerver CMS – Development Fundamentals

Module D
Working with Blocks

In this module, you will learn about the two uses of blocks: as an item of shared content and as a property type.

Episerver

293

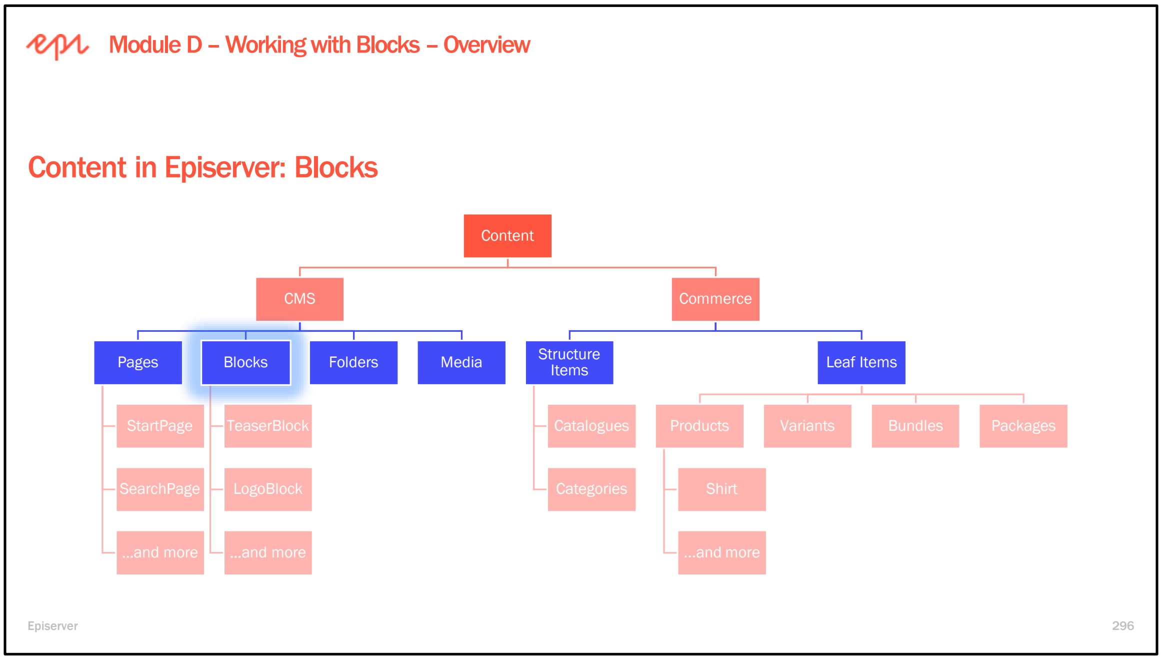


Module D – Working with Blocks

Module agenda

- Overview
  - Content in Episerver: Blocks
  - When to use a partial page or a block
  - Understanding Episerver’s content model
- Block types and templates
  - Creating a block type and block template
  - Improving performance with controller-less blocks
- Shared and property blocks
  - Using a block as a shared asset
  - Using a block as a property type
  - Rendering a ContentReference
- Previewing blocks
  - Defining a preview content template
- *Exercises D1 to D5 – Working with blocks*

Episerver 294

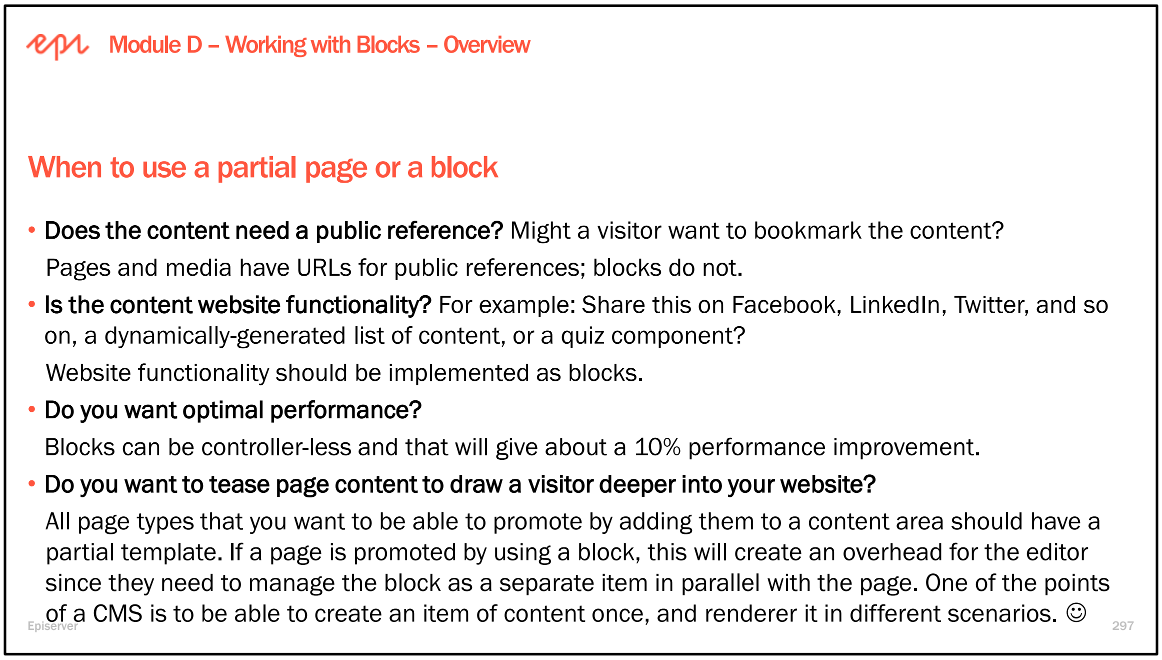


Module D – Working with Blocks – Overview

Content in Episerver: Blocks

Episerver 296

The structure of the website is made up of pages, where the names of the pages automatically form structures and menus. You can create reusable smaller content parts for editing on the pages across your website, called blocks.
Blocks are a group of properties. For example, a form block with heading, main body and a XForm, or an image URL and an image description. With blocks you can add an image URL property and an image description property to a new block type and call it “image”.
When creating and editing blocks you will get a similar experience as when working with pages. You can rearrange the blocks on a page using drag and drop, and remove them. You can also see on which pages each block is used, for example, if you are deleting a block you will be prompted to a dialog that shows you which pages are affected.
A block can be either shared or used as a property. The blocks you can rearrange on a page are “shared”. When using the block as a property on a particular page type, the values are set in the page by the editor just as for other properties, and it can not be dragged and dropped.

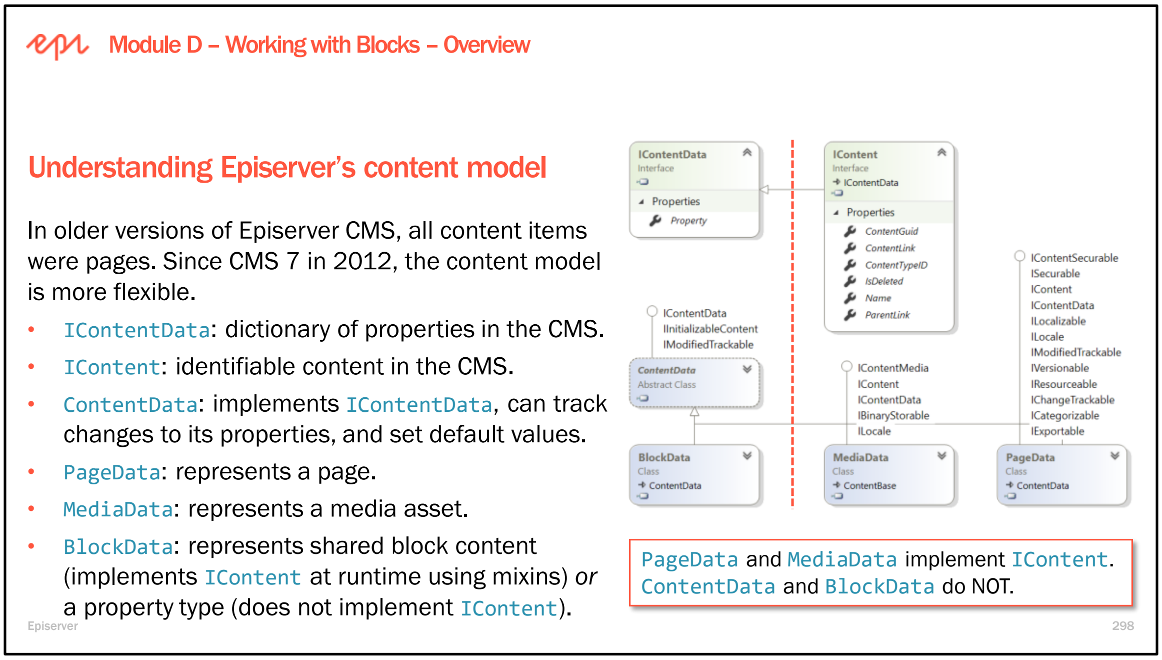


Module D – Working with Blocks – Overview

When to use a partial page or a block

- **Does the content need a public reference?** Might a visitor want to bookmark the content?
  Pages and media have URLs for public references; blocks do not.
- **Is the content website functionality?** For example: Share this on Facebook, LinkedIn, Twitter, and so on, a dynamically-generated list of content, or a quiz component?
  Website functionality should be implemented as blocks.
- **Do you want optimal performance?**
  Blocks can be controller-less and that will give about a 10% performance improvement.
- **Do you want to tease page content to draw a visitor deeper into your website?**
  All page types that you want to be able to promote by adding them to a content area should have a partial template. If a page is promoted by using a block, this will create an overhead for the editor since they need to manage the block as a separate item in parallel with the page. One of the points of a CMS is to be able to create an item of content once, and renderer it in different scenarios. ☺

Episerver 297

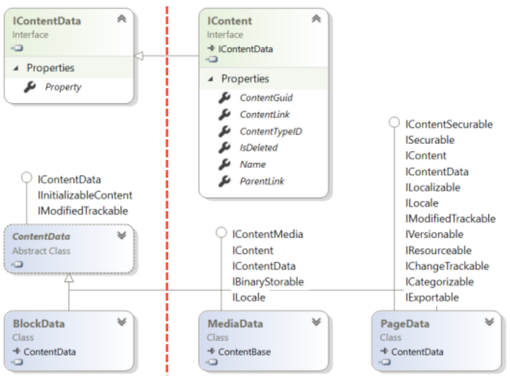


Module D – Working with Blocks – Overview

Understanding Episerver's content model

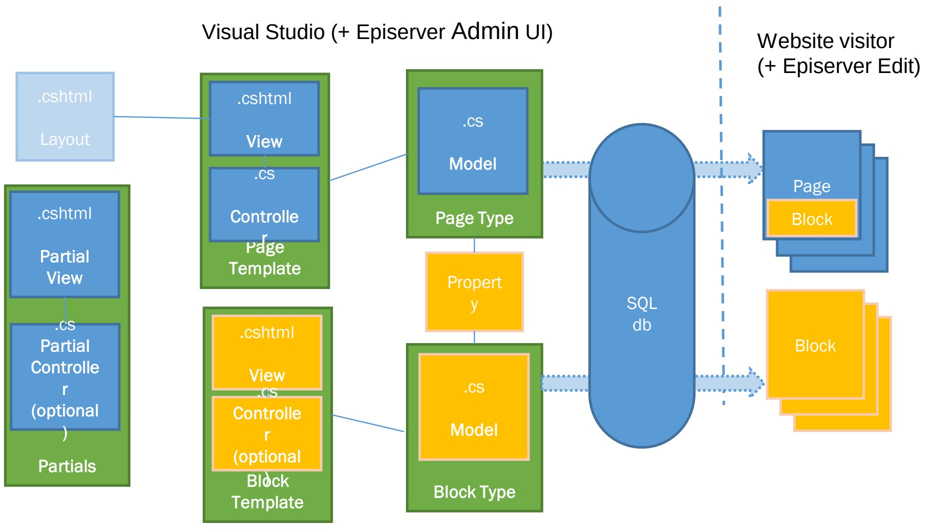
In older versions of Episerver CMS, all content items were pages. Since CMS 7 in 2012, the content model is more flexible.

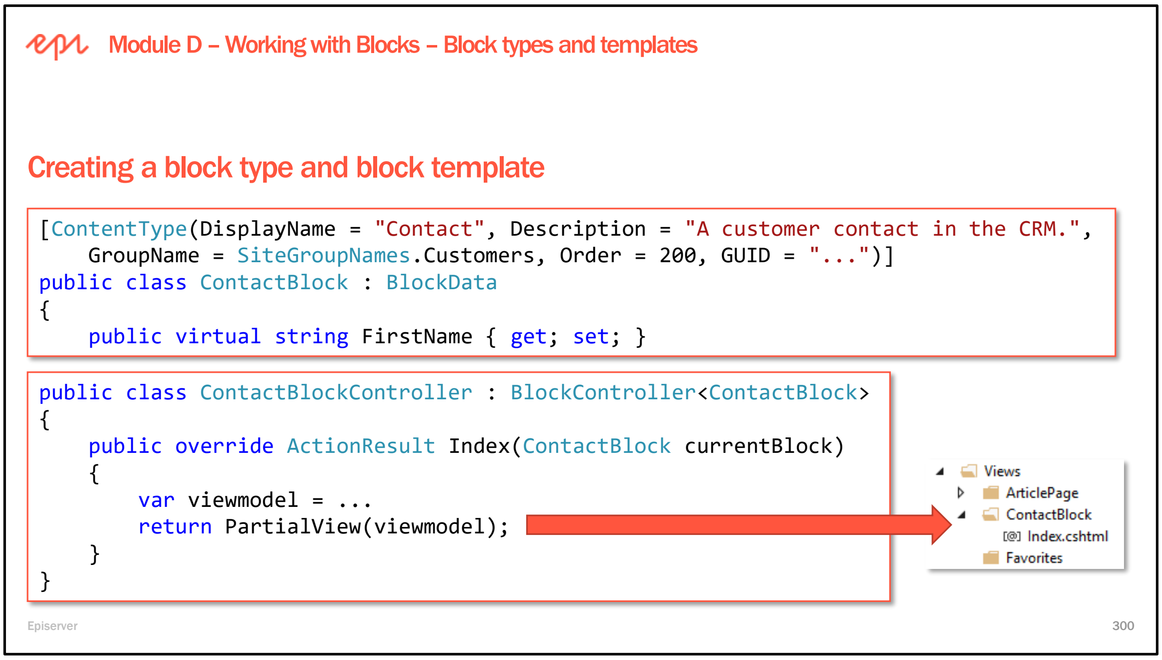
- `IContentData`: dictionary of properties in the CMS.
- `IContent`: identifiable content in the CMS.
- `ContentData`: implements `IContentData`, can track changes to its properties, and set default values.
- `PageData`: represents a page.
- `MediaData`: represents a media asset.
- `BlockData`: represents shared block content (implements `IContent` at runtime using mixins) *or* a property type (does not implement `IContent`).



`PageData` and `MediaData` implement `IContent`.
`ContentData` and `BlockData` do NOT.

Episerver 298





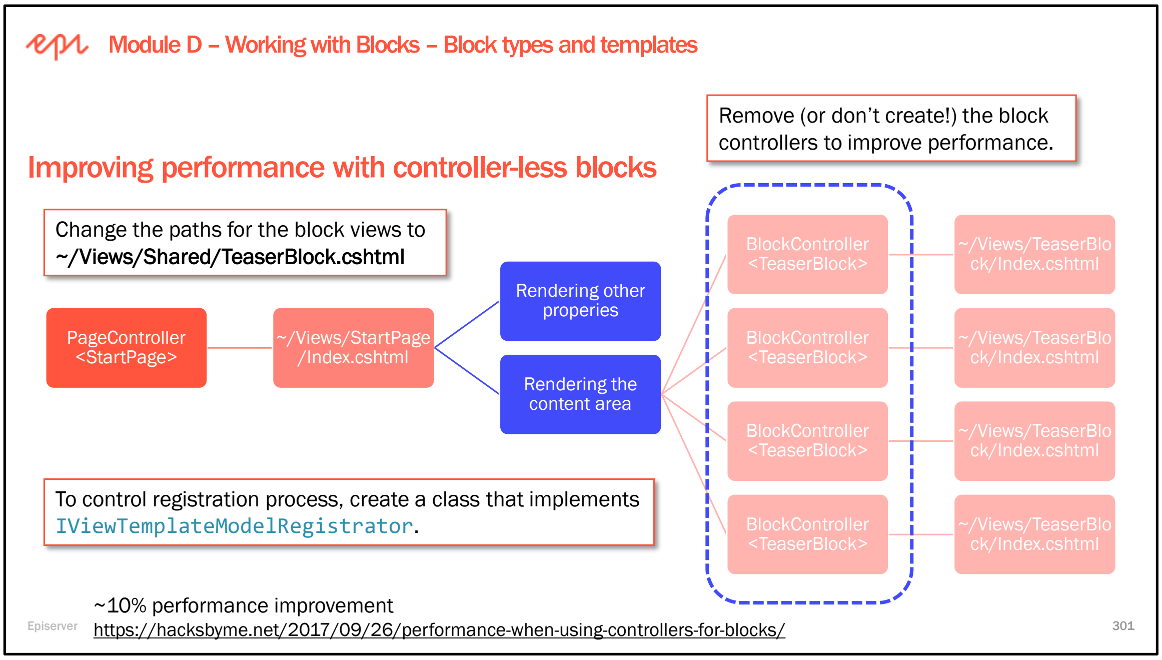
Creating a block type and block template

```
[ContentType(DisplayName = "Contact", Description = "A customer contact in the CRM.",
    GroupName = SiteGroupNames.Customers, Order = 200, GUID = "...")]
public class ContactBlock : BlockData
{
    public virtual string FirstName { get; set; }
```

```
public class ContactBlockController : BlockController<ContactBlock>
{
    public override ActionResult Index(ContactBlock currentBlock)
    {
        var viewmodel = ...
        return PartialView(viewmodel);
    }
}
```

```
Views
  ArticlePage
  ContactBlock
    Index.cshtml
  Favorites
```

currentBlock is the instance of your block type and contains the properties defined in it, regardless if the block is shared or used as a property.



Module D – Working with Blocks – Block types and templates

Improving performance with controller-less blocks

Remove (or don't create!) the block controllers to improve performance.

Change the paths for the block views to ~/Views/Shared/TeaserBlock.cshtml

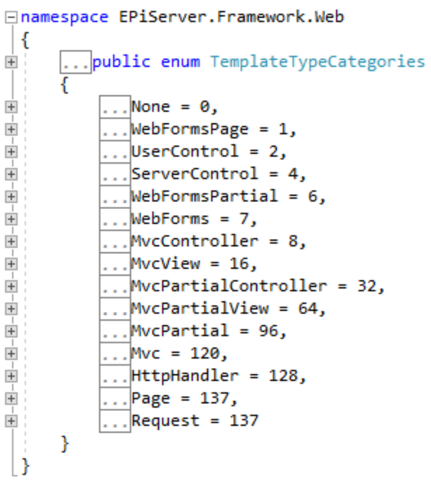
To control registration process, create a class that implements `IViewTemplateModelRegistrator`.

~10% performance improvement
https://hacksbyme.net/2017/09/26/performance-when-using-controllers-for-blocks/

PageController <StartPage>
~/Views/StartPage/Index.cshtml
Rendering other properies
Rendering the content area
BlockController <TeaserBlock>
~/Views/TeaserBlock/Index.cshtml
Episerver
301

The reason that you may choose to register a content template using the `IViewTemplateModelRegistrator` is that for templates with no controller, the developer cannot decorate with [`TemplateDescriptor`] attribute.

```
public class TemplateRegistrator : IViewTemplateModelRegistrator
{
    public void Register(TemplateModelCollection viewTemplateModelRegistrator)
    {
        // register two templates for StartPage

        var defaultTemplate = new TemplateModel
        {
            TemplateType = typeof(MuppetController),
            ModelType = typeof(StartPage),
            Default = true,
            DisplayName = "Start Page Template (Default)",
            Description = "Default template for StartPage",
        };

        var alternativeTemplate = new TemplateModel
        {
            TemplateType = typeof(StartPageController),
            ModelType = typeof(StartPage),
            DisplayName = "Start Page Template (Alternative)",
            Description = "Alternative template for StartPage",
        };

        viewTemplateModelRegistrator.Add(typeof(StartPage),
            defaultTemplate, alternativeTemplate);
    }
}
```

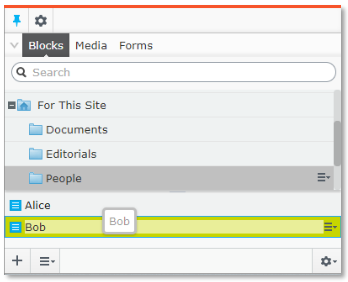


```
namespace EPiServer.Framework.Web
{
    public enum TemplateTypeCategories
    {
        None = 0,
        WebFormsPage = 1,
        UserControl = 2,
        ServerControl = 4,
        WebFormsPartial = 6,
        WebForms = 7,
        MvcController = 8,
        MvcView = 16,
        MvcPartialController = 32,
        MvcPartialView = 64,
        MvcPartial = 96,
        Mvc = 120,
        HttpHandler = 128,
        Page = 137,
        Request = 137
    }
}
```

Module D – Working with Blocks – Shared and property blocks

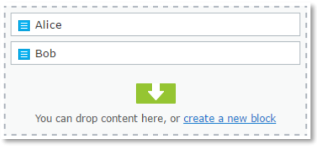
Using a block as a shared asset

Shared blocks aka global blocks are stored in folders in the **Assets** pane **Blocks** tab.



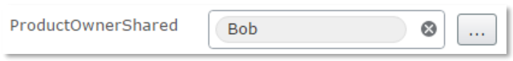
They can be added to a **ContentArea** property:

```
// can have references to any content with a partial template
public virtual ContentArea PagesBlocksAndMedia { get; set; }
```



...or used to set a **ContentReference** property:

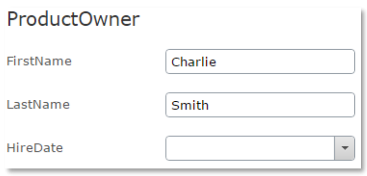
```
// can have a reference to one shared block
[AllowedTypes(typeof(EmployeeBlock))]
public virtual ContentReference
    ProductOwnerShared { get; set; }
```



Using a block as a property type

Blocks used as a property type are stored, versioned and loaded with the content that has a property of that block type, and cannot be referenced on their own, unlike a shared block.

```
// cannot reference a shared block
public virtual EmployeeBlock ProductOwner { get; set; }
```



ProductOwner

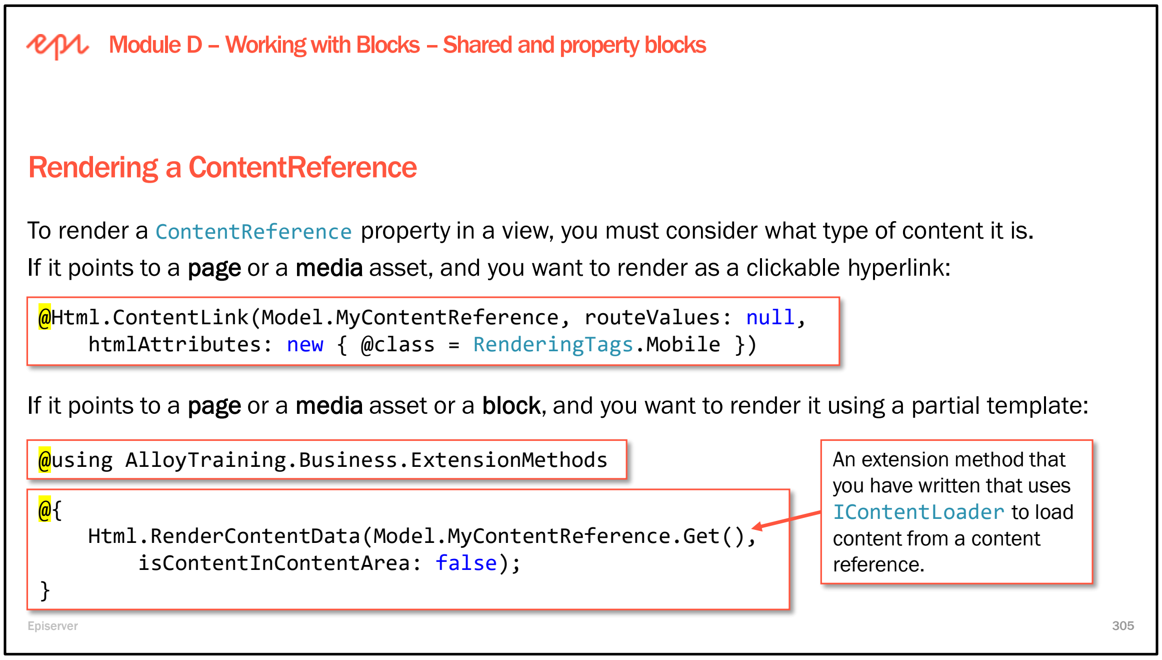
FirstName	Charlie
LastName	Smith
HireDate	

If you have a block used as a property type, you can render it in a view using **PropertyFor**:

```
@Html.PropertyFor(m => m.ProductOwner)
```

To prevent a block type from being used as a shared block, but still allow it to be a property type, use the AvailableContentTypes attribute to prevent it being created inside a folder:

```
[ContentType(DisplayName = "Person")]
[AvailableContentTypes(Availability = Availability.Specific,
    ExcludeOn = new[] { typeof(ContentFolder) })]
public class PersonBlock : BlockData
```



Module D – Working with Blocks – Shared and property blocks

Rendering a ContentReference

To render a `ContentReference` property in a view, you must consider what type of content it is.
If it points to a **page** or a **media** asset, and you want to render as a clickable hyperlink:

```
@Html.ContentLink(Model.MyContentReference, routeValues: null,
    htmlAttributes: new { @class = RenderingTags.Mobile })
```

If it points to a **page** or a **media** asset or a **block**, and you want to render it using a partial template:

```
@using AlloyTraining.Business.ExtensionMethods
```

```
@{
    Html.RenderContentData(Model.MyContentReference.Get(),
        isContentInContentArea: false);
}
```

An extension method that you have written that uses `IContentLoader` to load content from a content reference.

Episerver 305

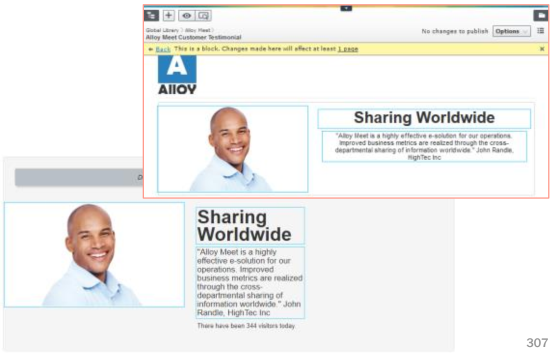
Module D – Working with Blocks – Previewing shared blocks

Defining a preview content template

```
[TemplateDescriptor(Inherited = true, Tags = new[] { RenderingTags.Preview })]
public class PreviewController : ActionControllerBase, IRenderTemplate<BlockData>
```

```
<div class="row">
  <div class="span12">
   @Html.PropertyFor(x => x.ContentArea,
     new { Tags = new[] { "full" })
  </div>
</div>
```

```
<div class="row">
  <div class="span8">
   @Html.PropertyFor(x => x.ContentArea,
     new { Tags = new[] { "wide" })
  </div>
</div>
```


Alloy Meet Customer Testimonial
No changes to publish
Options
Back This is a block. Changes made here will affect at least 1 page
Alloy
Sharing Worldwide
Sharing Worldwide

"Alloy Meet is a highly effective e-solution for our operations. Improved business metrics are realized through the cross-departmental sharing of information worldwide." John Randle, HighTec Inc

There have been 344 visitors today.

307

Preview rendering for blocks

A preview adds on-page editing functionality, and a realistic view of what the block will look like when added to content areas with different widths.

The basic idea is that when editing shared blocks you should have a page template registered as IRenderTemplate<BlockData> with a TemplateDescriptor that has Tag = "Preview". That template will then be used when editing the block in on-page Edit view.

https://world.episerver.com/documentation/developer-guides/CMS/rendering/preview-rendering-for-blocks/

More information about preview of blocks during on-page edit is available on Episerver World, for example in this blog article by Johan Björnfot:

http://world.episerver.com/Blogs/Johan-Bjornfot/Dates1/2012/9/Episerver-7--Rendering-of-content/

Module D – Working with Blocks

Exercises D1 to D5 – Working with blocks

Estimated time: 60 minutes

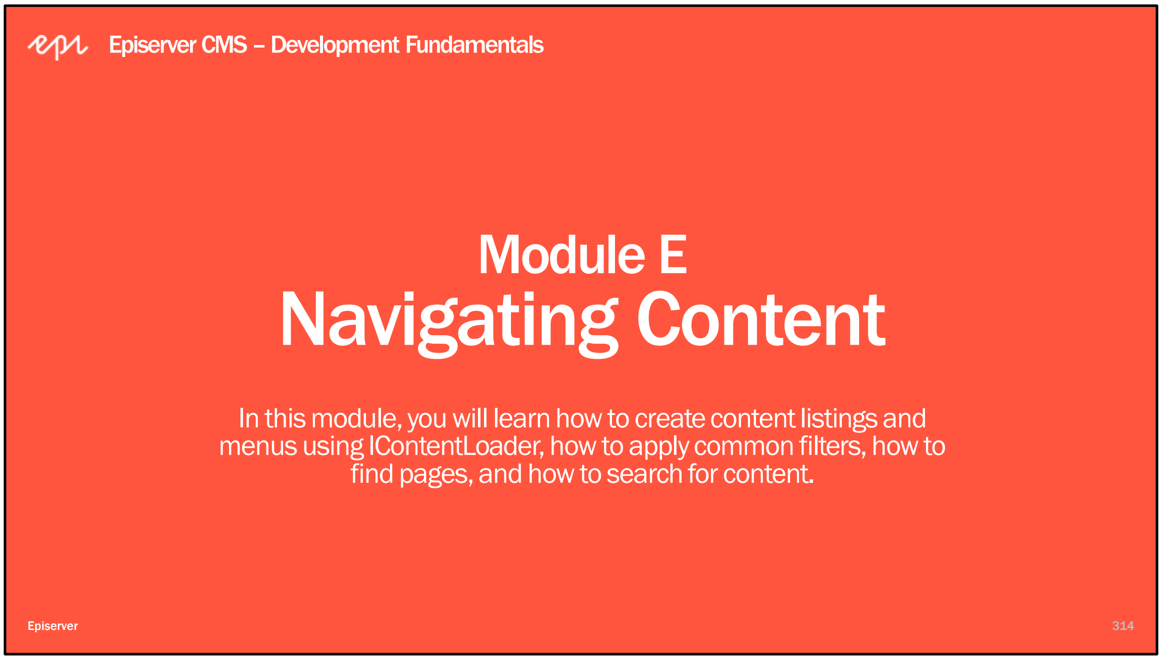
Prerequisites: Exercises B1 – B4, C1 – C4

1. Creating a controller-less block
2. Creating a block with a controller
3. Creating a preview renderer for partial pages and shared blocks
4. Moving properties to the basic info area

Prerequisites: Exercises B1 – B4.

5. Using a block as a content property type

Episerver 308



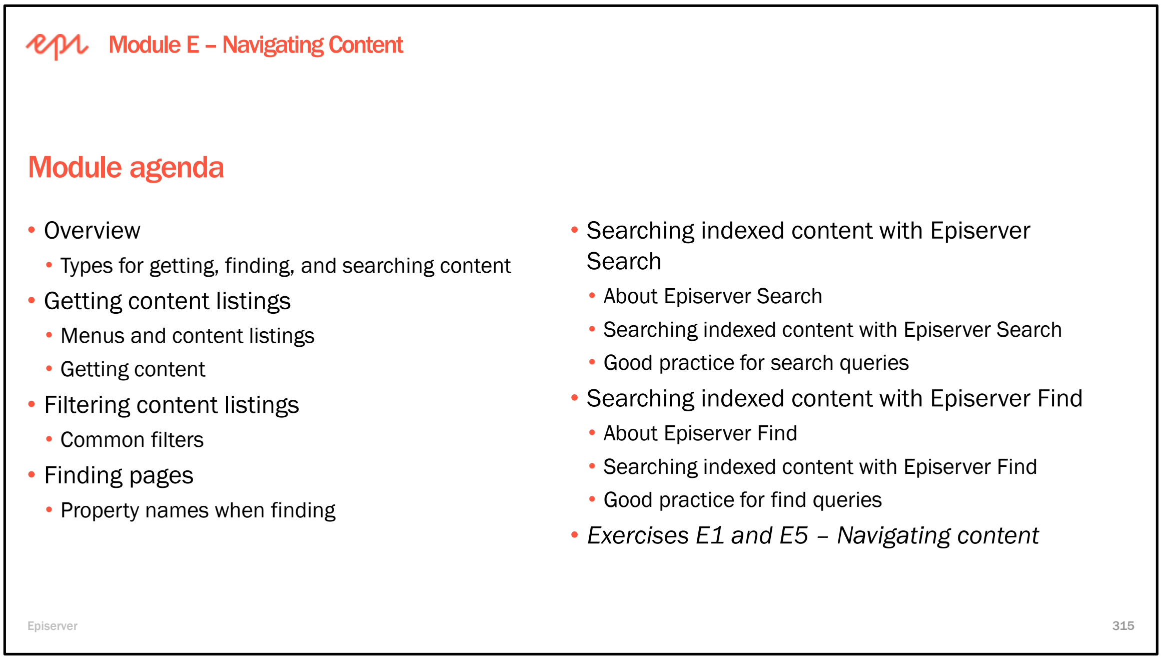
Episerver CMS – Development Fundamentals

Module E
Navigating Content

In this module, you will learn how to create content listings and menus using IContentLoader, how to apply common filters, how to find pages, and how to search for content.

Episerver

314

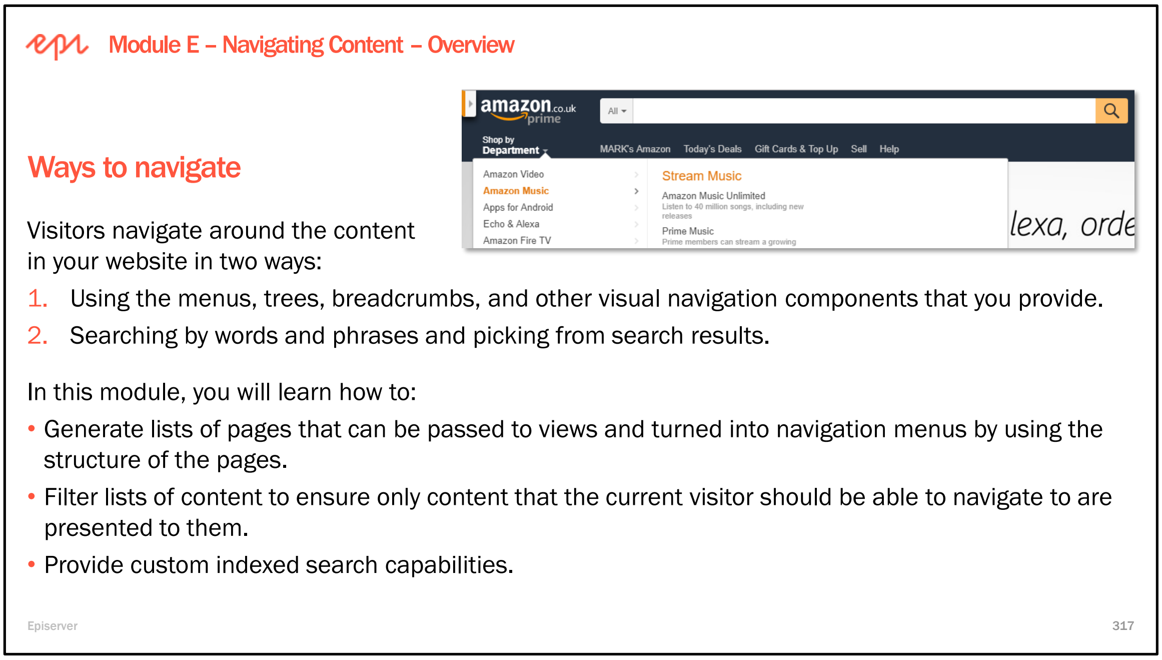


Module E – Navigating Content

Module agenda

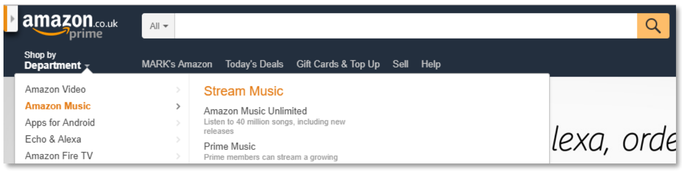
- Overview
  - Types for getting, finding, and searching content
- Getting content listings
  - Menus and content listings
  - Getting content
- Filtering content listings
  - Common filters
- Finding pages
  - Property names when finding
- Searching indexed content with Episerver Search
  - About Episerver Search
  - Searching indexed content with Episerver Search
  - Good practice for search queries
- Searching indexed content with Episerver Find
  - About Episerver Find
  - Searching indexed content with Episerver Find
  - Good practice for find queries
- *Exercises E1 and E5 – Navigating content*

Episerver 315



Module E – Navigating Content – Overview

Ways to navigate



Visitors navigate around the content in your website in two ways:

1. Using the menus, trees, breadcrumbs, and other visual navigation components that you provide.
2. Searching by words and phrases and picking from search results.

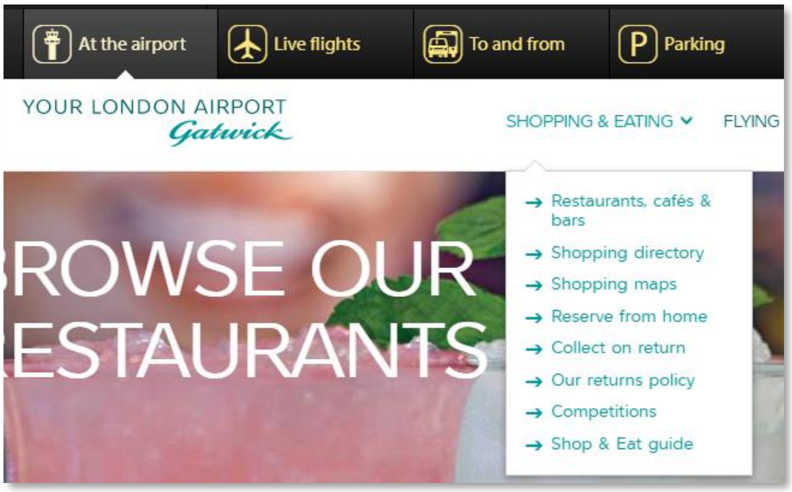
In this module, you will learn how to:

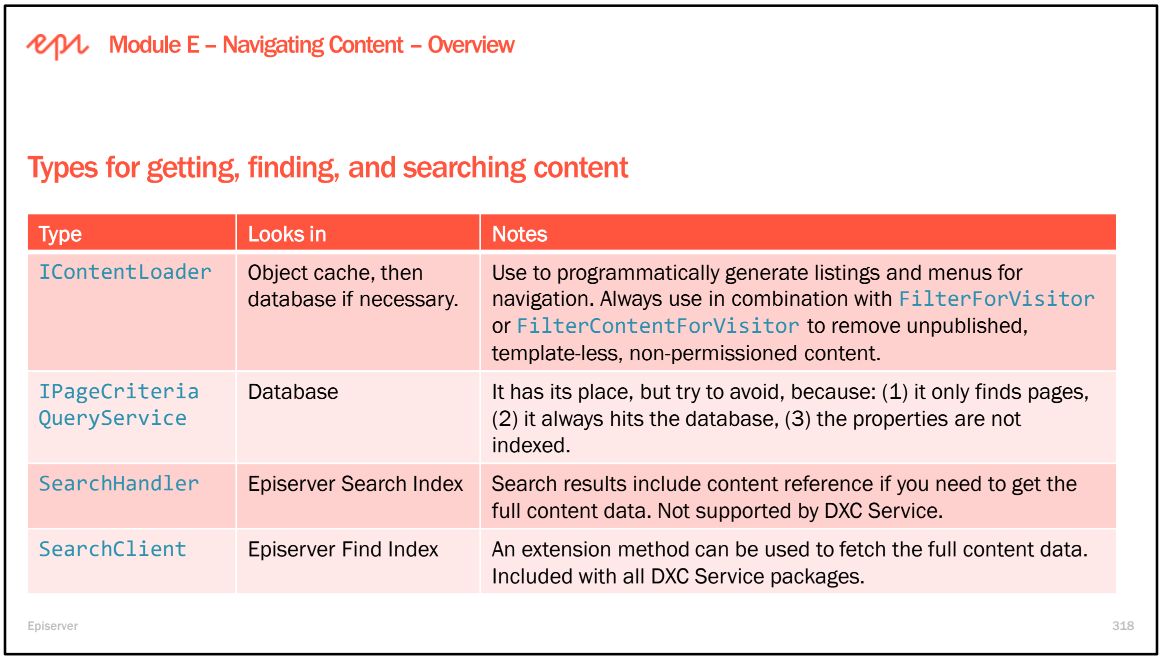
- Generate lists of pages that can be passed to views and turned into navigation menus by using the structure of the pages.
- Filter lists of content to ensure only content that the current visitor should be able to navigate to are presented to them.
- Provide custom indexed search capabilities.

Episerver 317

Menus and content listings are collections of pages that are rendered in a particular way, e.g.
- **Top-level menu**: children of the Start page with icons
- **Submenu**: children of the default menu item
- **Dropdown menu**: children of each of the submenu items

Simply output URLs, name or title, publish dates, and so on, that you want to show to visitors.



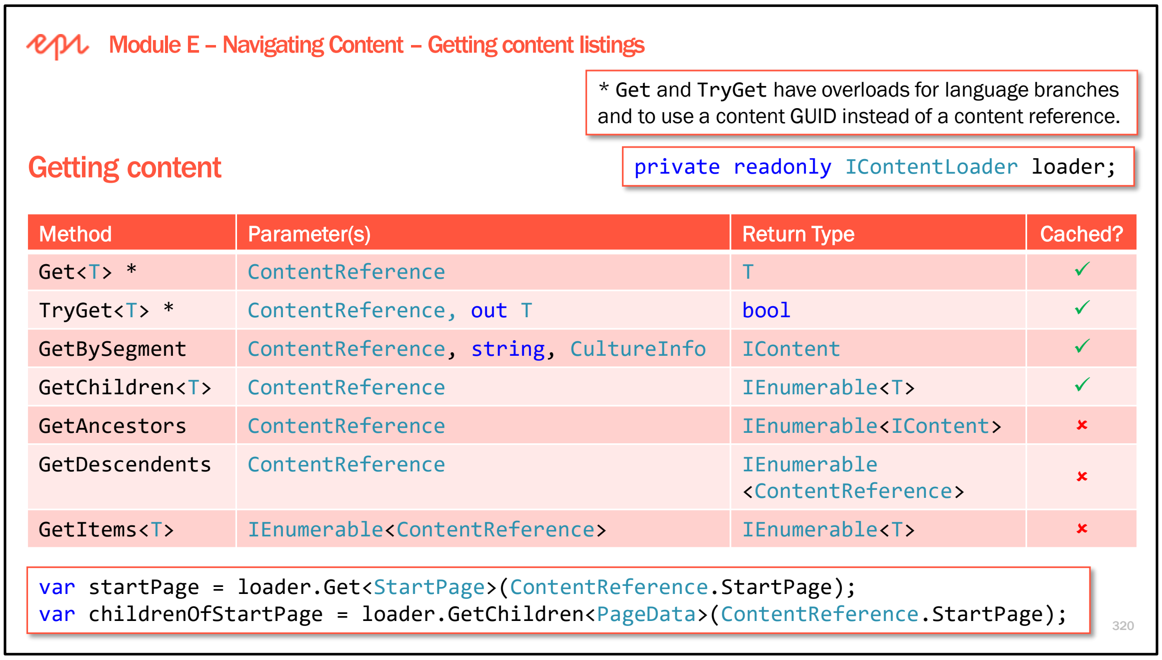


Module E – Navigating Content – Overview

Types for getting, finding, and searching content

Type	Looks in	Notes
`IContentLoader`	Object cache, then database if necessary.	Use to programmatically generate listings and menus for navigation. Always use in combination with `FilterForVisitor` or `FilterContentForVisitor` to remove unpublished, template-less, non-permissioned content.
`IPageCriteriaQueryService`	Database	It has its place, but try to avoid, because: (1) it only finds pages, (2) it always hits the database, (3) the properties are not indexed.
`SearchHandler`	Episerver Search Index	Search results include content reference if you need to get the full content data. Not supported by DXC Service.
`SearchClient`	Episerver Find Index	An extension method can be used to fetch the full content data. Included with all DXC Service packages.

Episerver 318



Module E – Navigating Content – Getting content listings

* Get and TryGet have overloads for language branches and to use a content GUID instead of a content reference.

Getting content

```
private readonly IContentLoader loader;
```

Method	Parameter(s)	Return Type	Cached?
`Get<T>` *	`ContentReference`	`T`	✓
`TryGet<T>` *	`ContentReference, out T`	`bool`	✓
`GetBySegment`	`ContentReference, string, CultureInfo`	`IContent`	✓
`GetChildren<T>`	`ContentReference`	`IEnumerable<T>`	✓
`GetAncestors`	`ContentReference`	`IEnumerable<IContent>`	×
`GetDescendents`	`ContentReference`	`IEnumerable <ContentReference>`	×
`GetItems<T>`	`IEnumerable<ContentReference>`	`IEnumerable<T>`	×

```
var startPage = loader.Get<StartPage>(ContentReference.StartPage);
var childrenOfStartPage = loader.GetChildren<PageData>(ContentReference.StartPage);
```

320

Getting a single content item

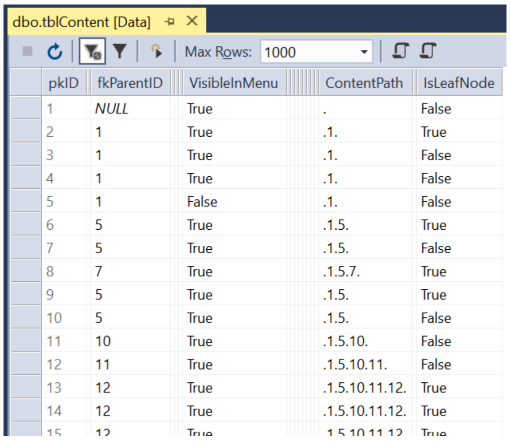
Code defensively when getting content, for example:

```
// throws exception if it is NOT a NewsPage
var newsPage = loader.Get<NewsPage>(contentReference);
```

```
// returns null if it is NOT a NewsPage
var newsPage = loader.Get<IContent>(contentReference) as NewsPage;
```

Getting ancestors

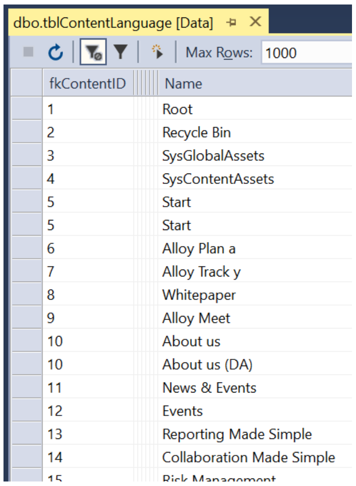
Although `GetAncestors()` hits the database, it only needs to return a single row to return the IDs of its ancestors, for example, pkID 13 is **Reporting Made Simple**, whose parent is **Events**, whose parent is **News & Events**, whose parent is **About us**, whose parent is **Start**, whose parent is **Root**:



dbo.tblContent [Data]

Max Rows: 1000

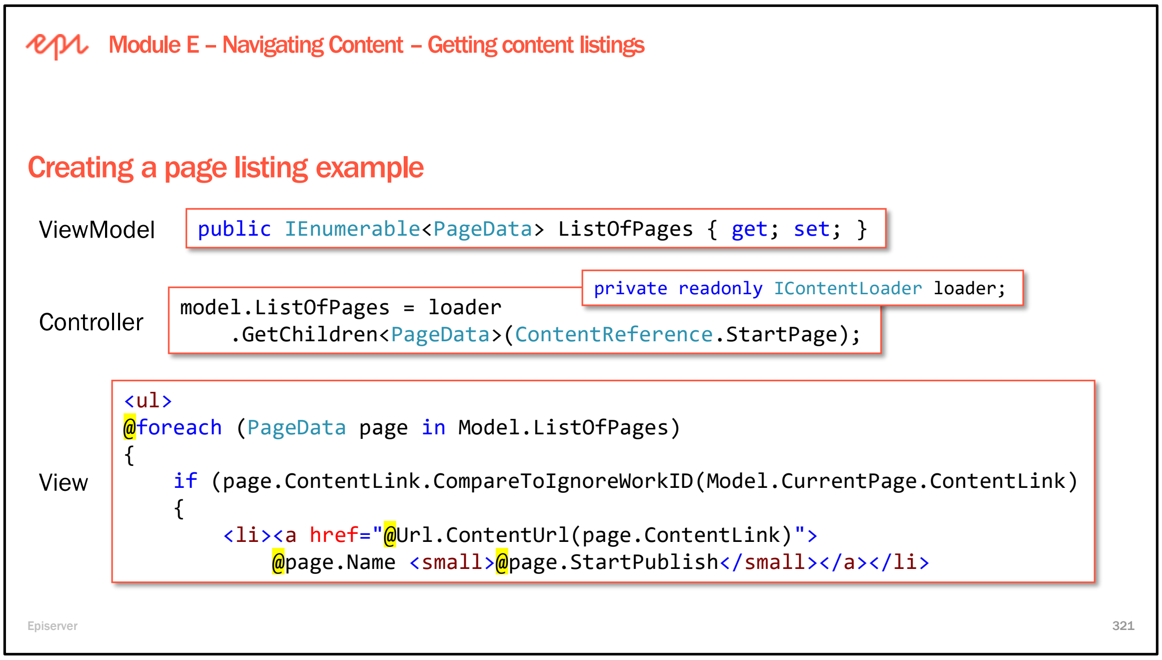
pkID	fkParentID	VisibleInMenu	ContentPath	IsLeafNode
1	*NULL*	True	.	False
2	1	True	.1.	True
3	1	True	.1.	False
4	1	True	.1.	False
5	1	False	.1.	False
6	5	True	.1.5.	True
7	5	True	.1.5.	False
8	7	True	.1.5.7.	True
9	5	True	.1.5.	True
10	5	True	.1.5.	False
11	10	True	.1.5.10.	False
12	11	True	.1.5.10.11.	False
13	12	True	.1.5.10.11.12.	True
14	12	True	.1.5.10.11.12.	True
15	12	True	.1.5.10.11.12.	True



dbo.tblContentLanguage [Data]

Max Rows: 1000

fkContentID	Name
1	Root
2	Recycle Bin
3	SysGlobalAssets
4	SysContentAssets
5	Start
5	Start
6	Alloy Plan a
7	Alloy Track y
8	Whitepaper
9	Alloy Meet
10	About us
10	About us (DA)
11	News & Events
12	Events
13	Reporting Made Simple
14	Collaboration Made Simple
15	Risk Management



Module E – Navigating Content – Getting content listings

Creating a page listing example

ViewModel

```
public IEnumerable<PageData> ListOfPages { get; set; }
```

Controller

```
private readonly IContentLoader loader;
```

```
model.ListOfPages = loader
    .GetChildren<PageData>(ContentReference.StartPage);
```

View

```
<ul>
@foreach (PageData page in Model.ListOfPages)
{
    if (page.ContentLink.CompareToIgnoreWorkID(Model.CurrentPage.ContentLink)
    {
        <li><a href="@Url.ContentUrl(page.ContentLink)">
            @page.Name <small>@page.StartPublish</small></a></li>
```

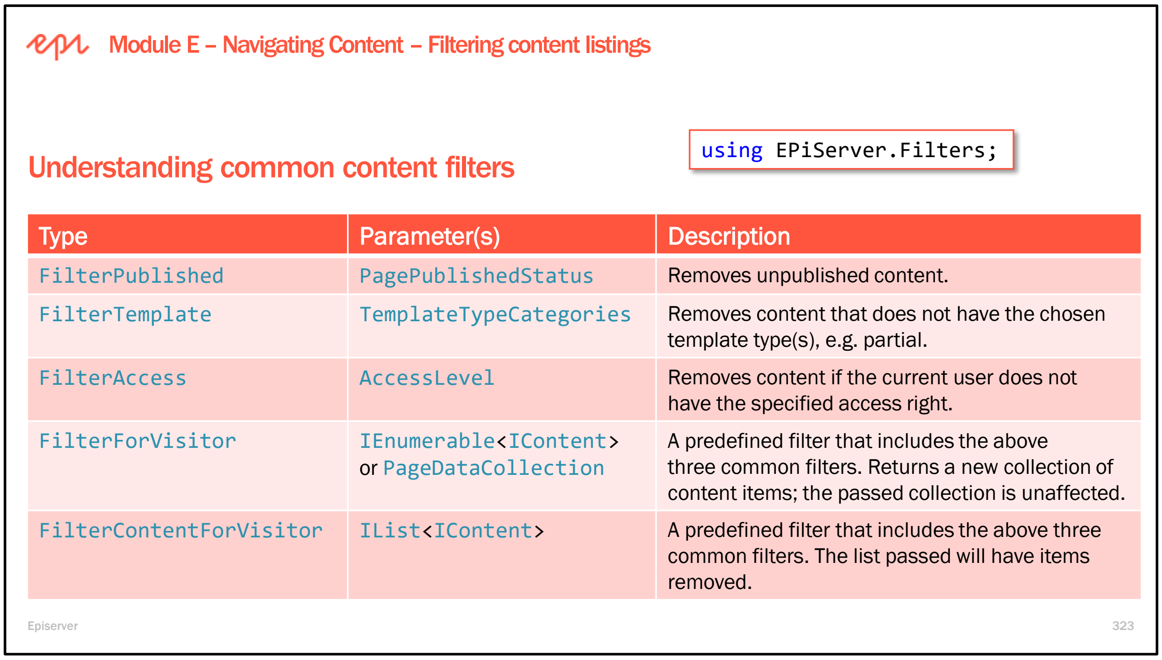
Episerver 321

Getting descendants and then getting items

Get an instance of an IContentLoader and get the descendants of the Start page. This would be ALL children, grand-children, great-grand-children, and so on, so the method doesn't return the PageData instances, instead it returns ContentReference instances.

To fetch the actual PageData instances, IContentLoader has a **GetItems** method. You must pass an instance of **LoaderOptions** as the second parameter even if you don't need to set options like language branches.

```
var loader1 = ServiceLocator.Current.GetInstance<IContentLoader>();
IEnumerable<ContentReference> descOfStartAsRefs =
    loader1.GetDescendents(ContentReference.StartPage);
IEnumerable<IContent> descOfStartAsContent =
    loader1.GetItems(descOfStartAsRefs, new LoaderOptions());
```



Module E – Navigating Content – Filtering content listings

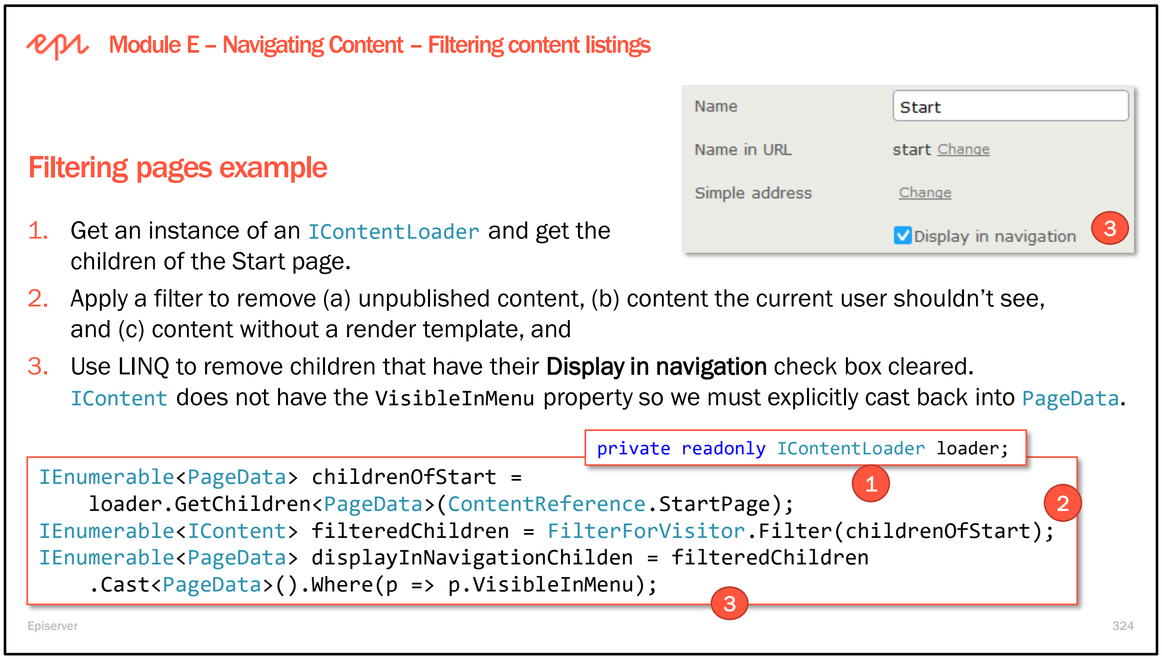
Understanding common content filters

```
using EPiServer.Filters;
```

Type	Parameter(s)	Description
FilterPublished	PagePublishedStatus	Removes unpublished content.
FilterTemplate	TemplateTypeCategories	Removes content that does not have the chosen template type(s), e.g. partial.
FilterAccess	AccessLevel	Removes content if the current user does not have the specified access right.
FilterForVisitor	IEnumerable<IContent> or PageDataCollection	A predefined filter that includes the above three common filters. Returns a new collection of content items; the passed collection is unaffected.
FilterContentForVisitor	IList<IContent>	A predefined filter that includes the above three common filters. The list passed will have items removed.

Episerver 323

Filtering for access rights
The pages retrieved using calls to, for example, GetChildren will not automatically be filtered based on the site visitor's read access, published status, or template availability.
To achieve this, you can use EPiServer.Filters class FilterForVisitor, which calls:
- FilterPublished
- FilterAccess
- FilterTemplate



Module E – Navigating Content – Filtering content listings

Filtering pages example

Name	Start
Name in URL	start Change
Simple address	Change
	☑ Display in navigation 3

1. Get an instance of an `IContentLoader` and get the children of the Start page.
2. Apply a filter to remove (a) unpublished content, (b) content the current user shouldn't see, and (c) content without a render template, and
3. Use LINQ to remove children that have their **Display in navigation** check box cleared. `IContent` does not have the `VisibleInMenu` property so we must explicitly cast back into `PageData`.

```
private readonly IContentLoader loader;
```

```
IEnumerable<PageData> childrenOfStart =
    loader.GetChildren<PageData>(ContentReference.StartPage);
IEnumerable<IContent> filteredChildren = FilterForVisitor.Filter(childrenOfStart);
IEnumerable<PageData> displayInNavigationChilden = filteredChildren
    .Cast<PageData>().Where(p => p.VisibleInMenu);
```

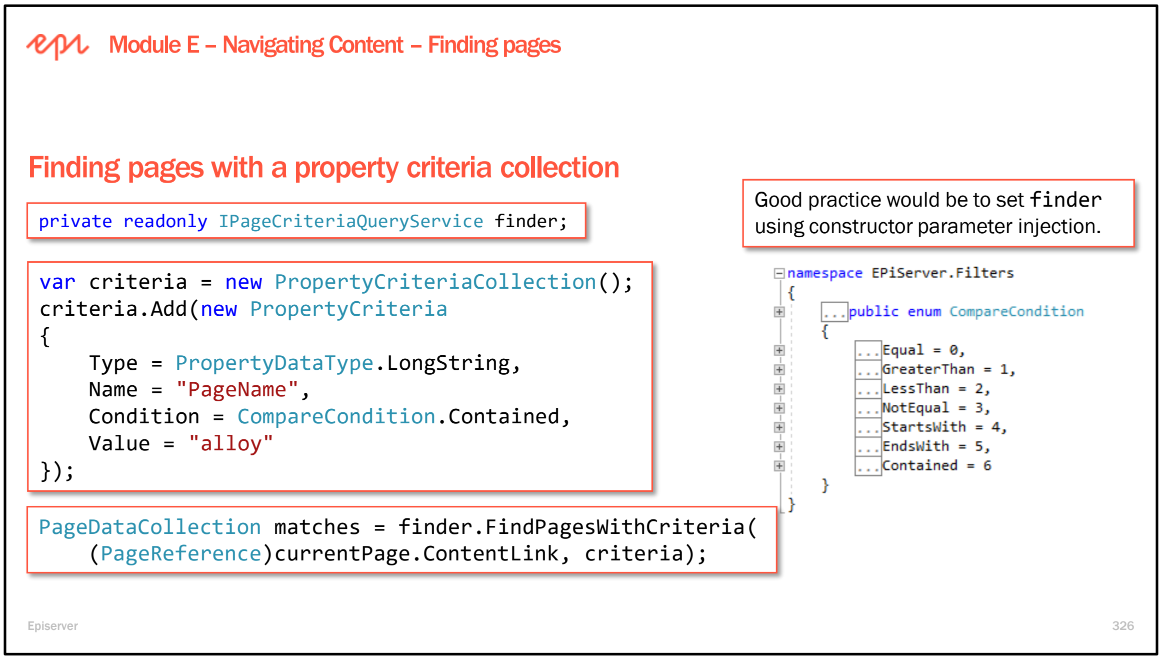
1 2 3

Episerver 324

Using lambda expressions in LINQ

In most cases, avoid Episerver filters for sorting and filtering, and use LINQ instead:

```
IContentRepository contentRepository =
    ServiceLocator.Current.GetInstance<IContentRepository>();

IEnumerable<ProductPage> pages =
    contentRepository.GetChildren<ProductPage>(ContentReference.StartPage);

IEnumerable<IContent> filteredContent = FilterForVisitor.Filter(pages);

// need to cast back into ProductPages before using StartPublish property
IEnumerable<ProductPage> filteredPages = filterContent.Take(3).Cast<ProductPage>();
IEnumerable<ProductPage> sortedPages = filteredPages.OrderBy(page => page.StartPublish);
```



Module E – Navigating Content – Finding pages

Finding pages with a property criteria collection

```
private readonly IPageCriteriaQueryService finder;
```

Good practice would be to set `finder` using constructor parameter injection.

```
var criteria = new PropertyCriteriaCollection();
criteria.Add(new PropertyCriteria
{
    Type = PropertyDataType.LongString,
    Name = "PageName",
    Condition = CompareCondition.Contained,
    Value = "alloy"
});
```

```
namespace EPiServer.Filters
{
    public enum CompareCondition
    {
        Equal = 0,
        GreaterThan = 1,
        LessThan = 2,
        NotEqual = 3,
        StartsWith = 4,
        EndsWith = 5,
        Contained = 6
    }
}
```

```
PageDataCollection matches = finder.FindPagesWithCriteria(
    (PageReference)currentPage.ContentLink, criteria);
```

Episerver 326

Method located in the IPageCriteriaQueryService interface
FindPagesWithCriteria()
Returns a PageDataCollection with the results
Varying parameters:
A starting point for the search
A set of criteria
Optionally, a required access
Optionally, a language branch
Optionally, a language selector

Example that lists all pages for a certain page type:

```
public EPiServer.Core.PageDataCollection
FindPagesOfPageType(EPiServer.Core.PageReference pageLink)
{
    PropertyCriteriaCollection criteria = new PropertyCriteriaCollection();
    PropertyCriteria criterion = new PropertyCriteria();
    criterion.Condition = EPiServer.Filters.CompareCondition.Equal;
    criterion.Name = "PageTypeID";
    criterion.Type = EPiServer.Core.PropertyDataType.PageType;
    criterion.Value = Locate.ContentTypeRepository().Load("StandardPage").ID.ToString();
    criterion.Required = true;
    criteria.Add(criterion);
    return
ServiceLocator.Current.GetInstance<IPageCriteriaQueryService>().FindPagesWithCriteria(pageLink,
criteria);
}
```

Note: For queries that rarely change, you should cache the result that comes back from FindPagesWithCriteria.

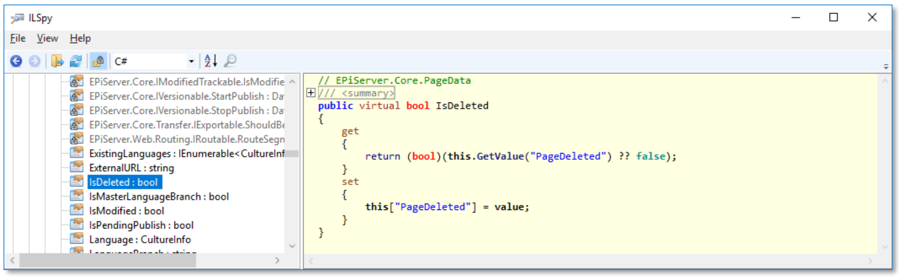
Module E – Navigating Content – Finding pages

Discovering property names to use when finding

When using the `FindPagesWithCriteria` method, you need to supply the name of a property using its name stored in the CMS database, not the name of a property in the class.

`PageData` has a property named `IsDeleted`, but you must use `PageDeleted` as the name instead.

You can check what name to use by using tools like **ILSpy** to look inside the class implementation.



```
EPiServer.Core.IModifiedTrackable.IsModifie
EPiServer.Core.IVersionable.StartPublish : Da
EPiServer.Core.IVersionable.StopPublish : Da
EPiServer.Core.Transfer.IExportable.ShouldBe
EPiServer.Web.Routing.IRoutable.RouteSegn
ExistingLanguages : IEnumerable<CultureInf
ExternalURL : string
IsDeleted : bool
IsMasterLanguageBranch : bool
IsModified : bool
IsPendingPublish : bool
Language : CultureInfo
```

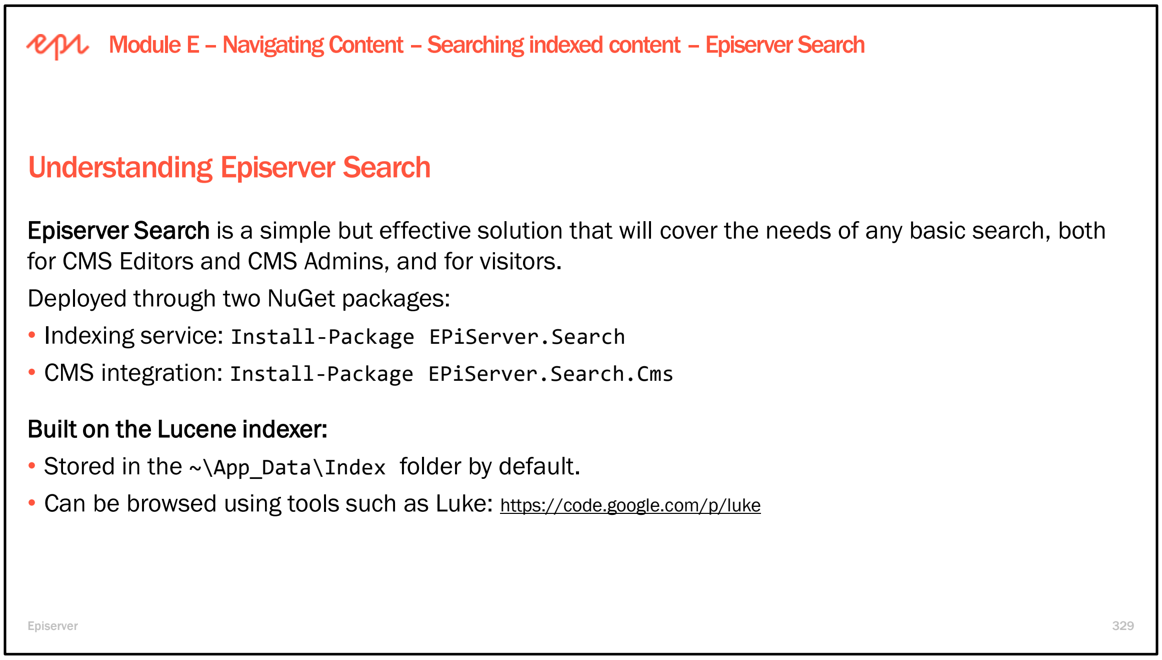
```
// EPiServer.Core.PageData
/// <summary>
public virtual bool IsDeleted
{
    get
    {
        return (bool)(this.GetValue("PageDeleted") ?? false);
    }
    set
    {
        this["PageDeleted"] = value;
    }
}
```

Episerver 327

Criteria are rules used to constrain search results
Specify the property to be examined
Specify the condition by using the CompareCondition enum in EPiServer.Filters namespace

- Equal/NotEqual
- Less/GreaterThan
- Starts/EndsWith
- Contained

Specify the value used for comparison
Specify the type of the value to be examined
Create instances of the PropertyCriteria class
Multiple criteria make up a PropertyCriteriaCollection



Module E – Navigating Content – Searching indexed content – Episerver Search

Understanding Episerver Search

Episerver Search is a simple but effective solution that will cover the needs of any basic search, both for CMS Editors and CMS Admins, and for visitors.

Deployed through two NuGet packages:

- Indexing service: `Install-Package EPiServer.Search`
- CMS integration: `Install-Package EPiServer.Search.Cms`

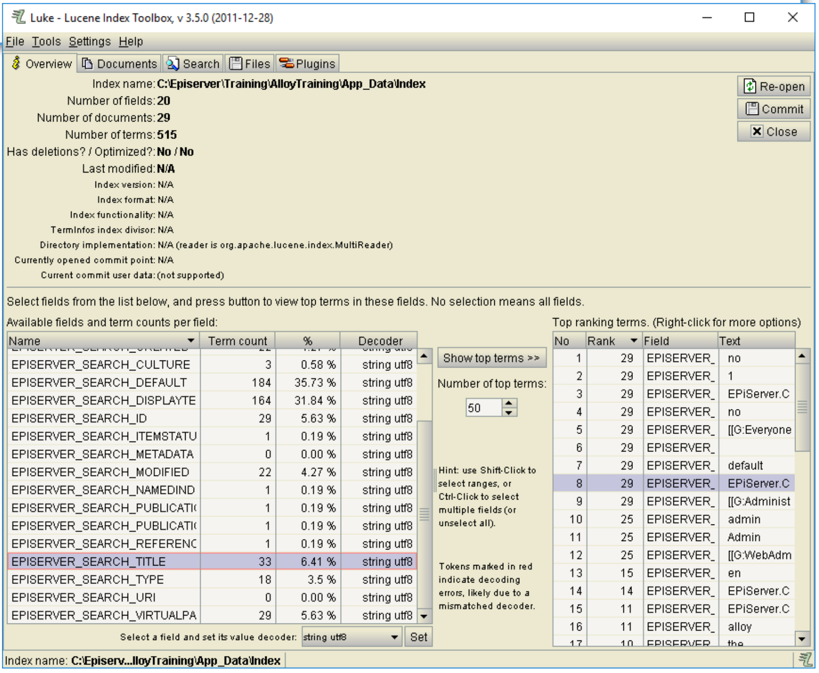
Built on the Lucene indexer:

- Stored in the `~\App_Data\Index` folder by default.
- Can be browsed using tools such as Luke: https://code.google.com/p/luke

Episerver 329

```
<episerver.search active="true">
  <namedIndexingServices defaultService="serviceName">
    <services>
      <add name="serviceName" accessKey="local"
           baseUri="http://localhost:53991/IndexingService/IndexingService.svc" />
    </services>
  </namedIndexingServices>
  <searchResultFilter defaultInclude="true">
    <providers />
  </searchResultFilter>
</episerver.search>
```

The Java-based **Luke** tool can be useful for understanding and fixing indexing problems in Lucene indexes.


Luke - Lucene Index Toolbox, v 3.5.0 (2011-12-28)
File Tools Settings Help
Overview Documents Search Files Plugins
Re-open
Commit
Close
Show top terms >>
Set

```
Index name: C:\Episerver\Training\AlloyTraining\App_Data\Index
Number of fields: 20
Number of documents: 29
Number of terms: 515
Has deletions? / Optimized?: No / No
Last modified: N/A
Index version: N/A
Index format: N/A
Index functionality: N/A
TermInfos index divisor: N/A
Directory implementation: N/A (reader is org.apache.lucene.index.MultiReader)
Currently opened commit point: N/A
Current commit user data: (not supported)
```

Select fields from the list below, and press button to view top terms in these fields. No selection means all fields.

Available fields and term counts per field:

Name	Term count	%	Decoder
EPISERVER_SEARCH_CULTURE	3	0.58 %	string utf8
EPISERVER_SEARCH_DEFAULT	184	35.73 %	string utf8
EPISERVER_SEARCH_DISPLAYTE	164	31.84 %	string utf8
EPISERVER_SEARCH_ID	29	5.63 %	string utf8
EPISERVER_SEARCH_ITEMSTATU	1	0.19 %	string utf8
EPISERVER_SEARCH_METADATA	0	0.00 %	string utf8
EPISERVER_SEARCH_MODIFIED	22	4.27 %	string utf8
EPISERVER_SEARCH_NAMEDIND	1	0.19 %	string utf8
EPISERVER_SEARCH_PUBLICATI(	1	0.19 %	string utf8
EPISERVER_SEARCH_PUBLICATI(	1	0.19 %	string utf8
EPISERVER_SEARCH_REFERENC	1	0.19 %	string utf8
EPISERVER_SEARCH_TITLE	33	6.41 %	string utf8
EPISERVER_SEARCH_TYPE	18	3.5 %	string utf8
EPISERVER_SEARCH_URI	0	0.00 %	string utf8
EPISERVER_SEARCH_VIRTUALPA	29	5.63 %	string utf8

Number of top terms: 50

Hint: use Shift-Click to select ranges, or Ctrl-Click to select multiple fields (or unselect all).

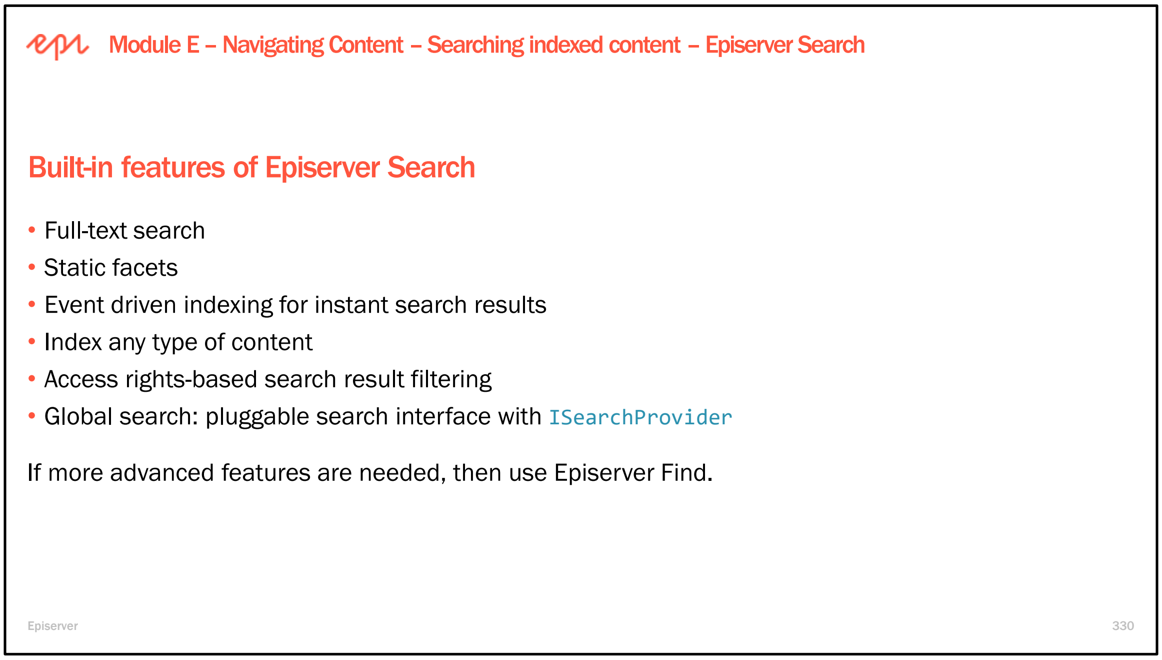
Tokens marked in red indicate decoding errors, likely due to a mismatched decoder.

Select a field and set its value decoder: string utf8

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	29	EPISERVER_	no
2	29	EPISERVER_	1
3	29	EPISERVER_	EPiServer.C
4	29	EPISERVER_	no
5	29	EPISERVER_	[[G:Everyone
6	29	EPISERVER_	
7	29	EPISERVER_	default
8	29	EPISERVER_	EPiServer.C
9	29	EPISERVER_	[[G:Administ
10	25	EPISERVER_	admin
11	25	EPISERVER_	Admin
12	25	EPISERVER_	[[G:WebAdm
13	15	EPISERVER_	en
14	14	EPISERVER_	EPiServer.C
15	11	EPISERVER_	EPiServer.C
16	11	EPISERVER_	alloy
17	10	EPISERVER_	the

Index name: C:\Episerv...lloyTraining\App_Data\Index



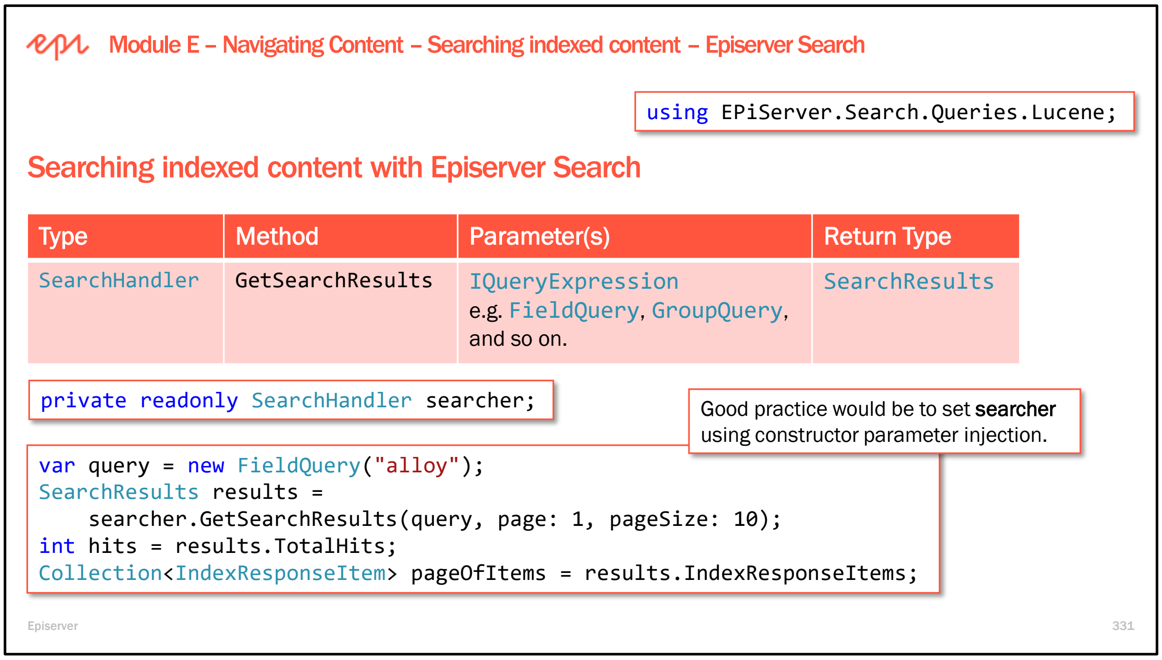
Module E – Navigating Content – Searching indexed content – Episerver Search

Built-in features of Episerver Search

- Full-text search
- Static facets
- Event driven indexing for instant search results
- Index any type of content
- Access rights-based search result filtering
- Global search: pluggable search interface with `ISearchProvider`

If more advanced features are needed, then use Episerver Find.

Episerver 330



Module E – Navigating Content – Searching indexed content – Episerver Search

```
using EPiServer.Search.Queries.Lucene;
```

Searching indexed content with Episerver Search

Type	Method	Parameter(s)	Return Type
`SearchHandler`	`GetSearchResults`	`IQueryExpression` e.g. `FieldQuery`, `GroupQuery`, and so on.	`SearchResults`

```
private readonly SearchHandler searcher;
```

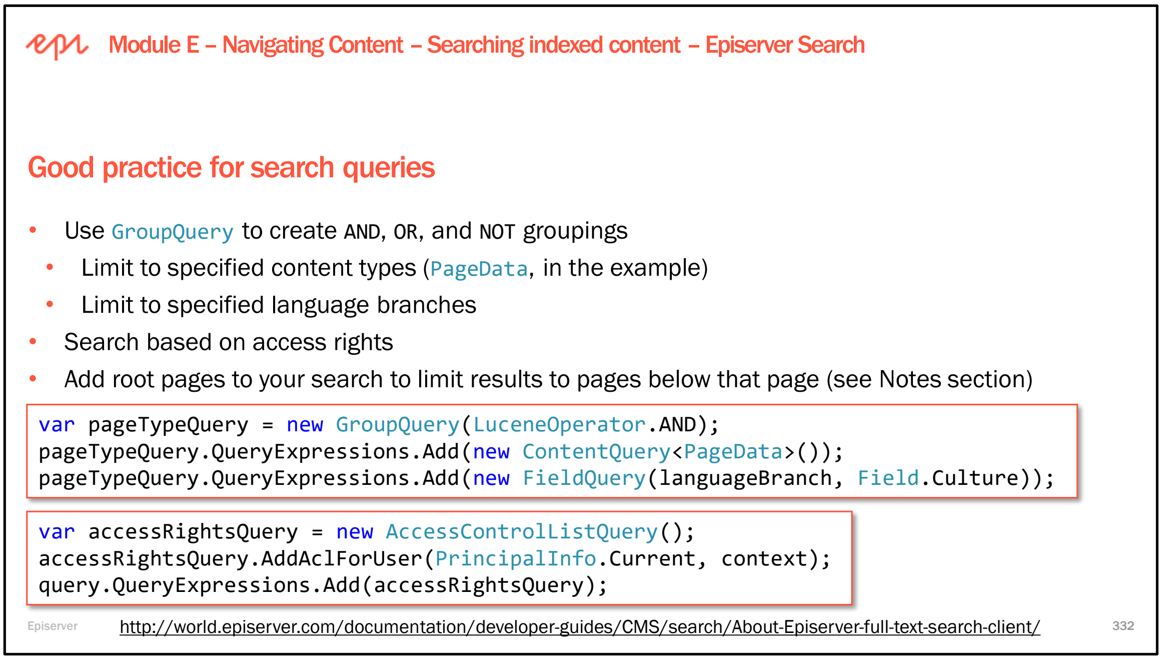
Good practice would be to set **searcher** using constructor parameter injection.

```
var query = new FieldQuery("alloy");
SearchResults results =
    searcher.GetSearchResults(query, page: 1, pageSize: 10);
int hits = results.TotalHits;
Collection<IndexResponseItem> pageOfItems = results.IndexResponseItems;
```

Episerver 331

Performing a simple query search

- Build a query using EPiServer.Search.Queries.Lucene.FieldQuery(string q)
- Pass the query into SearchHandler.Instance.GetSearchResults(fieldQuery)
- Convert these results to a EPiServer.Search.IndexResponseItem List
- IndexResponseItem contains Content Guid should you want access to the entire Content object



Module E – Navigating Content – Searching indexed content – Episerver Search

Good practice for search queries

- Use `GroupQuery` to create `AND`, `OR`, and `NOT` groupings
  - Limit to specified content types (`PageData`, in the example)
  - Limit to specified language branches
- Search based on access rights
- Add root pages to your search to limit results to pages below that page (see Notes section)

```
var pageTypeQuery = new GroupQuery(LuceneOperator.AND);
pageTypeQuery.QueryExpressions.Add(new ContentQuery<PageData>());
pageTypeQuery.QueryExpressions.Add(new FieldQuery(languageBranch, Field.Culture));
```

```
var accessRightsQuery = new AccessControlListQuery();
accessRightsQuery.AddAclForUser(PrincipalInfo.Current, context);
query.QueryExpressions.Add(accessRightsQuery);
```

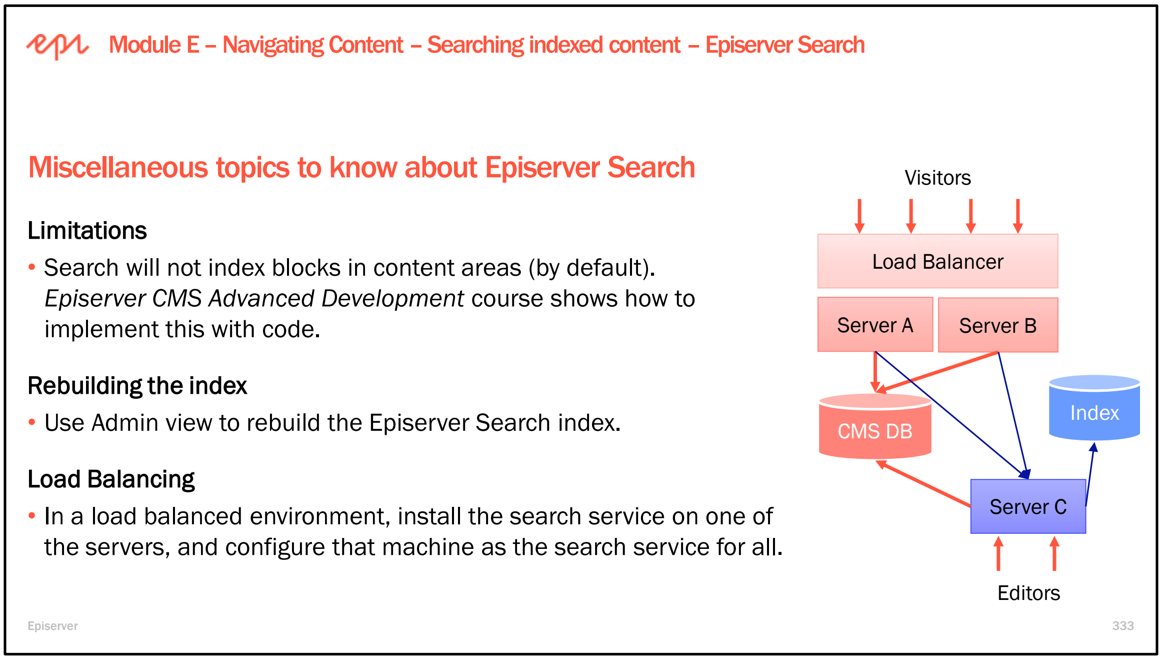
Episerver http://world.episerver.com/documentation/developer-guides/CMS/search/About-Episerver-full-text-search-client/ 332

See ~\Business\SearchService.cs in the Alloy (MVC) project template for more details.

Types of queries
- AccessControlQuery
- CategoryQuery
- CreatedDateRangeQuery
- FieldQuery
- FuzzyQuery
- GroupQuery
- ItemStatusQuery
- ModifiedDateRangeQuery
- ProximityQuery
- RangeQuery
- TermBoostQuery
- VirtualPathQuery

EPiServer - Simple search and shared blocks
https://www.dcaric.com/blog/episerver-simple-search-and-shared-blocks

Extending EPiServer search - part 2
https://www.dcaric.com/blog/extending-episerver-search-part-2



Module E – Navigating Content – Searching indexed content – Episerver Search

Miscellaneous topics to know about Episerver Search

Limitations

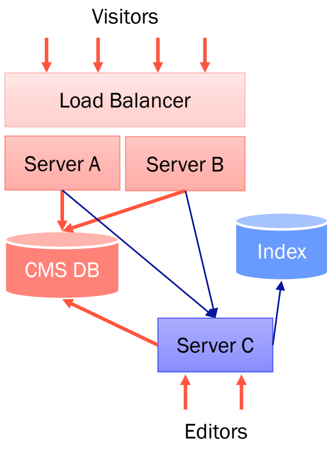
- Search will not index blocks in content areas (by default). *Episerver CMS Advanced Development* course shows how to implement this with code.

Rebuilding the index

- Use Admin view to rebuild the Episerver Search index.

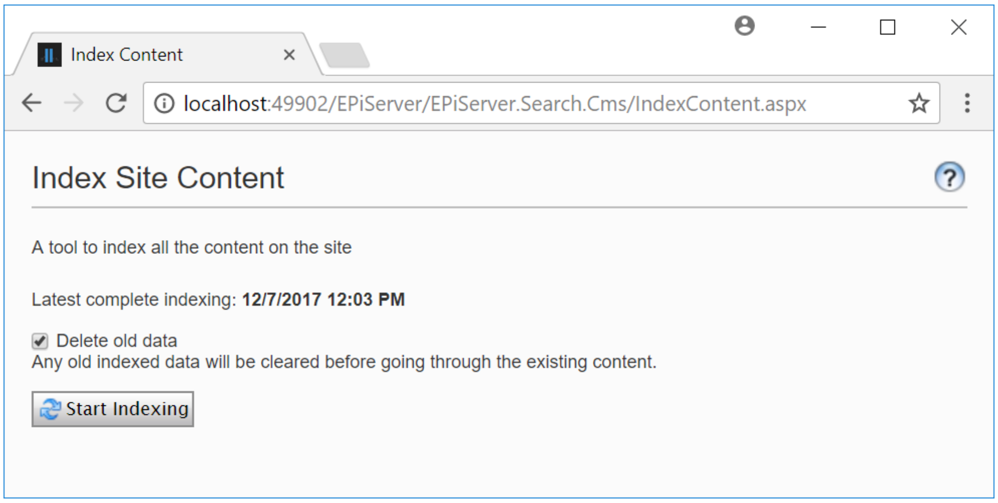
Load Balancing

- In a load balanced environment, install the search service on one of the servers, and configure that machine as the search service for all.



Episerver 333

Manually re-indexing site content with a hidden feature, as shown in the following screenshot:

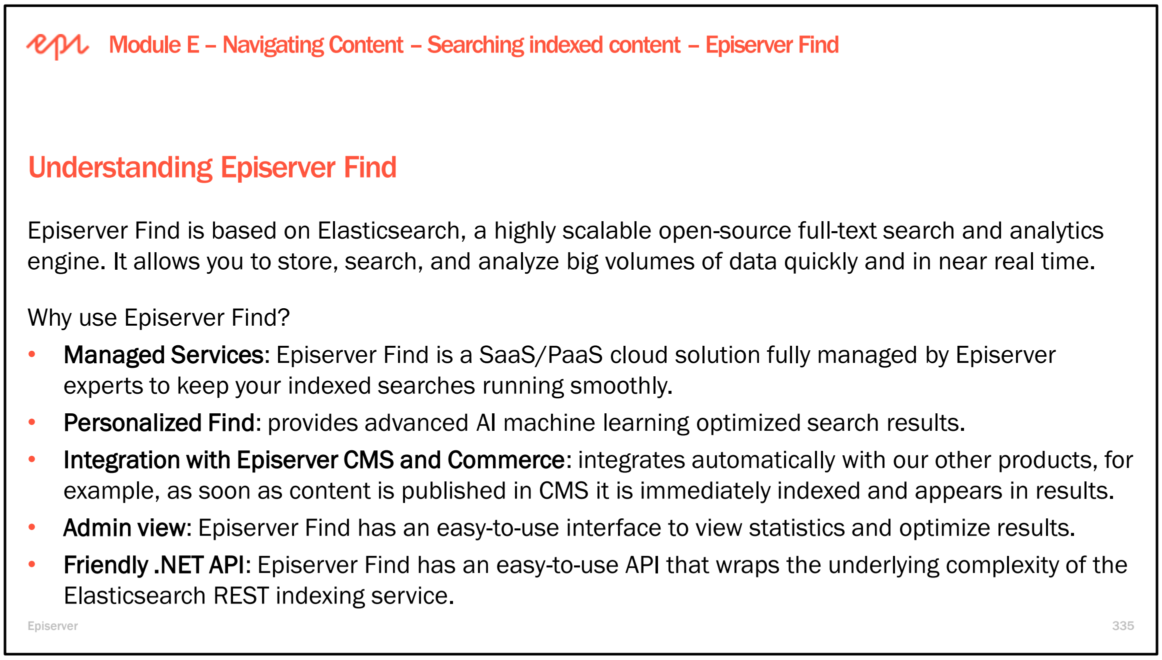

Index Content
localhost:49902/EPiServer/EPiServer.Search.Cms/IndexContent.aspx

Index Site Content

A tool to index all the content on the site

Latest complete indexing: **12/7/2017 12:03 PM**

☑ Delete old data
Any old indexed data will be cleared before going through the existing content.



Module E – Navigating Content – Searching indexed content – Episerver Find

Understanding Episerver Find

Episerver Find is based on Elasticsearch, a highly scalable open-source full-text search and analytics engine. It allows you to store, search, and analyze big volumes of data quickly and in near real time.

Why use Episerver Find?

- **Managed Services**: Episerver Find is a SaaS/PaaS cloud solution fully managed by Episerver experts to keep your indexed searches running smoothly.
- **Personalized Find**: provides advanced AI machine learning optimized search results.
- **Integration with Episerver CMS and Commerce**: integrates automatically with our other products, for example, as soon as content is published in CMS it is immediately indexed and appears in results.
- **Admin view**: Episerver Find has an easy-to-use interface to view statistics and optimize results.
- **Friendly .NET API**: Episerver Find has an easy-to-use API that wraps the underlying complexity of the Elasticsearch REST indexing service.

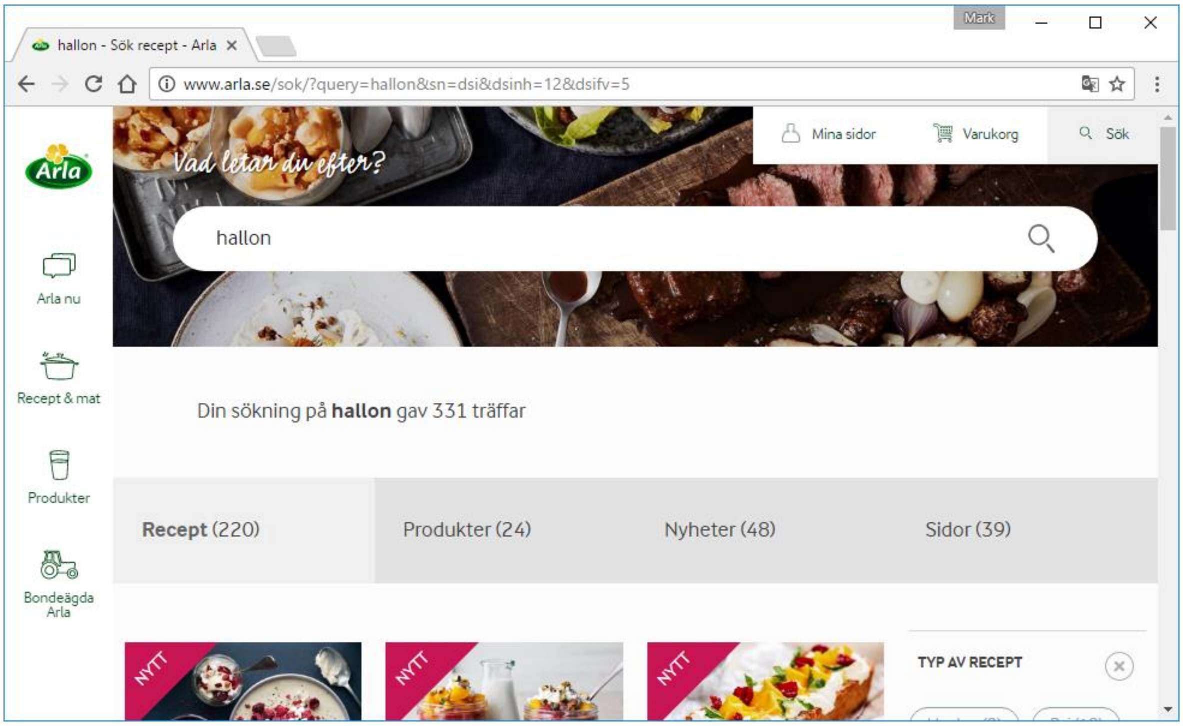
Episerver 335

Sites that use Episerver Find

Arla
http://www.arla.se/

Small Luxury Hotels of the World
http://www.slh.com/

SMALL LUXURY HOTELS OF THE WORLD™
Independently minded

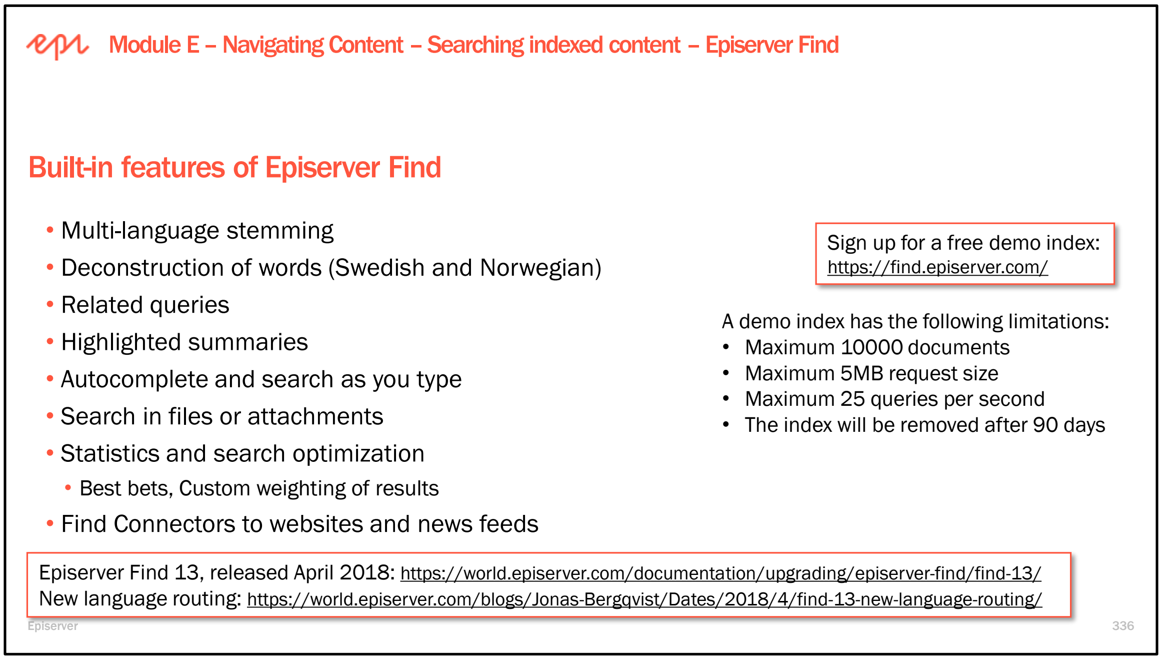


hallon - Sök recept - Arla

www.arla.se/sok/?query=hallon&sn=dsi&dsinh=12&dsifv=5

Din sökning på **hallon** gav 331 träffar

Recept (220) Produkter (24) Nyheter (48) Sidor (39)



Module E – Navigating Content – Searching indexed content – Episerver Find

Built-in features of Episerver Find

- Multi-language stemming
- Deconstruction of words (Swedish and Norwegian)
- Related queries
- Highlighted summaries
- Autocomplete and search as you type
- Search in files or attachments
- Statistics and search optimization
  - Best bets, Custom weighting of results
- Find Connectors to websites and news feeds

Sign up for a free demo index:
https://find.episerver.com/

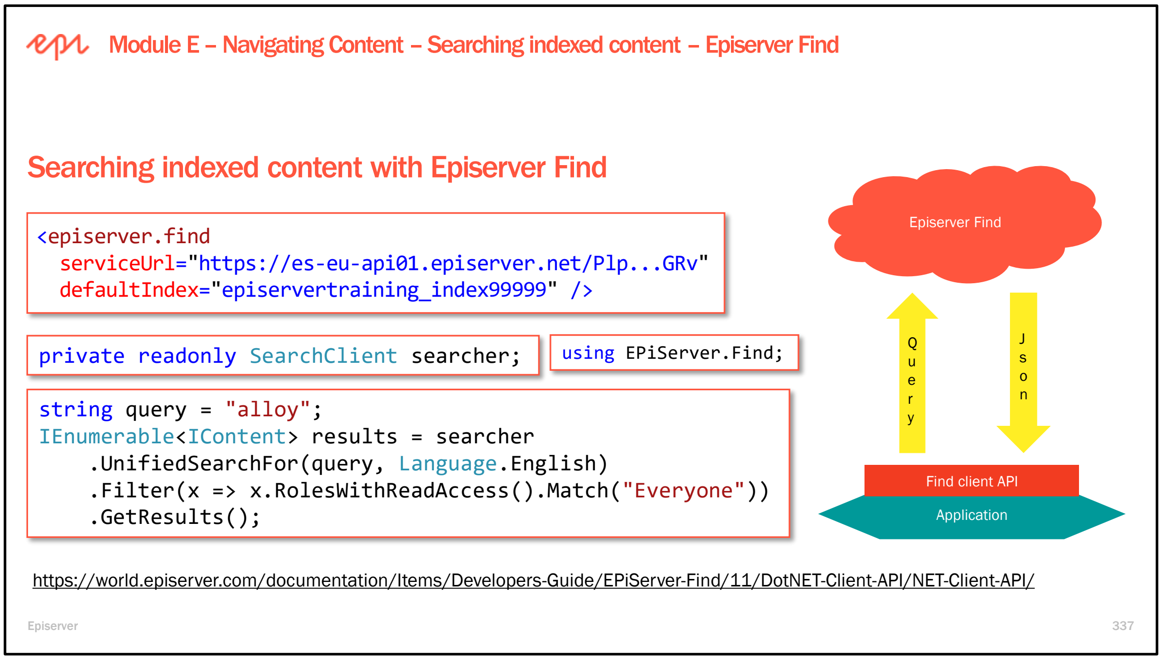
A demo index has the following limitations:
- Maximum 10000 documents
- Maximum 5MB request size
- Maximum 25 queries per second
- The index will be removed after 90 days

Episerver Find 13, released April 2018: https://world.episerver.com/documentation/upgrading/episerver-find/find-13/
New language routing: https://world.episerver.com/blogs/Jonas-Bergqvist/Dates/2018/4/find-13-new-language-routing/

Episerver 336

Decompounding
- cheeseburger → cheese burger
- football → foot ball
- blårutigskjortan → blå rutig skjorta n (the blue checkered shirt)
- banan → bana n (the trajectory)
- banan → banana

Installing Episerver Find
- Installed through NuGet
- Requires additional license + create an index in cloud service
- Support for Episerver CMS 6 and higher
- Support for Episerver Commerce
- Requires the full .NET framework (not Client Profile)
- Depends on JSON.NET (Newtonsoft.Json.dll)



Module E – Navigating Content – Searching indexed content – Episerver Find

Searching indexed content with Episerver Find

```
<episerver.find
  serviceUrl="https://es-eu-api01.episerver.net/Plp...GRv"
  defaultIndex="episervertraining_index99999" />
```

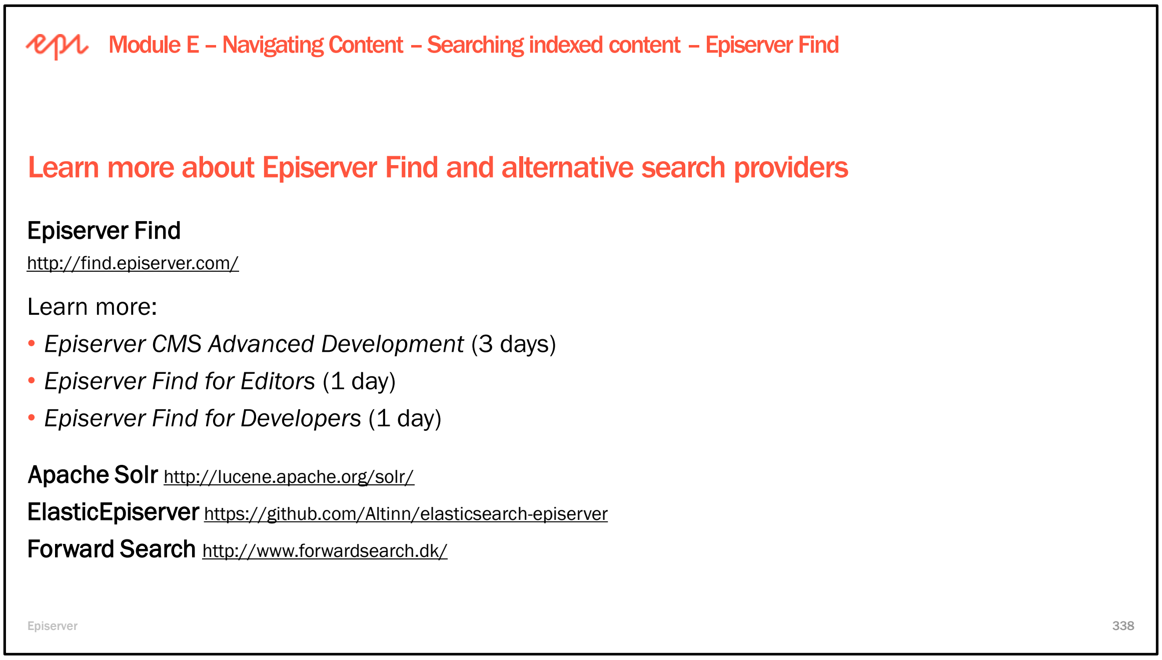
```
private readonly SearchClient searcher;
```

```
using EPiServer.Find;
```

```
string query = "alloy";
IEnumerable<IContent> results = searcher
    .UnifiedSearchFor(query, Language.English)
    .Filter(x => x.RolesWithReadAccess().Match("Everyone"))
    .GetResults();
```

https://world.episerver.com/documentation/Items/Developers-Guide/EPiServer-Find/11/DotNET-Client-API/NET-Client-API/

Episerver 337



Module E – Navigating Content – Searching indexed content – Episerver Find

Learn more about Episerver Find and alternative search providers

Episerver Find
http://find.episerver.com/

Learn more:

- *Episerver CMS Advanced Development* (3 days)
- *Episerver Find for Editors* (1 day)
- *Episerver Find for Developers* (1 day)

Apache Solr http://lucene.apache.org/solr/
ElasticEpiserver https://github.com/Altinn/elasticsearch-episerver
Forward Search http://www.forwardsearch.dk/

Episerver 338

Module E – Navigating Content

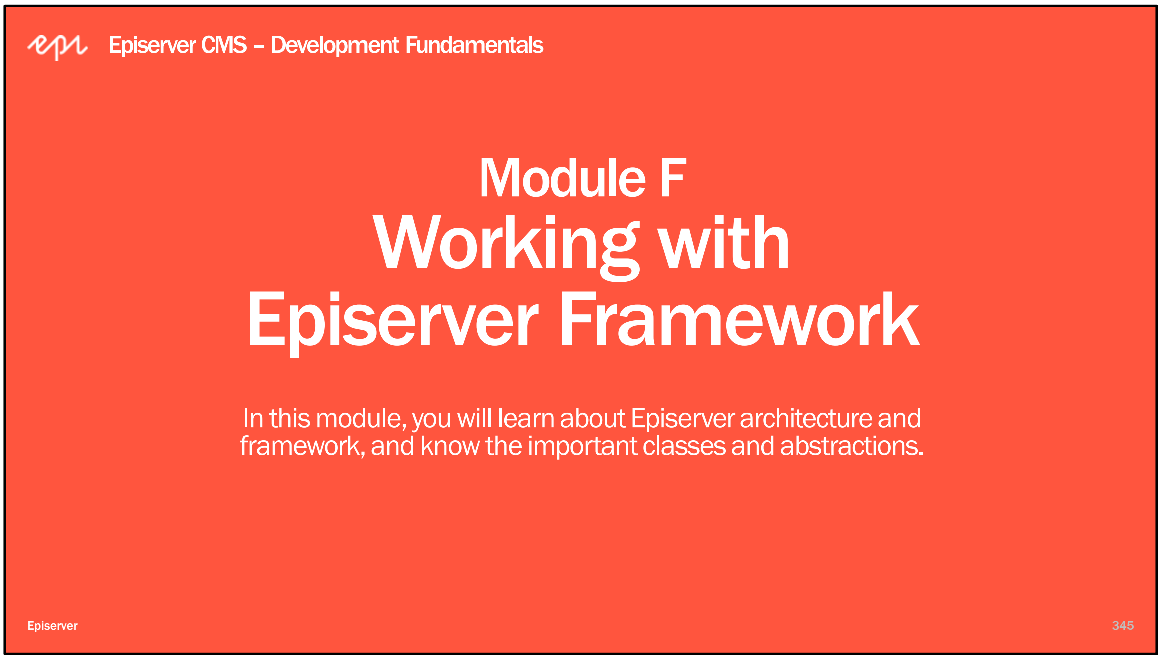
Exercises E1 to E5 – Navigating content

Estimated time: 60 minutes

Prerequisites: Exercises B1 – B4.

1. Creating a page listing block
2. Creating a news landing page
3. Improving navigation menus
4. Creating a search page for visitors using Episerver Find or Episerver Search
5. Adding a search box to the top navigation menu

Episerver 339

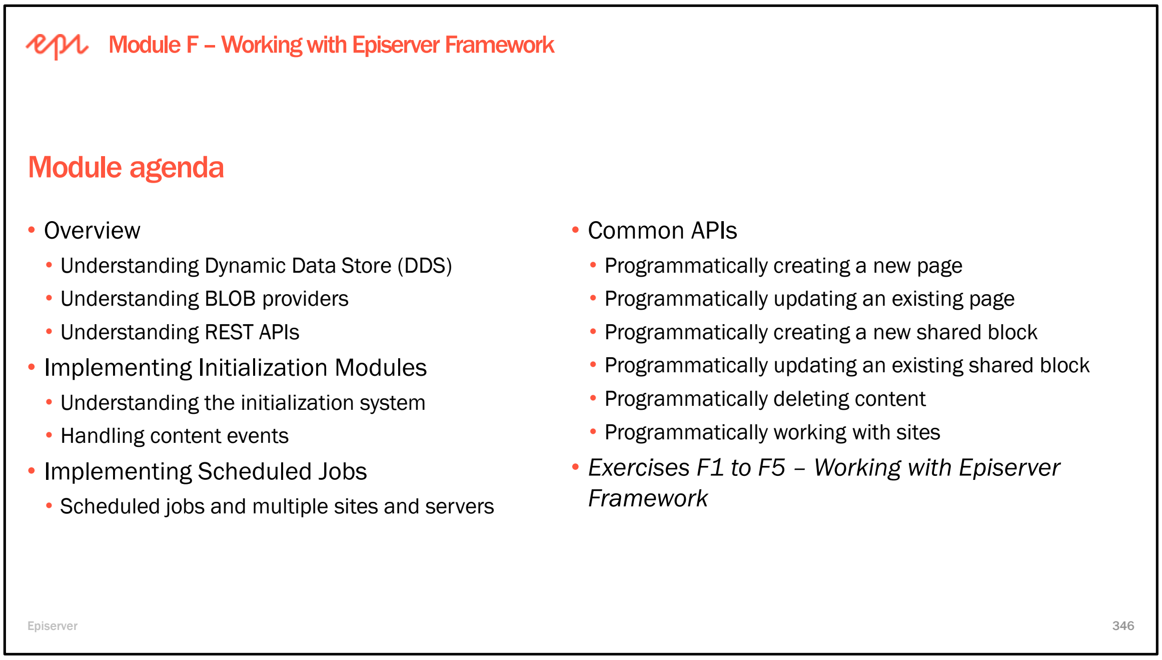


Episerver CMS – Development Fundamentals

Module F
Working with Episerver Framework

In this module, you will learn about Episerver architecture and framework, and know the important classes and abstractions.

Episerver 345

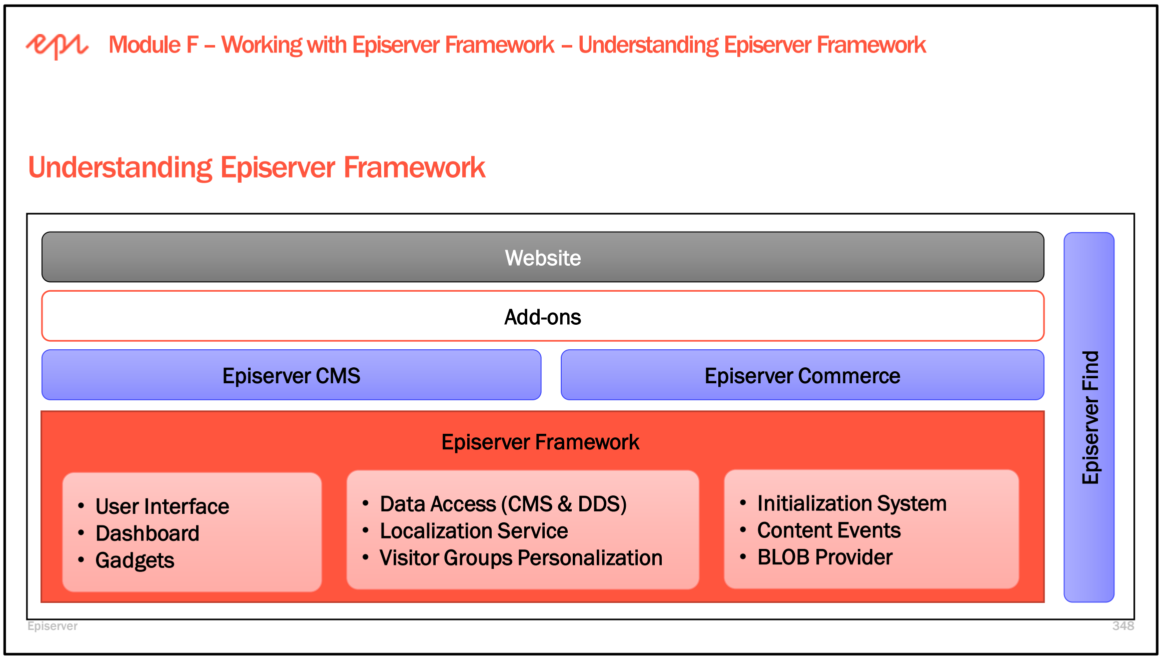


Module F – Working with Episerver Framework

Module agenda

- Overview
  - Understanding Dynamic Data Store (DDS)
  - Understanding BLOB providers
  - Understanding REST APIs
- Implementing Initialization Modules
  - Understanding the initialization system
  - Handling content events
- Implementing Scheduled Jobs
  - Scheduled jobs and multiple sites and servers
- Common APIs
  - Programmatically creating a new page
  - Programmatically updating an existing page
  - Programmatically creating a new shared block
  - Programmatically updating an existing shared block
  - Programmatically deleting content
  - Programmatically working with sites
- *Exercises F1 to F5 – Working with Episerver Framework*

Episerver 346



Module F – Working with Episerver Framework – Understanding Episerver Framework

Understanding Episerver Framework

Episerver 348

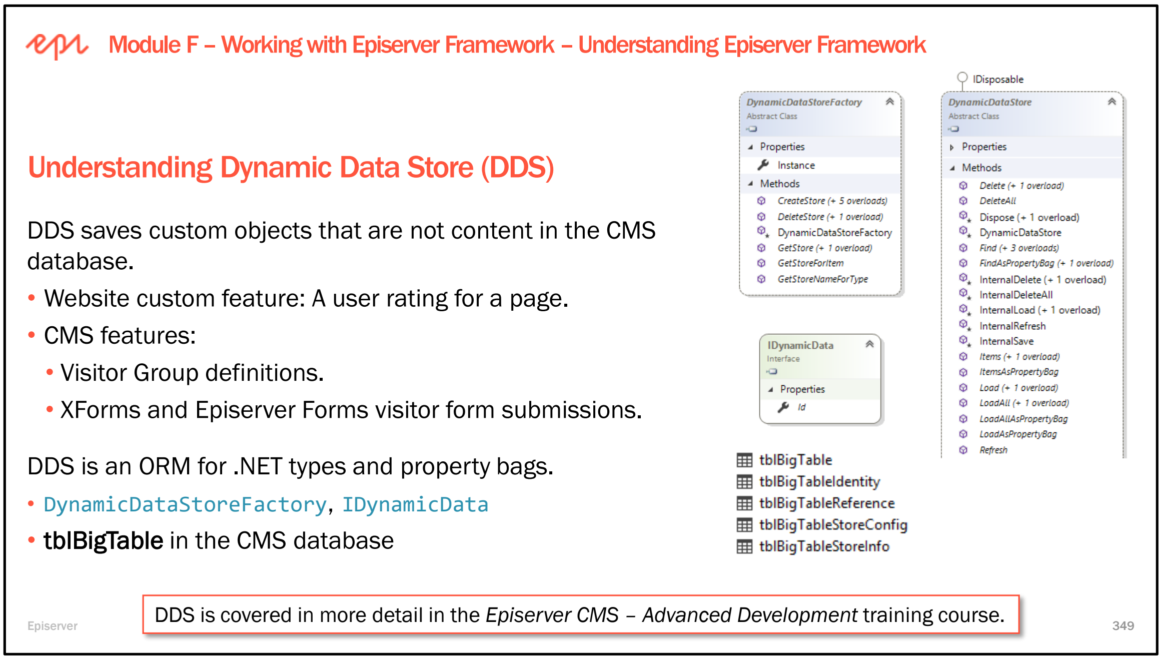
The areas of the platform that this course is focusing on are the Episerver CMS product and the Website built upon it using the CMS API.
The Episerver Framework is briefly discussed in this section. More information is available in the Episerver Framework SDK.

The key functions of the Episerver Framework:
Contains common UI and API functionality intended to be used by all Episerver products
Handles license management

- All product licenses are contained in one file: License.config.

Includes support for communication between servers in a load balanced setup, using global event handling.

- Examples of update of nodes in load-balanced environments:
    - Content is added or updated in a page or block
    - The editor updates a file



Module F – Working with Episerver Framework – Understanding Episerver Framework

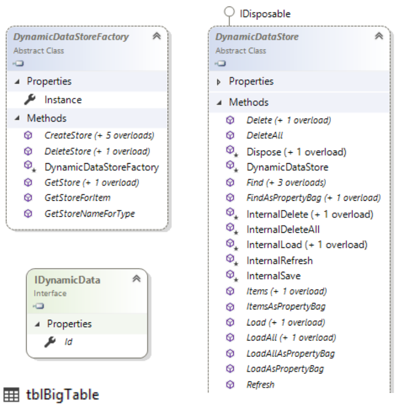
Understanding Dynamic Data Store (DDS)

DDS saves custom objects that are not content in the CMS database.

- Website custom feature: A user rating for a page.
- CMS features:
  - Visitor Group definitions.
  - XForms and Episerver Forms visitor form submissions.

DDS is an ORM for .NET types and property bags.

- `DynamicDataStoreFactory`, `IDynamicData`
- **tblBigTable** in the CMS database



tblBigTable
tblBigTableIdentity
tblBigTableReference
tblBigTableStoreConfig
tblBigTableStoreInfo

DDS is covered in more detail in the *Episerver CMS – Advanced Development* training course.

Episerver 349

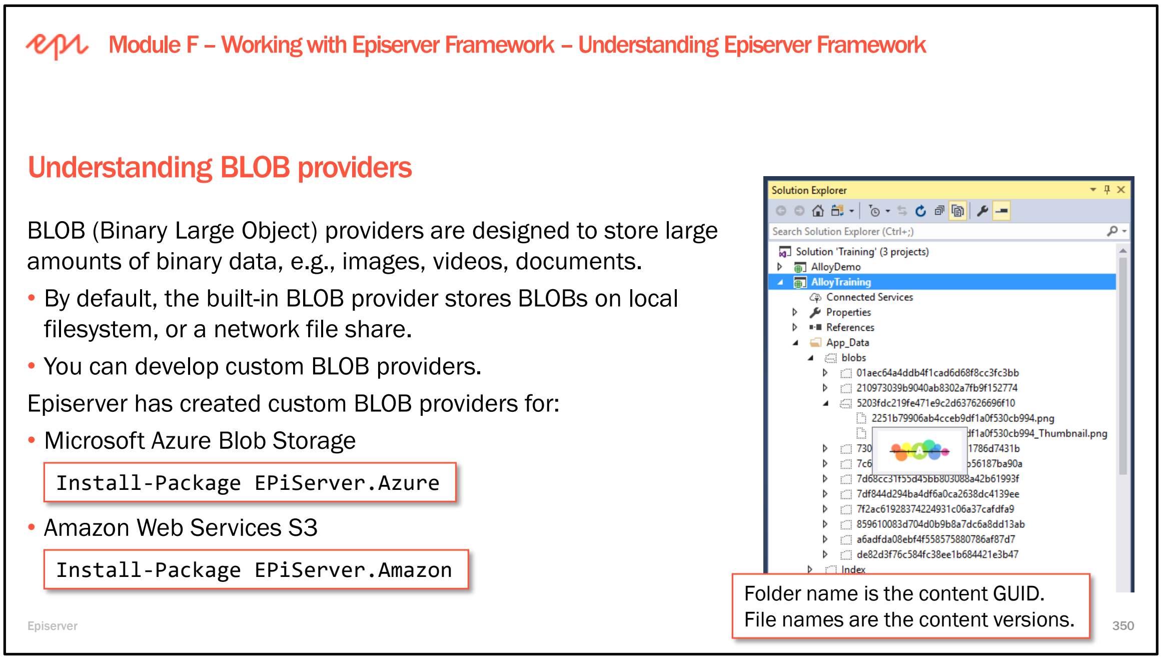
Dynamic Data Store offers an API and infrastructure for saving, loading and searching of both compile time data types (.NET object instances) and runtime data types (property bags) to the database

- Dynamic Data Store is essentially an object-relational mapper

Stores are created, obtained and deleted using the DynamicDataStoreFactory class.

- The class has a single instance which can be obtained from the static Instance property.

Dynamic Data Store uses the 'big table' approach to storing data (the default DDS "big table" is called tblBigTable).



Module F – Working with Episerver Framework – Understanding Episerver Framework

Understanding BLOB providers

BLOB (Binary Large Object) providers are designed to store large amounts of binary data, e.g., images, videos, documents.

- By default, the built-in BLOB provider stores BLOBs on local filesystem, or a network file share.
- You can develop custom BLOB providers.

Episerver has created custom BLOB providers for:

- Microsoft Azure Blob Storage

```
Install-Package EPiServer.Azure
```

- Amazon Web Services S3

```
Install-Package EPiServer.Amazon
```

```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Training' (3 projects)
  AlloyDemo
  AlloyTraining
    Connected Services
    Properties
    References
    App_Data
      blobs
        01aec64a4ddb4f1cad6d68f8cc3fc3bb
        210973039b9040ab8302a7fb9f152774
        5203fdc219fe471e9c2d637626696f10
          2251b79906ab4cceb9df1a0f530cb994.png
          df1a0f530cb994_Thumbnail.png
        730                1786d7431b
        7c6                56187ba90a
        7d68cc31f55d45bb803088a42b61993f
        7df844d294ba4df6a0ca2638dc4139ee
        7f2ac61928374224931c06a37cafdfa9
        859610083d704d0b9b8a7dc6a8dd13ab
        a6adfda08ebf4f558575880786af87d7
        de82d3f76c584fc38ee1b684421e3b47
      Index
```

Folder name is the content GUID.
File names are the content versions.

Episerver 350

BLOB (Binary Large Object) providers is a framework designed to store large amounts of binary data in a more optimized and cost-effective solution such as cloud storage, instead of in the database. The Episerver platform supports BLOB storage of assets using a provider-based setup, and has a built-in file BLOB provider. You have the following options:

- Built-in BLOB provider. Episerver has a built-in BLOB provider for media files such as images, videos and documents. By default this provider will store files on local disc or a file share which will be defined during installation.
- Customized BLOB provider. You can also develop and configure your own customized BLOB provider for your specific hosting environment. As an example, BLOB providers for Microsoft Azure and Amazon Web Services are available via the Episerver Nuget feed.

Module F – Working with Episerver Framework – Understanding Episerver Framework

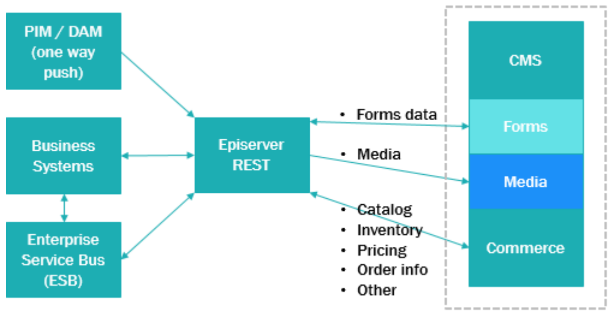
Understanding Service API

Service API enables integration with external systems such as PIM, DAM and ERP.

Use Service API with **Episerver CMS** to:
- Import and export of "episerverdata" files.

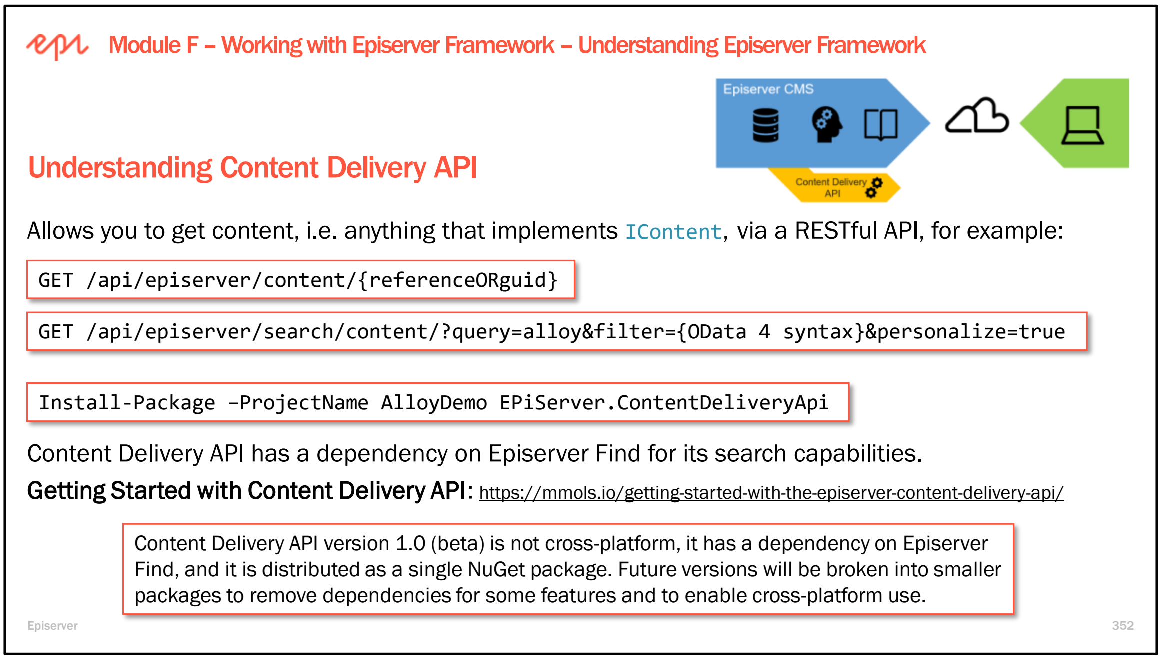
Use Service API with **Episerver Forms** to:
- Import and export form submissions.

User Service API with **Media** to:
- Import media assets.



Use Service API with **Episerver Commerce** to:
- Bulk import, export, and asset link between media and catalog data.
- Perform REST CRUD operations on catalogs, nodes, entries, and warehouses.

Episerver 351



Module F – Working with Episerver Framework – Understanding Episerver Framework

Episerver CMS
Content Delivery API

Understanding Content Delivery API

Allows you to get content, i.e. anything that implements `IContent`, via a RESTful API, for example:

```
GET /api/episerver/content/{referenceORguid}
```

```
GET /api/episerver/search/content/?query=alloy&filter={OData 4 syntax}&personalize=true
```

```
Install-Package -ProjectName AlloyDemo EPiServer.ContentDeliveryApi
```

Content Delivery API has a dependency on Episerver Find for its search capabilities.

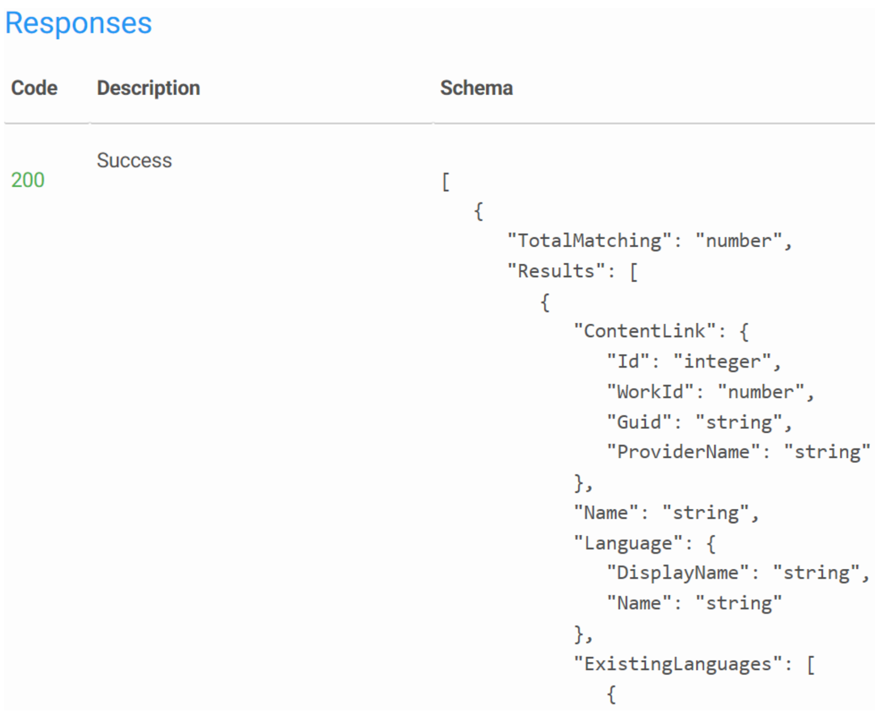
Getting Started with Content Delivery API: https://mmols.io/getting-started-with-the-episerver-content-delivery-api/

Content Delivery API version 1.0 (beta) is not cross-platform, it has a dependency on Episerver Find, and it is distributed as a single NuGet package. Future versions will be broken into smaller packages to remove dependencies for some features and to enable cross-platform use.

Episerver 352

Episerver Content Api
https://sdk.episerver.com/ContentDeliveryAPI/Index.html

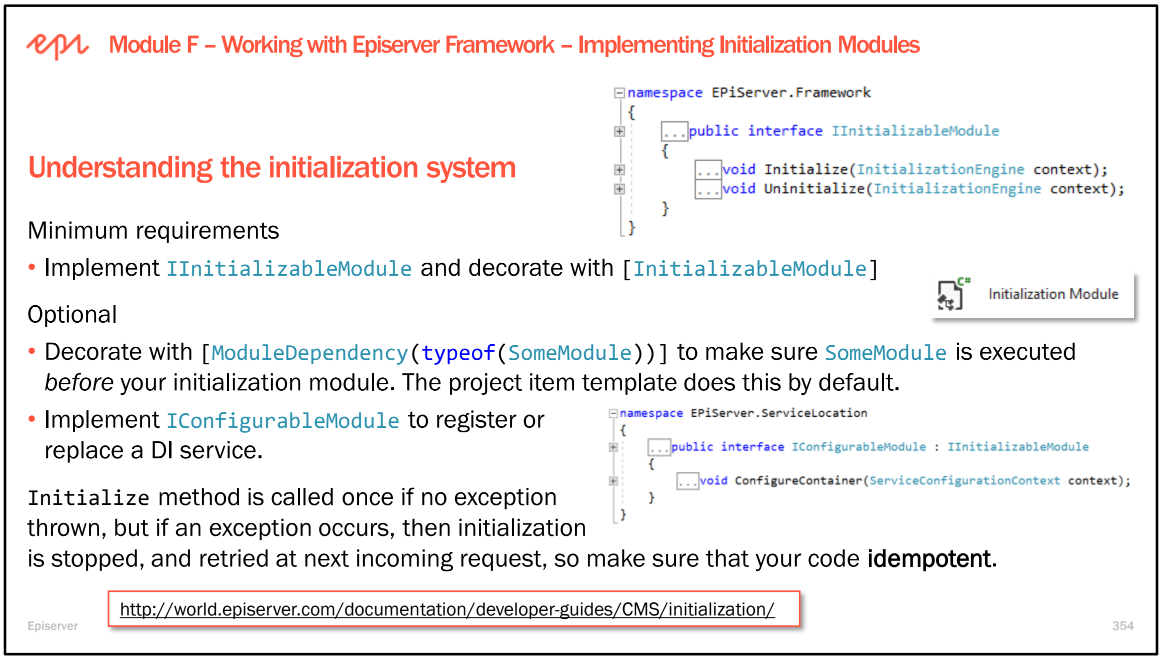
Content Delivery API
https://world.episerver.com/documentation/developer-guides/CMS/Content/content-delivery-api/



Responses

Code	Description	Schema
200	Success	see below

```
[
  {
    "TotalMatching": "number",
    "Results": [
      {
        "ContentLink": {
          "Id": "integer",
          "WorkId": "number",
          "Guid": "string",
          "ProviderName": "string"
        },
        "Name": "string",
        "Language": {
          "DisplayName": "string",
          "Name": "string"
        },
        "ExistingLanguages": [
          {
```



Module F – Working with Episerver Framework – Implementing Initialization Modules

Understanding the initialization system

```
namespace EPiServer.Framework
{
    ...public interface IInitializableModule
    {
        ...void Initialize(InitializationEngine context);
        ...void Uninitialize(InitializationEngine context);
    }
}
```

Minimum requirements

- Implement `IInitializableModule` and decorate with `[InitializableModule]`

Optional

- Decorate with `[ModuleDependency(typeof(SomeModule))]` to make sure `SomeModule` is executed *before* your initialization module. The project item template does this by default.
- Implement `IConfigurableModule` to register or replace a DI service.

```
namespace EPiServer.ServiceLocation
{
    ...public interface IConfigurableModule : IInitializableModule
    {
        ...void ConfigureContainer(ServiceConfigurationContext context);
    }
}
```

`Initialize` method is called once if no exception thrown, but if an exception occurs, then initialization is stopped, and retried at next incoming request, so make sure that your code **idempotent.**

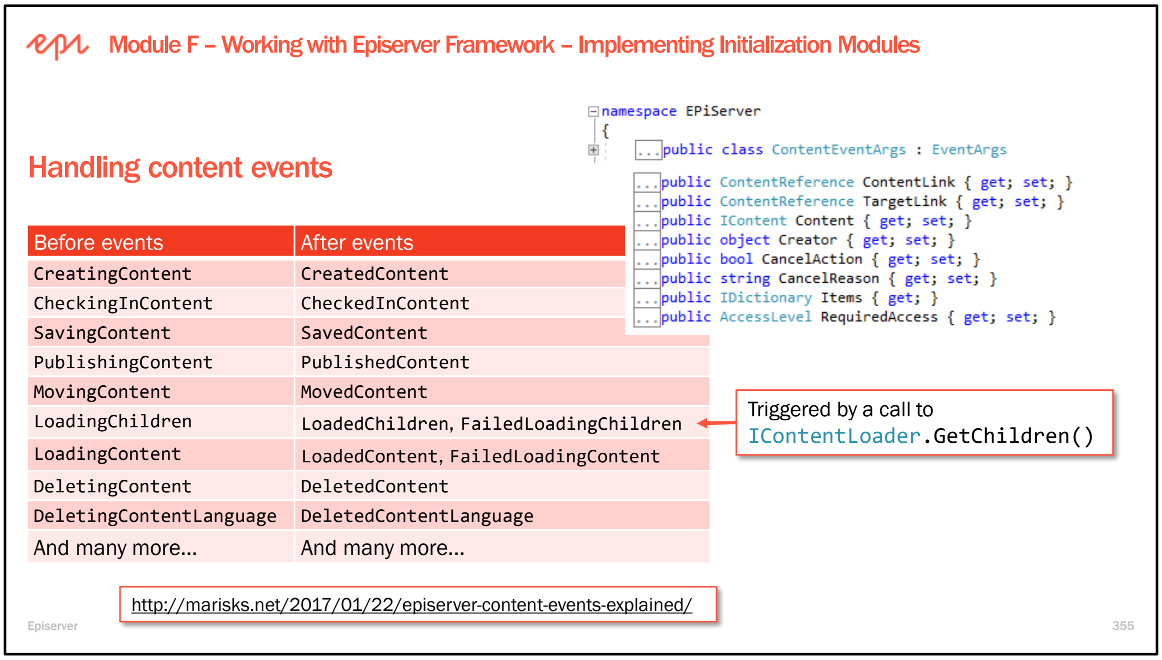
http://world.episerver.com/documentation/developer-guides/CMS/initialization/

Episerver 354

Examples of modules:
ClassFactoryInitialization.cs in Alloy: Use their own control to render content areas.
Hooks up to the initialization to register their own class to be used when content areas are rendered.
Dependency sorting
The initialization system has a dependency sorting algorithm to decide the execution order of the modules
Example: If you want to log when pages are saved in your website, the logging and DataFactory has to exist. To ensure this you use the dependency sorting algorithm by using the ModuleDependency attribute on your class: [ModuleDependency(typeof(ModuleThatIDependOn))].
Execution engine
The execution engine hooks into ASP.NET to handle re-execution of initialization modules in the face of exceptions during startup.

Example: removing the suggested page types feature
When adding a new page, the first group is named **Suggested Page Types** and contains recently used page types. You can remove this by ejecting the implementation of **IContentTypeAdvisor** like this:

```
[InitializableModule]
[ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
public class RemoveSuggestedPageTypesInitializationModule : IConfigurableModule
{
    public void ConfigureContainer(ServiceConfigurationContext context)
    {
        context.Container.EjectAllInstancesOf<IContentTypeAdvisor>();
    }
    public void Initialize(InitializationEngine context) { }
    public void Uninitialize(InitializationEngine context) { }
}
```



Module F – Working with Episerver Framework – Implementing Initialization Modules

Handling content events

```
namespace EPiServer
{
    public class ContentEventArgs : EventArgs

    public ContentReference ContentLink { get; set; }
    public ContentReference TargetLink { get; set; }
    public IContent Content { get; set; }
    public object Creator { get; set; }
    public bool CancelAction { get; set; }
    public string CancelReason { get; set; }
    public IDictionary Items { get; }
    public AccessLevel RequiredAccess { get; set; }
```

Before events	After events
CreatingContent	CreatedContent
CheckingInContent	CheckedInContent
SavingContent	SavedContent
PublishingContent	PublishedContent
MovingContent	MovedContent
LoadingChildren	LoadedChildren, FailedLoadingChildren
LoadingContent	LoadedContent, FailedLoadingContent
DeletingContent	DeletedContent
DeletingContentLanguage	DeletedContentLanguage
And many more…	And many more…

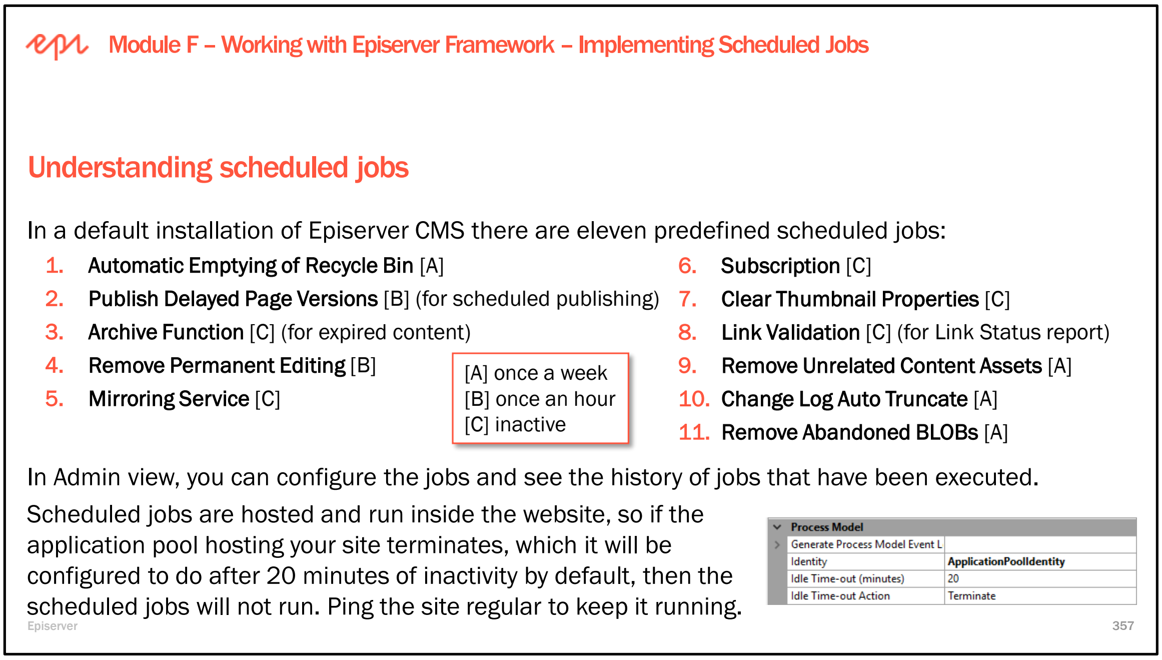
Triggered by a call to IContentLoader.GetChildren()

http://marisks.net/2017/01/22/episerver-content-events-explained/

Episerver 355

```
[InitializableModule]
[ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
public class PreventPublishingInitializationModule : IInitializableModule
{
    private bool executed = false;
    private IContentEvents events;
    public void Initialize(InitializationEngine context)
    {
        if (!executed)
        {
            events = ServiceLocator.Current.GetInstance<IContentEvents>();
            events.PublishingContent += Events_PublishingContent;
            executed = true;
        }
    }

    private void Events_PublishingContent(object sender, EPiServer.ContentEventArgs e)
    {
        if ((e.Content as PageData).Name.ToLower().Contains("bad word"))
        {
            e.CancelAction = true;
            e.CancelReason = "Content names cannot contain \"bad word\".";
        }
    }

    public void Uninitialize(InitializationEngine context)
    {
        events.PublishingContent -= Events_PublishingContent;
    }
}
```

```
private void Events_MovedContent(object sender, EPiServer.ContentEventArgs e)
{
    // does nothing because you cannot cancel an "after" event
    e.CancelAction = true;
    e.CancelReason = "This does nothing!";
}
```



Module F – Working with Episerver Framework – Implementing Scheduled Jobs

Understanding scheduled jobs

In a default installation of Episerver CMS there are eleven predefined scheduled jobs:

1. **Automatic Emptying of Recycle Bin** [A]
2. **Publish Delayed Page Versions** [B] (for scheduled publishing)
3. **Archive Function** [C] (for expired content)
4. **Remove Permanent Editing** [B]
5. **Mirroring Service** [C]
6. **Subscription** [C]
7. **Clear Thumbnail Properties** [C]
8. **Link Validation** [C] (for Link Status report)
9. **Remove Unrelated Content Assets** [A]
10. **Change Log Auto Truncate** [A]
11. **Remove Abandoned BLOBs** [A]

[A] once a week
[B] once an hour
[C] inactive

In Admin view, you can configure the jobs and see the history of jobs that have been executed.

Scheduled jobs are hosted and run inside the website, so if the application pool hosting your site terminates, which it will be configured to do after 20 minutes of inactivity by default, then the scheduled jobs will not run. Ping the site regular to keep it running.

Process Model	
Generate Process Model Event L	
Identity	**ApplicationPoolIdentity**
Idle Time-out (minutes)	20
Idle Time-out Action	Terminate

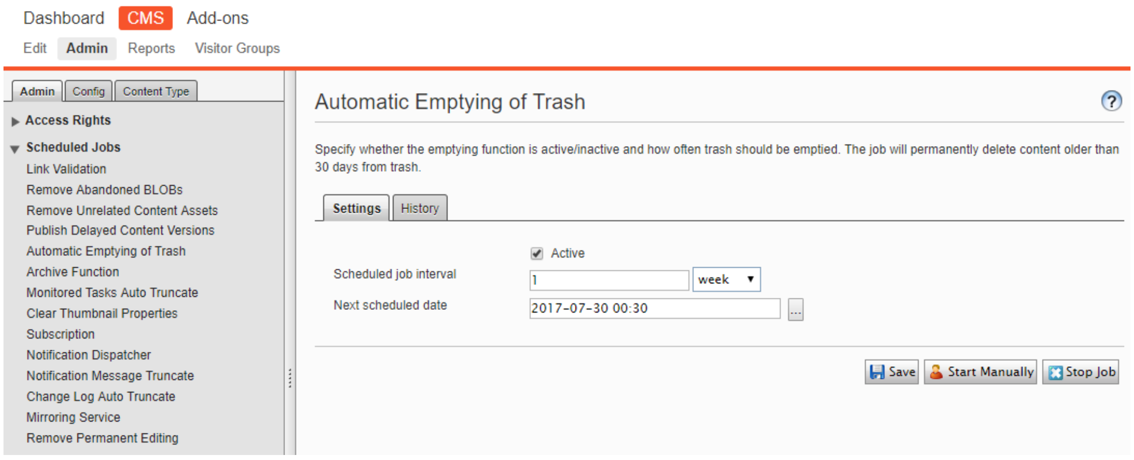
Episerver 357

From version 7.5 of Episerver the Windows Scheduler Service is no longer used in Episerver CMS sites. The Scheduled jobs have always been running inside the site, the scheduler service just pinged the site to make sure it was up and running. Since its not compatible with either Azure or xcopy deployment most sites will have website monitoring anyway so moving the responsibility for keeping the site up and running to the hosting environment seemed like a better approach.

Since scheduled jobs are executed on the site a requirement for the job to be executed is that the site is up and running. This can be done for example by using IIS feature "Application Initialization" or having a website supervisor that periodically pings the site.

Automatic Emptying of Recycle Bin

- Must be activated for emptying to be done automatically.
- State how often emptying should be done and activate.
- Never deletes pages that have been there less than 30 days.

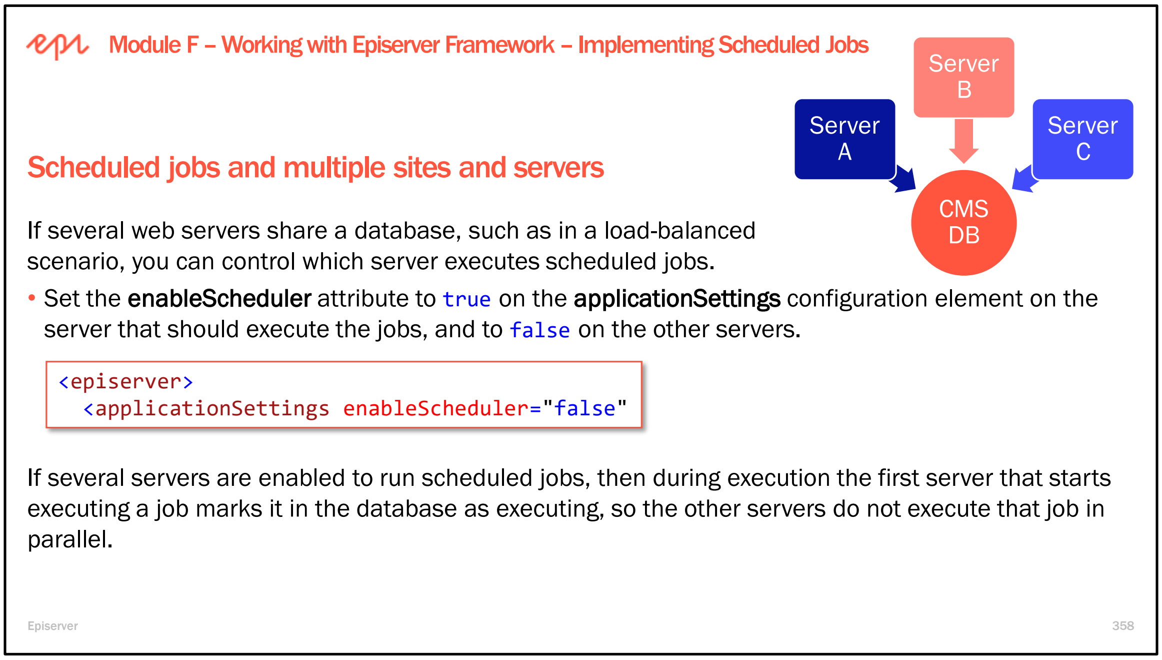


Automatic Emptying of Trash

Specify whether the emptying function is active/inactive and how often trash should be emptied. The job will permanently delete content older than 30 days from trash.

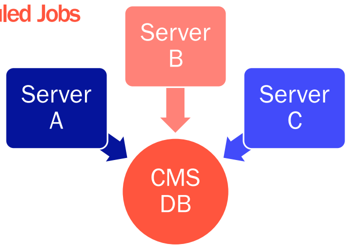
☑ Active

Scheduled job interval	1 week
Next scheduled date	2017-07-30 00:30



Module F – Working with Episerver Framework – Implementing Scheduled Jobs

Scheduled jobs and multiple sites and servers



If several web servers share a database, such as in a load-balanced scenario, you can control which server executes scheduled jobs.

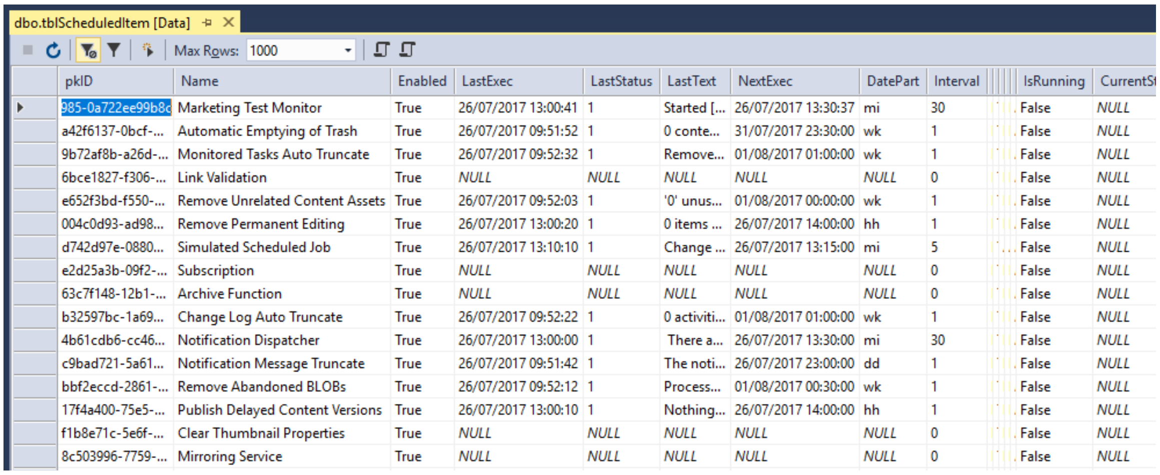
- Set the **enableScheduler** attribute to `true` on the **applicationSettings** configuration element on the server that should execute the jobs, and to `false` on the other servers.

```
<episerver>
  <applicationSettings enableScheduler="false"
```

If several servers are enabled to run scheduled jobs, then during execution the first server that starts executing a job marks it in the database as executing, so the other servers do not execute that job in parallel.

Episerver 358

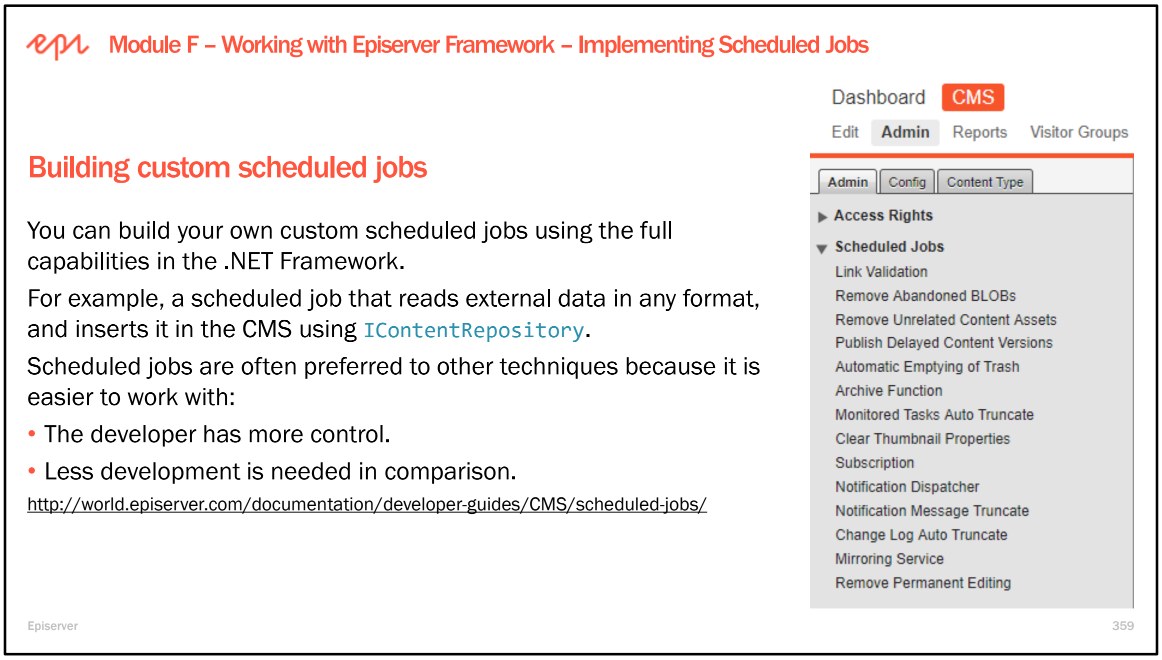
tblScheduledItem contains information about scheduled jobs and if they are running, as shown in the following screenshot:



dbo.tblScheduledItem [Data]

Max Rows: 1000

pkID	Name	Enabled	LastExec	LastStatus	LastText	NextExec	DatePart	Interval	IsRunning	CurrentSt
985-0a722ee99b8c	Marketing Test Monitor	True	26/07/2017 13:00:41	1	Started [...	26/07/2017 13:30:37	mi	30	False	*NULL*
a42f6137-0bcf-...	Automatic Emptying of Trash	True	26/07/2017 09:51:52	1	0 conte...	31/07/2017 23:30:00	wk	1	False	*NULL*
9b72af8b-a26d-...	Monitored Tasks Auto Truncate	True	26/07/2017 09:52:32	1	Remove...	01/08/2017 01:00:00	wk	1	False	*NULL*
6bce1827-f306-...	Link Validation	True	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	0	False	*NULL*
e652f3bd-f550-...	Remove Unrelated Content Assets	True	26/07/2017 09:52:03	1	'0' unus...	01/08/2017 00:00:00	wk	1	False	*NULL*
004c0d93-ad98...	Remove Permanent Editing	True	26/07/2017 13:00:20	1	0 items ...	26/07/2017 14:00:00	hh	1	False	*NULL*
d742d97e-0880...	Simulated Scheduled Job	True	26/07/2017 13:10:10	1	Change ...	26/07/2017 13:15:00	mi	5	False	*NULL*
e2d25a3b-09f2-...	Subscription	True	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	0	False	*NULL*
63c7f148-12b1-...	Archive Function	True	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	0	False	*NULL*
b32597bc-1a69...	Change Log Auto Truncate	True	26/07/2017 09:52:22	1	0 activiti...	01/08/2017 01:00:00	wk	1	False	*NULL*
4b61cdb6-cc46...	Notification Dispatcher	True	26/07/2017 13:00:00	1	There a...	26/07/2017 13:30:00	mi	30	False	*NULL*
c9bad721-5a61...	Notification Message Truncate	True	26/07/2017 09:51:42	1	The noti...	26/07/2017 23:00:00	dd	1	False	*NULL*
bbf2eccd-2861-...	Remove Abandoned BLOBs	True	26/07/2017 09:52:12	1	Process...	01/08/2017 00:30:00	wk	1	False	*NULL*
17f4a400-75e5-...	Publish Delayed Content Versions	True	26/07/2017 13:00:10	1	Nothing...	26/07/2017 14:00:00	hh	1	False	*NULL*
f1b8e71c-5e6f-...	Clear Thumbnail Properties	True	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	0	False	*NULL*
8c503996-7759-...	Mirroring Service	True	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	0	False	*NULL*



Module F – Working with Episerver Framework – Implementing Scheduled Jobs

Building custom scheduled jobs

You can build your own custom scheduled jobs using the full capabilities in the .NET Framework.

For example, a scheduled job that reads external data in any format, and inserts it in the CMS using `IContentRepository`.

Scheduled jobs are often preferred to other techniques because it is easier to work with:

- The developer has more control.
- Less development is needed in comparison.

http://world.episerver.com/documentation/developer-guides/CMS/scheduled-jobs/

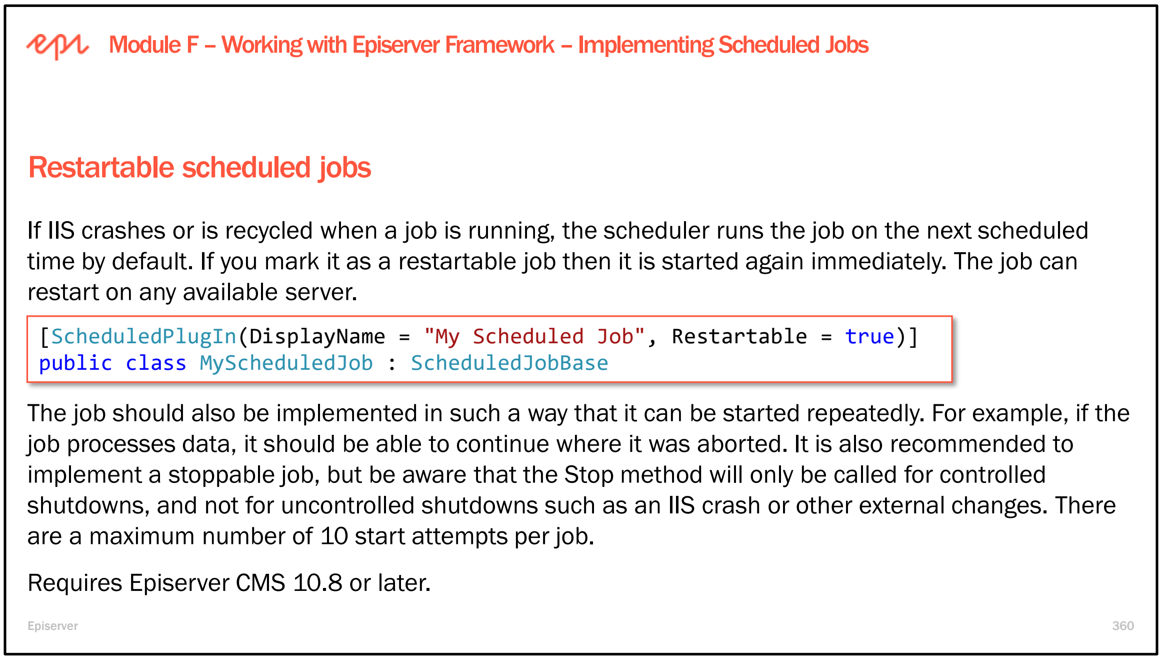
Dashboard CMS
Edit Admin Reports Visitor Groups
Admin Config Content Type

- Access Rights
- Scheduled Jobs
  - Link Validation
  - Remove Abandoned BLOBs
  - Remove Unrelated Content Assets
  - Publish Delayed Content Versions
  - Automatic Emptying of Trash
  - Archive Function
  - Monitored Tasks Auto Truncate
  - Clear Thumbnail Properties
  - Subscription
  - Notification Dispatcher
  - Notification Message Truncate
  - Change Log Auto Truncate
  - Mirroring Service
  - Remove Permanent Editing

Episerver 359

Add a new project item of type **Scheduled Job** and implement its **Execute** method as shown below. Unhandled exceptions are automatically caught and returned to the user interface as a “failed” job.

```
public override string Execute()
{
    // if this job is run manually then this will NOT be null and the current user
    // permissions will be checked, else, we might need to assign higher permissions.
    if (HttpContext.Current == null)
    {
        PrincipalInfo.CurrentPrincipal = new GenericPrincipal(
            new GenericIdentity("Scheduled Job Demo"),
            new[] { "Administrators" });
    }

    OnStatusChanged(string.Format("Starting execution of {0}", GetType()));

    var r = new Random();
    int percentComplete = 0;

    while (percentComplete < 100)
    {
        System.Threading.Thread.Sleep(2000);
        percentComplete += r.Next(5, 15);
        OnStatusChanged(string.Format(
            "{0}% complete. Please wait...", percentComplete));
        if (_stopSignaled)
        {
            return "Stop of job was called";
        }
    }
    return "Completed successfully!";
}
```



Module F – Working with Episerver Framework – Implementing Scheduled Jobs

Restartable scheduled jobs

If IIS crashes or is recycled when a job is running, the scheduler runs the job on the next scheduled time by default. If you mark it as a restartable job then it is started again immediately. The job can restart on any available server.

```
[ScheduledPlugIn(DisplayName = "My Scheduled Job", Restartable = true)]
public class MyScheduledJob : ScheduledJobBase
```

The job should also be implemented in such a way that it can be started repeatedly. For example, if the job processes data, it should be able to continue where it was aborted. It is also recommended to implement a stoppable job, but be aware that the Stop method will only be called for controlled shutdowns, and not for uncontrolled shutdowns such as an IIS crash or other external changes. There are a maximum number of 10 start attempts per job.

Requires Episerver CMS 10.8 or later.

Episerver 360

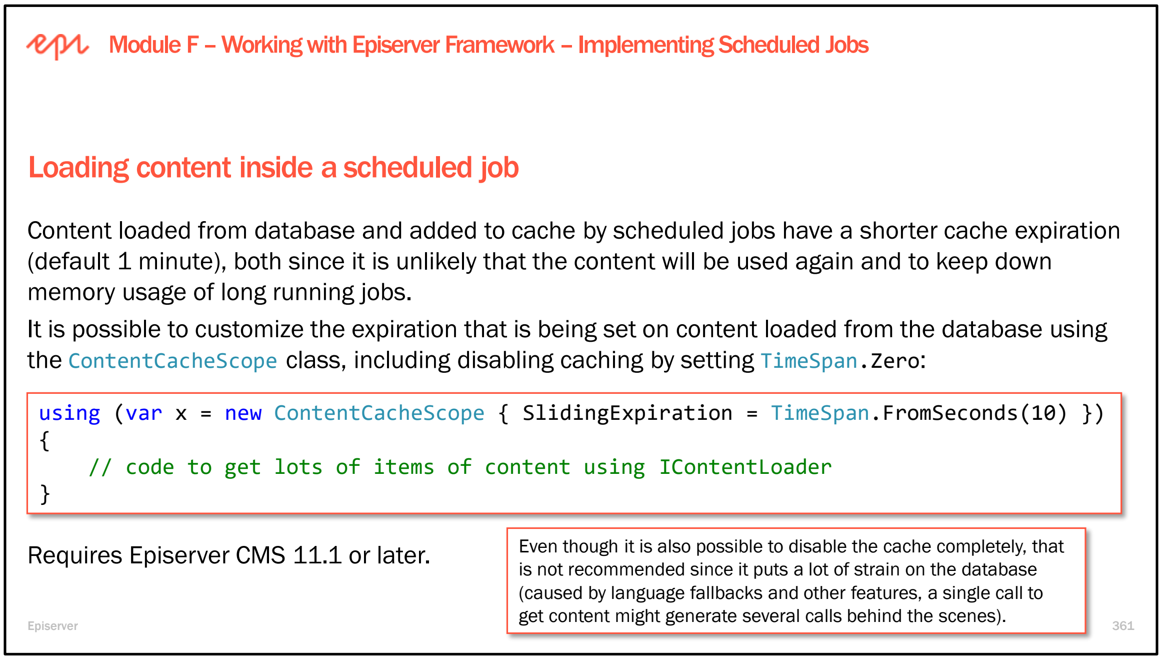
Scheduled jobs improvements with Episerver CMS 10.3 or later

You don't need to inherit from base class `EPiServer.Scheduler.ScheduledJobBase`. All you need is to have static string `Execute()` method. This allows you to implement jobs without any dependencies on the EPiServer assemblies.

You can use the `IScheduledJobFactory` interface and implementation to use proper dependency injection technique.

You can execute scheduled jobs using the `IScheduledJobExecutor` interface and implementation.

Read more on Waldis Iljuczonok's blog:
https://blog.tech-fellow.net/2016/12/28/scheduled-jobs-updates/



Module F – Working with Episerver Framework – Implementing Scheduled Jobs

Loading content inside a scheduled job

Content loaded from database and added to cache by scheduled jobs have a shorter cache expiration (default 1 minute), both since it is unlikely that the content will be used again and to keep down memory usage of long running jobs.

It is possible to customize the expiration that is being set on content loaded from the database using the `ContentCacheScope` class, including disabling caching by setting `TimeSpan.Zero`:

```
using (var x = new ContentCacheScope { SlidingExpiration = TimeSpan.FromSeconds(10) })
{
    // code to get lots of items of content using IContentLoader
}
```

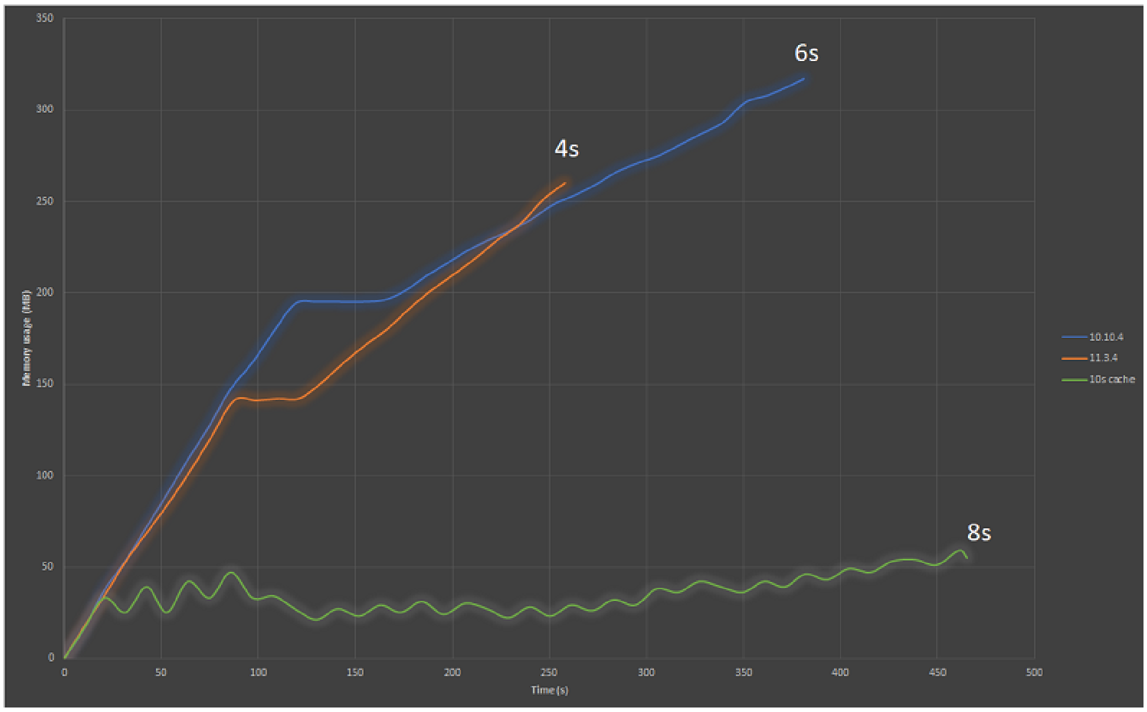
Requires Episerver CMS 11.1 or later.

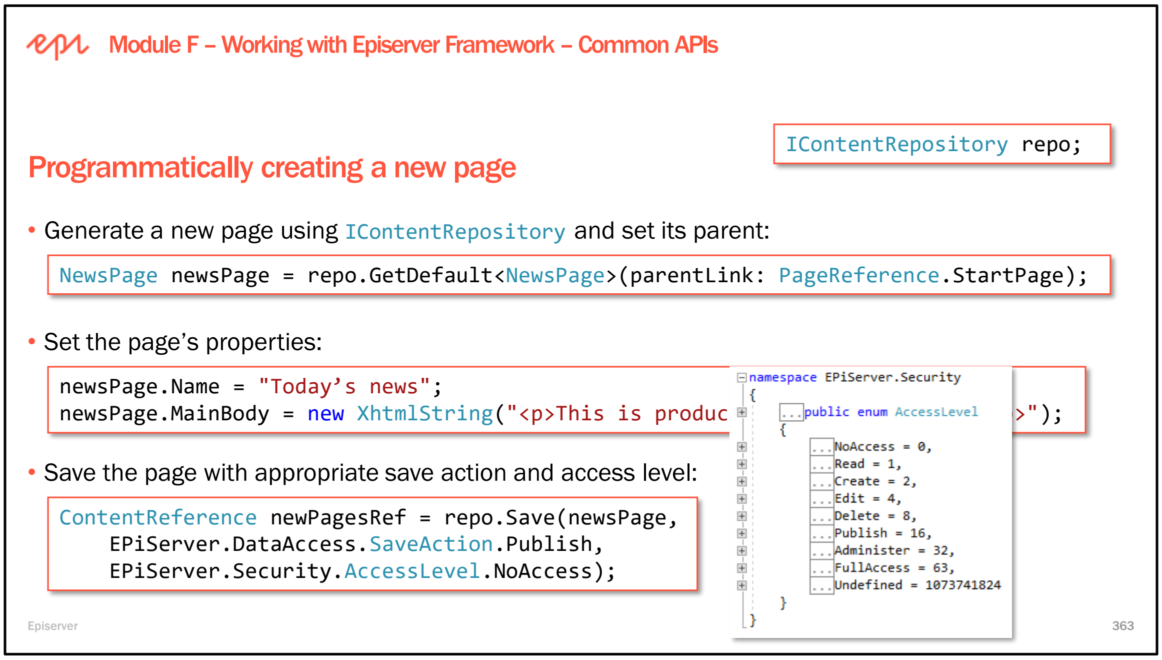
Even though it is also possible to disable the cache completely, that is not recommended since it puts a lot of strain on the database (caused by language fallbacks and other features, a single call to get content might generate several calls behind the scenes).

Episerver

361

Performance improvements in CMS 11
https://world.episerver.com/blogs/Per-Bjurstrom/Archive/2018/3/performance-improvements-in-cms-11/





Module F – Working with Episerver Framework – Common APIs

Programmatically creating a new page

```
IContentRepository repo;
```

- Generate a new page using `IContentRepository` and set its parent:

```
NewsPage newsPage = repo.GetDefault<NewsPage>(parentLink: PageReference.StartPage);
```

- Set the page's properties:

```
newsPage.Name = "Today's news";
newsPage.MainBody = new XhtmlString("<p>This is produc                 >");
```

- Save the page with appropriate save action and access level:

```
ContentReference newPagesRef = repo.Save(newsPage,
    EPiServer.DataAccess.SaveAction.Publish,
    EPiServer.Security.AccessLevel.NoAccess);
```

```
namespace EPiServer.Security
{
    public enum AccessLevel
    {
        NoAccess = 0,
        Read = 1,
        Create = 2,
        Edit = 4,
        Delete = 8,
        Publish = 16,
        Administer = 32,
        FullAccess = 63,
        Undefined = 1073741824
    }
}
```

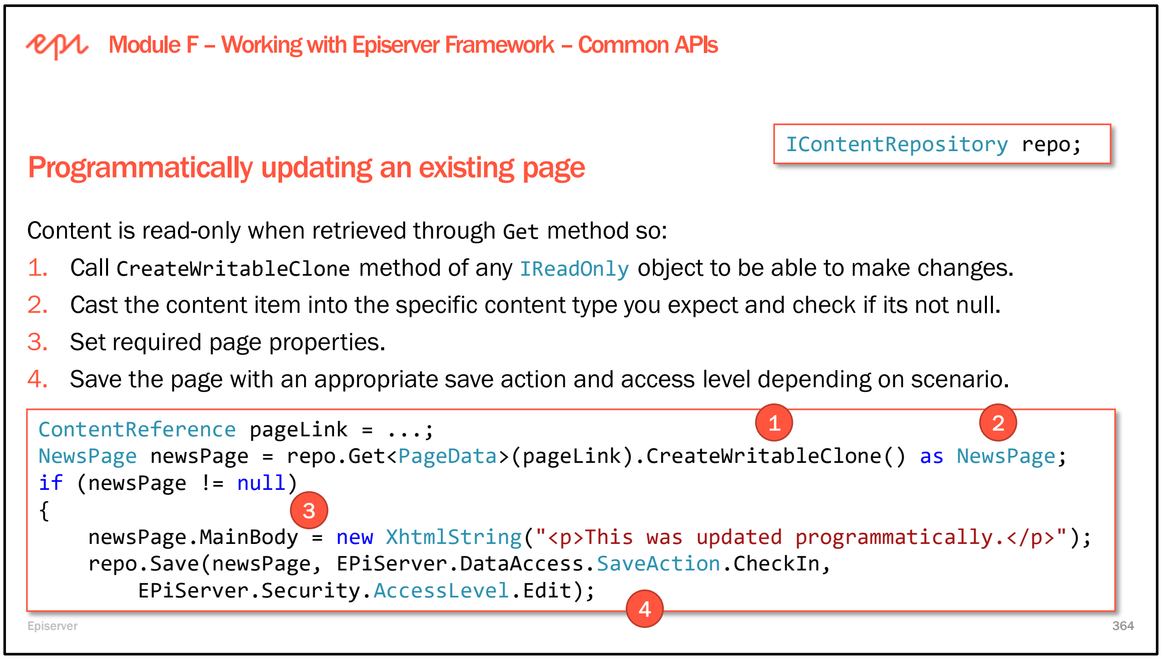
Episerver 363

AccessLevel enum
When calling IContentRepository.Save method, you can pass an AccessLevel. This is the ***minimal*** level that the current user (i.e. anonymous or logged in visitor or editor) must have in order for the method to succeed. If the current user does not have that minimal level then the Save method throws an exception. Therefore, if you pass the **NoAccess** value, you are allowing ***every*** user to successfully call the method.

SaveAction enum
One of the improvements in CMS 10 is to the IContentRepository Save API and its SaveAction enum values:
- CheckIn - Checks in a version indicating that it is ready to be published
- CheckOut - Checks out a version to indicate that it is being worked on. (New in CMS 10)
- RequestApproval – Indicate that the version is ready for an approval review.
- Reject - Rejects a version. This is normally done after a review has been done.
- Publish - Publishes a version. The currently published version will automatically transition to a previously published state.
- Schedule – Used to schedule a version for automatic publishing at a later date. (New in CMS 10)
- Save - deprecated (it won't appear in IntelliSense but it would compile)

Improving the Save experience in CMS 10
http://world.episerver.com/blogs/Henrik-Nystrom/Dates/2016/10/improving-the-saving-experience/

Content versions and states
A content item that supports different statuses implements IVersionable. The interface contains a property Status that specifies the current status of the content version.
A content version can have one of the following different statuses:
- NotCreated
- CheckedIn, CheckedOut
- AwaitingApproval, Rejected
- DelayedPublished, Published, PreviouslyPublished



Module F – Working with Episerver Framework – Common APIs

```
IContentRepository repo;
```

Programmatically updating an existing page

Content is read-only when retrieved through `Get` method so:

1. Call `CreateWritableClone` method of any `IReadOnly` object to be able to make changes.
2. Cast the content item into the specific content type you expect and check if its not null.
3. Set required page properties.
4. Save the page with an appropriate save action and access level depending on scenario.

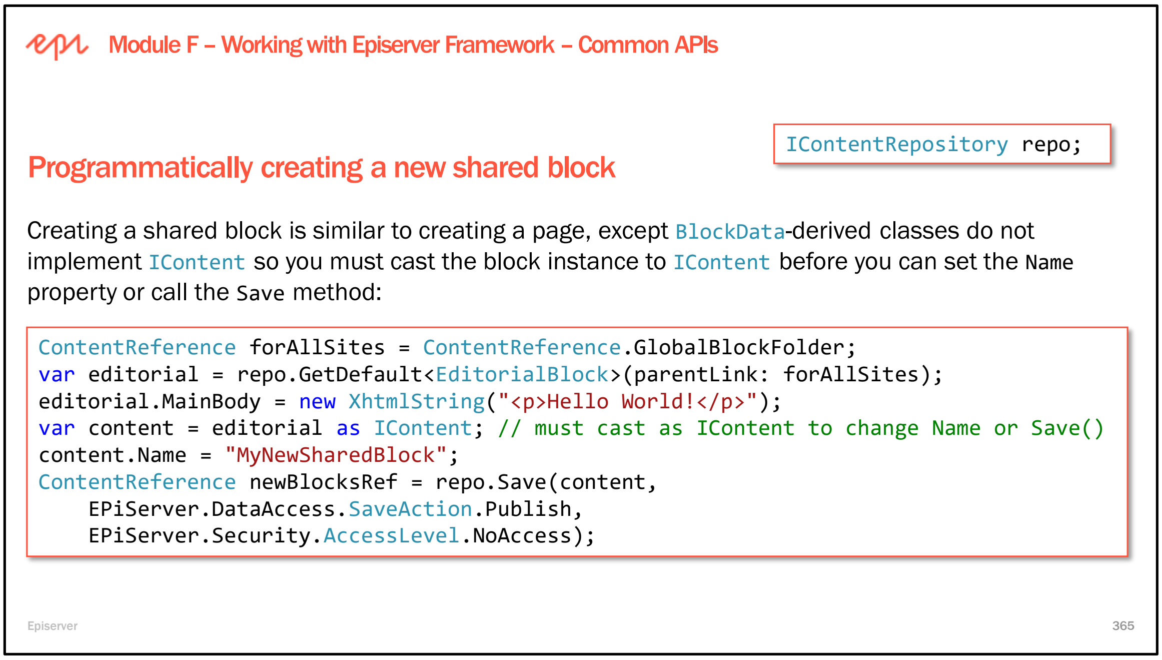
```
ContentReference pageLink = ...;
NewsPage newsPage = repo.Get<PageData>(pageLink).CreateWritableClone() as NewsPage;
if (newsPage != null)
{
    newsPage.MainBody = new XhtmlString("<p>This was updated programmatically.</p>");
    repo.Save(newsPage, EPiServer.DataAccess.SaveAction.CheckIn,
        EPiServer.Security.AccessLevel.Edit);
```

Episerver 364

Align status transitions, events and required access rights when saving
http://world.episerver.com/documentation/Release-Notes/ReleaseNote/?releaseNoteId=CMS-2078

CMS 10 and later Save API is based on the following principles:

- SaveAction.Default replaces SaveAction.None.
- SaveAction.CheckOut should be used to check out the current content version.
- Saving an item, regardless of SaveAction should update the current version, except when status is Published or PreviouslyPublished, in which case a new version is created.
- No status changes are allowed on the current version if status is Published or PreviouslyPublished.
- When ForceCurrentVersion is used on a Published or PreviouslyPublished version, it must be on its own (Default) or paired with the Publish action in case of Published content.
- Saving a content item without a version should be identical to saving the version that is actually loaded.
- SaveAction.Publish (or Default) should be the only action allowed on Unversioned content.
- Invalid SaveAction combinations should throw an InvalidOperationException, for example, using both ForceCurrentVersion or ForceNewVersion should be invalid.
- Create access is required if no previous version exists and when saving a new language branch.
- Edit access is required to CheckIn, CheckOut, or RequestReview a version.
- Publish access is required to Publish or Schedule a version or update a Published or DelayPublished version.
- IContentRepository Save extensions methods without a required access parameter pass AccessLevel.Undefined to the Save method.



Module F – Working with Episerver Framework – Common APIs

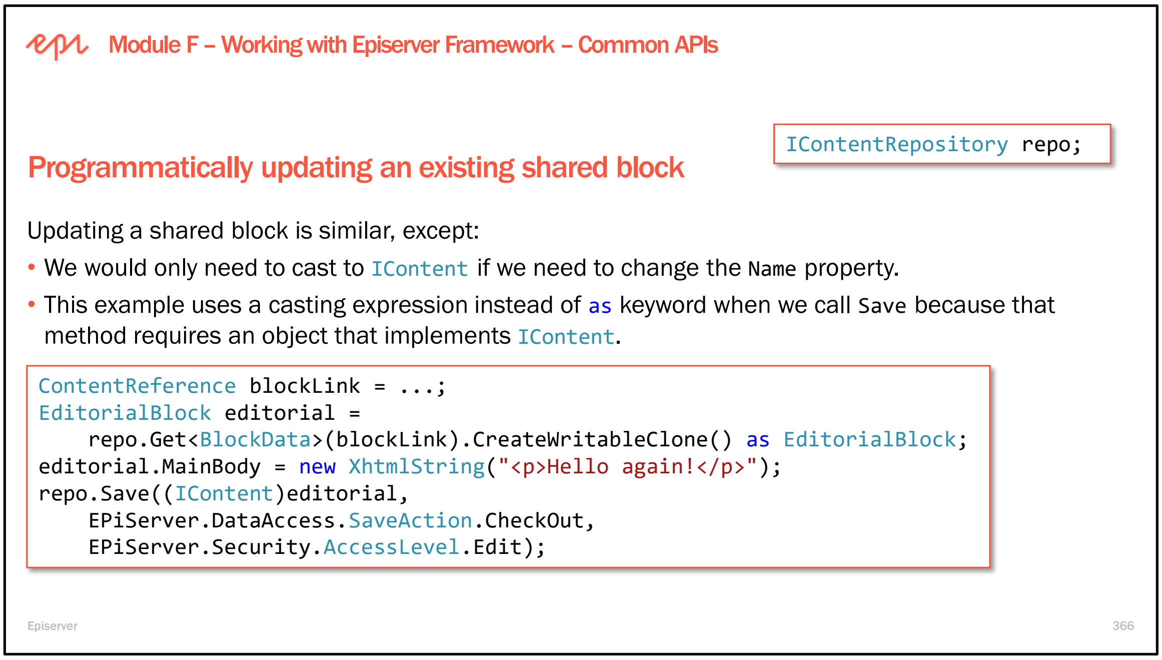
```
IContentRepository repo;
```

Programmatically creating a new shared block

Creating a shared block is similar to creating a page, except `BlockData`-derived classes do not implement `IContent` so you must cast the block instance to `IContent` before you can set the `Name` property or call the `Save` method:

```
ContentReference forAllSites = ContentReference.GlobalBlockFolder;
var editorial = repo.GetDefault<EditorialBlock>(parentLink: forAllSites);
editorial.MainBody = new XhtmlString("<p>Hello World!</p>");
var content = editorial as IContent; // must cast as IContent to change Name or Save()
content.Name = "MyNewSharedBlock";
ContentReference newBlocksRef = repo.Save(content,
    EPiServer.DataAccess.SaveAction.Publish,
    EPiServer.Security.AccessLevel.NoAccess);
```

Episerver 365



Module F – Working with Episerver Framework – Common APIs

```
IContentRepository repo;
```

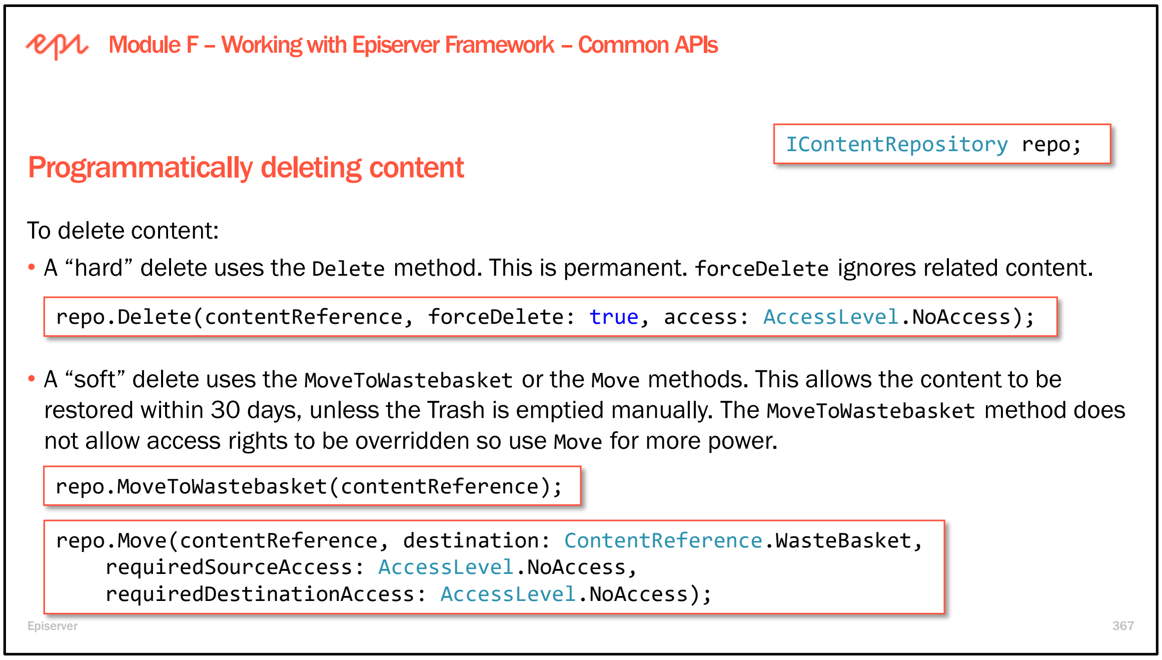
Programmatically updating an existing shared block

Updating a shared block is similar, except:

- We would only need to cast to `IContent` if we need to change the `Name` property.
- This example uses a casting expression instead of `as` keyword when we call `Save` because that method requires an object that implements `IContent`.

```
ContentReference blockLink = ...;
EditorialBlock editorial =
    repo.Get<BlockData>(blockLink).CreateWritableClone() as EditorialBlock;
editorial.MainBody = new XhtmlString("<p>Hello again!</p>");
repo.Save((IContent)editorial,
    EPiServer.DataAccess.SaveAction.CheckOut,
    EPiServer.Security.AccessLevel.Edit);
```

Episerver 366



Module F – Working with Episerver Framework – Common APIs

```
IContentRepository repo;
```

Programmatically deleting content

To delete content:

- A “hard” delete uses the `Delete` method. This is permanent. `forceDelete` ignores related content.

```
repo.Delete(contentReference, forceDelete: true, access: AccessLevel.NoAccess);
```

- A “soft” delete uses the `MoveToWastebasket` or the `Move` methods. This allows the content to be restored within 30 days, unless the Trash is emptied manually. The `MoveToWastebasket` method does not allow access rights to be overridden so use `Move` for more power.

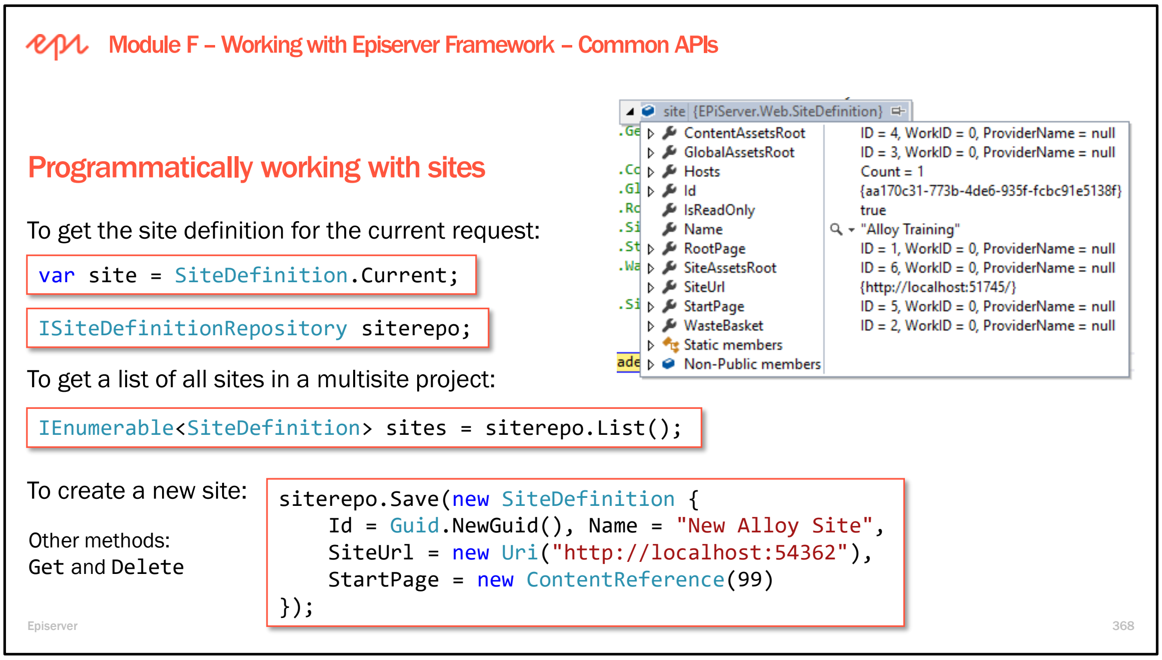
```
repo.MoveToWastebasket(contentReference);
```

```
repo.Move(contentReference, destination: ContentReference.WasteBasket,
    requiredSourceAccess: AccessLevel.NoAccess,
    requiredDestinationAccess: AccessLevel.NoAccess);
```

Episerver 367

Other methods to permanently delete content include:

```
void Delete(
    ContentReference contentLink,
    bool forceDelete,
    EPiServer.Security.AccessLevel access)

void DeleteChildren(
    ContentReference contentLink,
    bool forceDelete,
    EPiServer.Security.AccessLevel access)

void DeleteLanguageBranch(
    ContentReference contentLink,
    string languageBranch,
    EPiServer.Security.AccessLevel access)
```



Module F – Working with Episerver Framework – Common APIs

Programmatically working with sites

To get the site definition for the current request:

```
var site = SiteDefinition.Current;
```

```
ISiteDefinitionRepository siterepo;
```

To get a list of all sites in a multisite project:

```
IEnumerable<SiteDefinition> sites = siterepo.List();
```

```
site {EPiServer.Web.SiteDefinition}
  ContentAssetsRoot    ID = 4, WorkID = 0, ProviderName = null
  GlobalAssetsRoot     ID = 3, WorkID = 0, ProviderName = null
  Hosts                Count = 1
  Id                   {aa170c31-773b-4de6-935f-fcbc91e5138f}
  IsReadOnly           true
  Name                 "Alloy Training"
  RootPage             ID = 1, WorkID = 0, ProviderName = null
  SiteAssetsRoot       ID = 6, WorkID = 0, ProviderName = null
  SiteUrl              {http://localhost:51745/}
  StartPage            ID = 5, WorkID = 0, ProviderName = null
  WasteBasket          ID = 2, WorkID = 0, ProviderName = null
  Static members
  Non-Public members
```

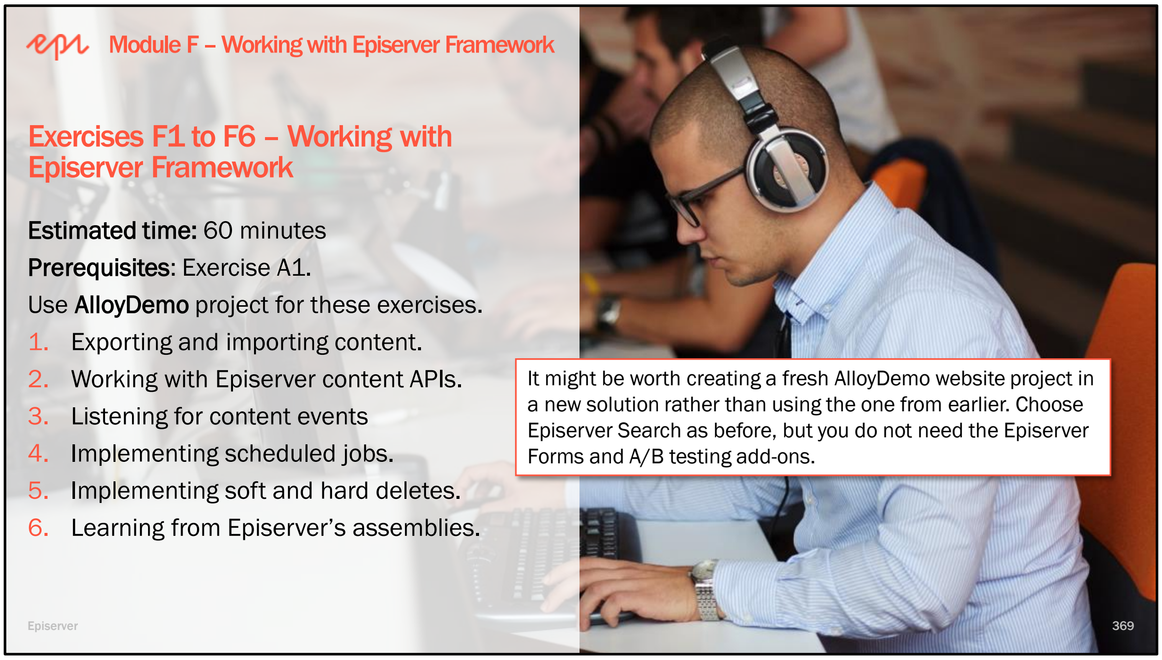
To create a new site:

```
siterepo.Save(new SiteDefinition {
    Id = Guid.NewGuid(), Name = "New Alloy Site",
    SiteUrl = new Uri("http://localhost:54362"),
    StartPage = new ContentReference(99)
});
```

Other methods:
Get and Delete

Episerver 368

Episerver CMS – Advanced Development training course covers more of the Episerver Framework and Episerver CMS APIs.



Module F – Working with Episerver Framework

Exercises F1 to F6 – Working with Episerver Framework

Estimated time: 60 minutes

Prerequisites: Exercise A1.

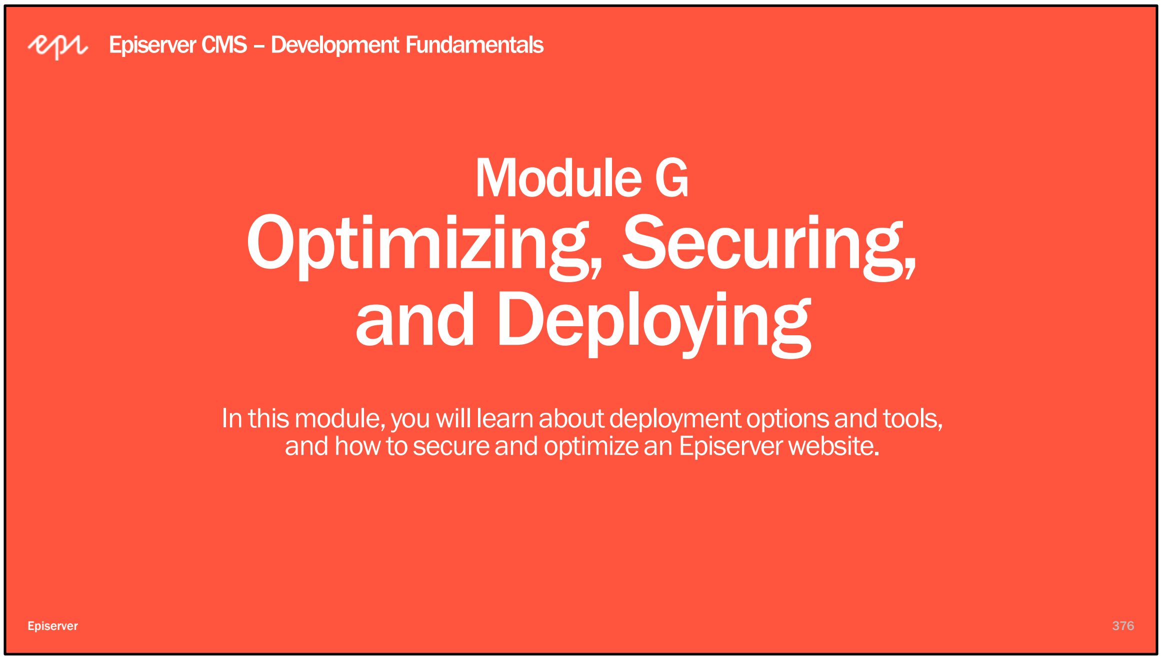
Use **AlloyDemo** project for these exercises.

1. Exporting and importing content.
2. Working with Episerver content APIs.
3. Listening for content events
4. Implementing scheduled jobs.
5. Implementing soft and hard deletes.
6. Learning from Episerver's assemblies.

It might be worth creating a fresh AlloyDemo website project in a new solution rather than using the one from earlier. Choose Episerver Search as before, but you do not need the Episerver Forms and A/B testing add-ons.

Episerver

369



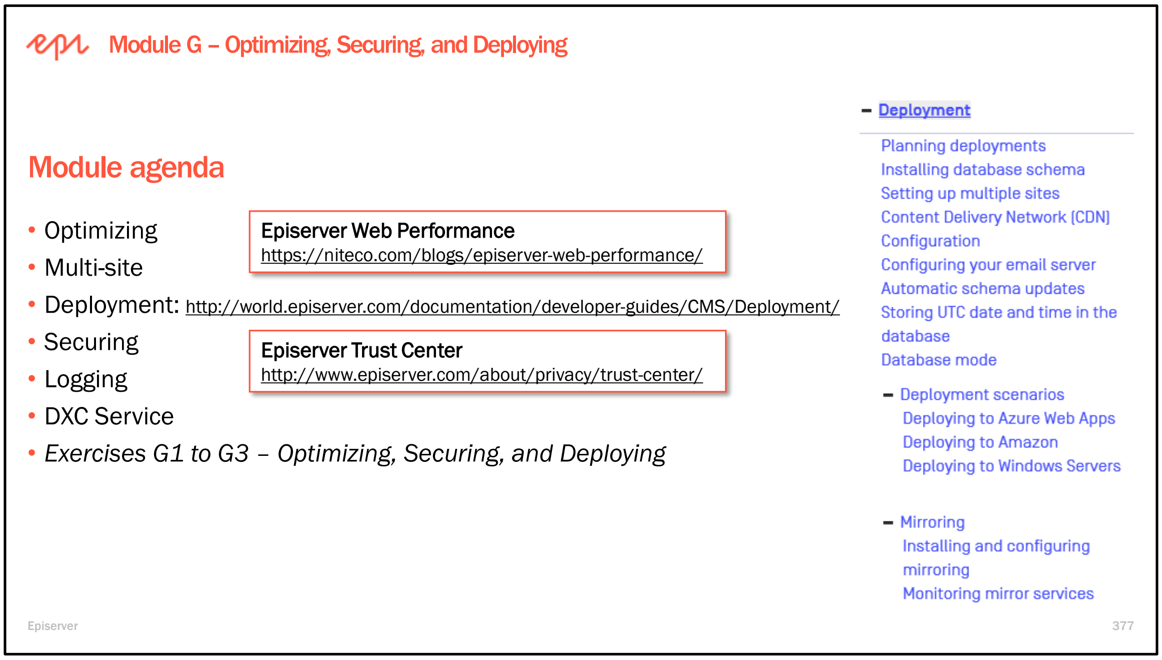
Episerver CMS – Development Fundamentals

Module G
Optimizing, Securing, and Deploying

In this module, you will learn about deployment options and tools, and how to secure and optimize an Episerver website.

Episerver

376



Module G – Optimizing, Securing, and Deploying

Module agenda

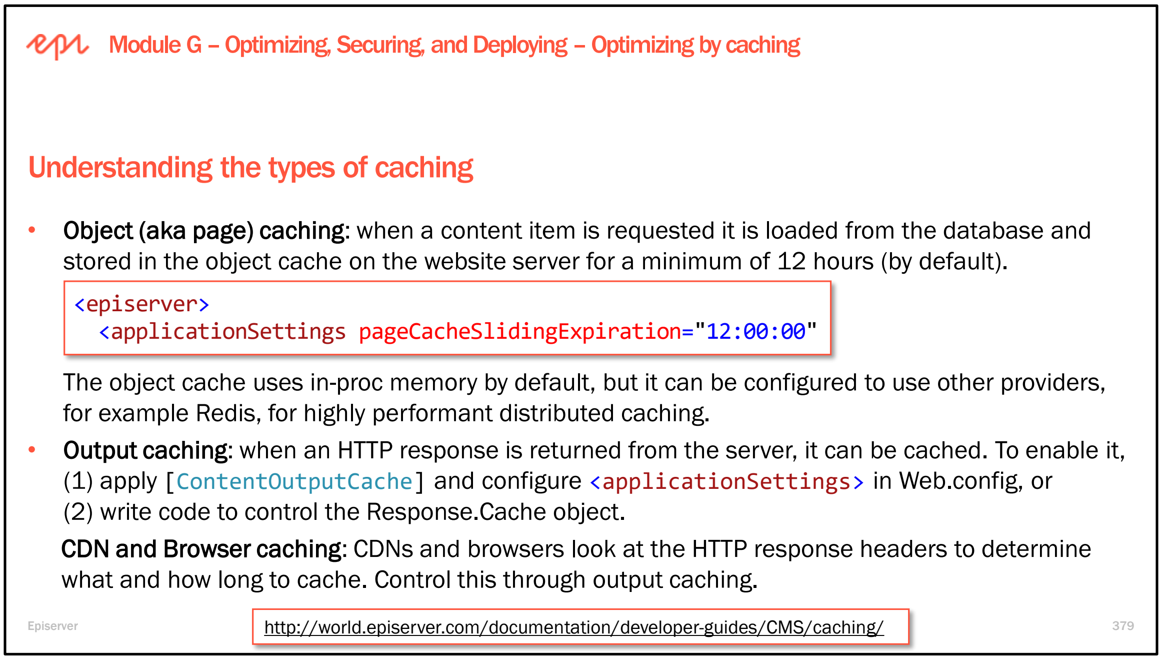
- Optimizing
- Multi-site
- Deployment: http://world.episerver.com/documentation/developer-guides/CMS/Deployment/
- Securing
- Logging
- DXC Service
- *Exercises G1 to G3 – Optimizing, Securing, and Deploying*

Episerver Web Performance
https://niteco.com/blogs/episerver-web-performance/

Episerver Trust Center
http://www.episerver.com/about/privacy/trust-center/

- Deployment
  - Planning deployments
  - Installing database schema
  - Setting up multiple sites
  - Content Delivery Network (CDN)
  - Configuration
  - Configuring your email server
  - Automatic schema updates
  - Storing UTC date and time in the database
  - Database mode
  - Deployment scenarios
    - Deploying to Azure Web Apps
    - Deploying to Amazon
    - Deploying to Windows Servers
  - Mirroring
    - Installing and configuring mirroring
    - Monitoring mirror services

Episerver 377



Module G – Optimizing, Securing, and Deploying – Optimizing by caching

Understanding the types of caching

- **Object (aka page) caching**: when a content item is requested it is loaded from the database and stored in the object cache on the website server for a minimum of 12 hours (by default).

```
<episerver>
  <applicationSettings pageCacheSlidingExpiration="12:00:00"
```

  The object cache uses in-proc memory by default, but it can be configured to use other providers, for example Redis, for highly performant distributed caching.

- **Output caching**: when an HTTP response is returned from the server, it can be cached. To enable it, (1) apply [`ContentOutputCache`] and configure `<applicationSettings>` in Web.config, or (2) write code to control the Response.Cache object.

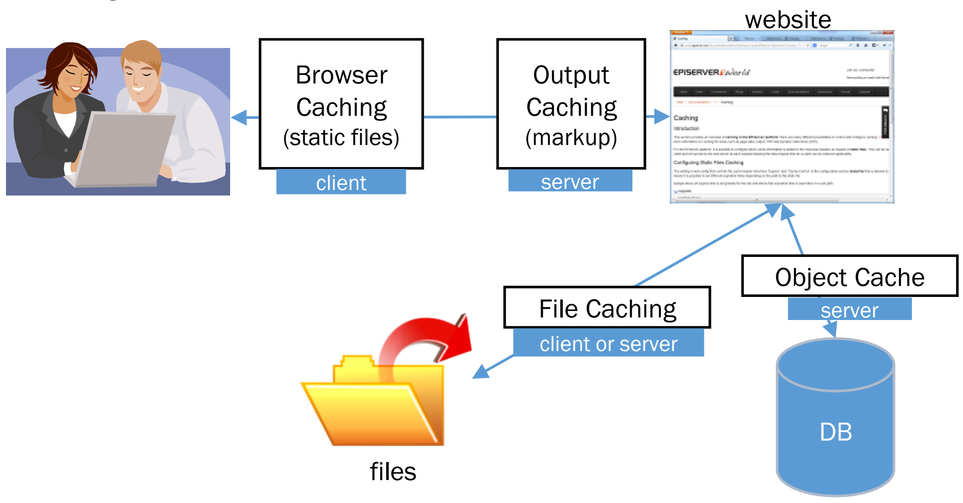
  CDN and Browser caching: CDNs and browsers look at the HTTP response headers to determine what and how long to cache. Control this through output caching.

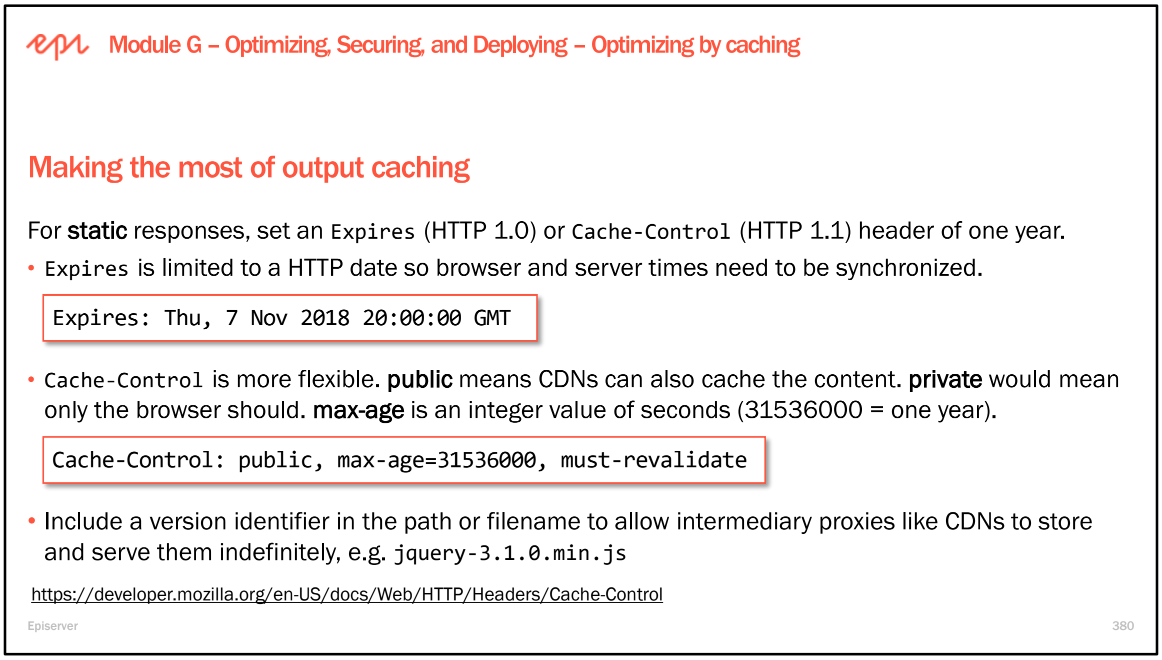
Episerver http://world.episerver.com/documentation/developer-guides/CMS/caching/ 379

Object Cache

- Based on the ASP.NET Cache.
- Automatically caches all objects in Episerver CMS that is being requested from the API, via for example IContentRepository.
- Only read-only objects are stored to enable great performance.
- Has a few advanced characteristics to improve scalability. For example, it uses an optimistic locking approach when multiple threads are reading the same data they will all piggyback on the same database calls to avoid putting to much load on the database for "hot" objects that have not yet been cached.

Types of caching





Module G – Optimizing, Securing, and Deploying – Optimizing by caching

Making the most of output caching

For **static** responses, set an `Expires` (HTTP 1.0) or `Cache-Control` (HTTP 1.1) header of one year.

- `Expires` is limited to a HTTP date so browser and server times need to be synchronized.

```
Expires: Thu, 7 Nov 2018 20:00:00 GMT
```

- `Cache-Control` is more flexible. **public** means CDNs can also cache the content. **private** would mean only the browser should. **max-age** is an integer value of seconds (31536000 = one year).

```
Cache-Control: public, max-age=31536000, must-revalidate
```

- Include a version identifier in the path or filename to allow intermediary proxies like CDNs to store and serve them indefinitely, e.g. `jquery-3.1.0.min.js`

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control

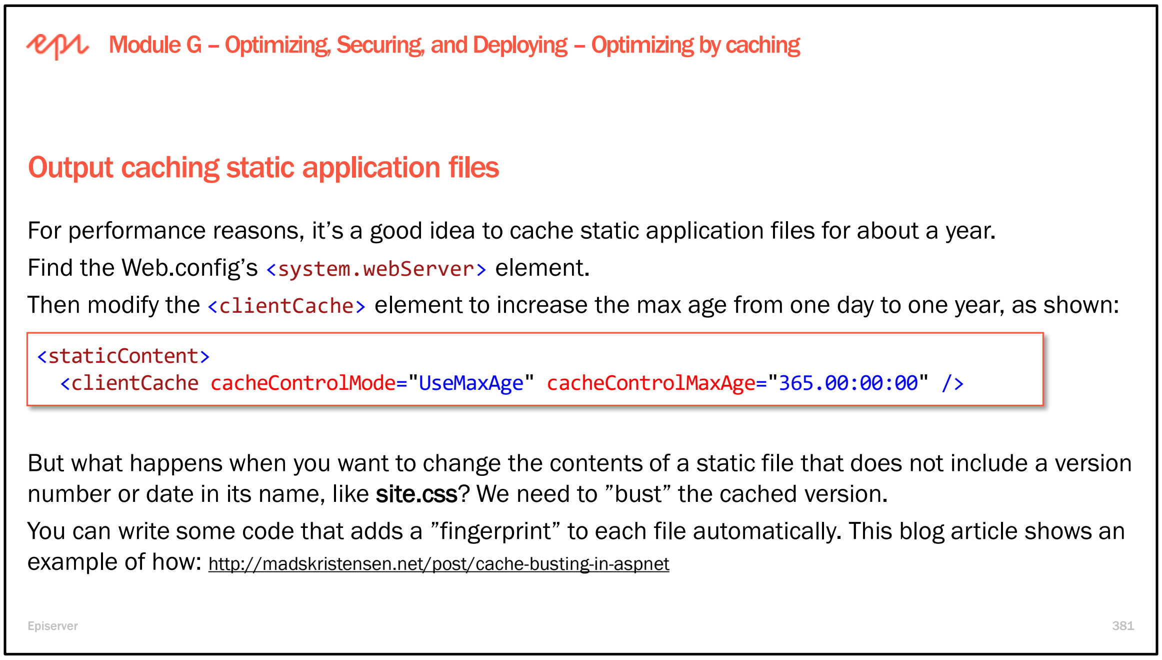
Episerver 380

Output Caching

- Based on ASP.NET Output Caching.
- This is an effective method since the entire rendered markup of a full or partial view will be cached for a specified duration.
- The cache is automatically invalidated when content in Episerver CMS is published.
- You can define dependency rules for the cache, as well as which parts of the website should be affected.

Russian Doll caching

The idea with Russian doll caching is to cache items in several layers where each layer has a dependency on next layer.
https://world.episerver.com/blogs/Johan-Bjornfot/Dates1/2017/12/html-caching-in-redis/



Module G – Optimizing, Securing, and Deploying – Optimizing by caching

Output caching static application files

For performance reasons, it's a good idea to cache static application files for about a year.

Find the Web.config's `<system.webServer>` element.

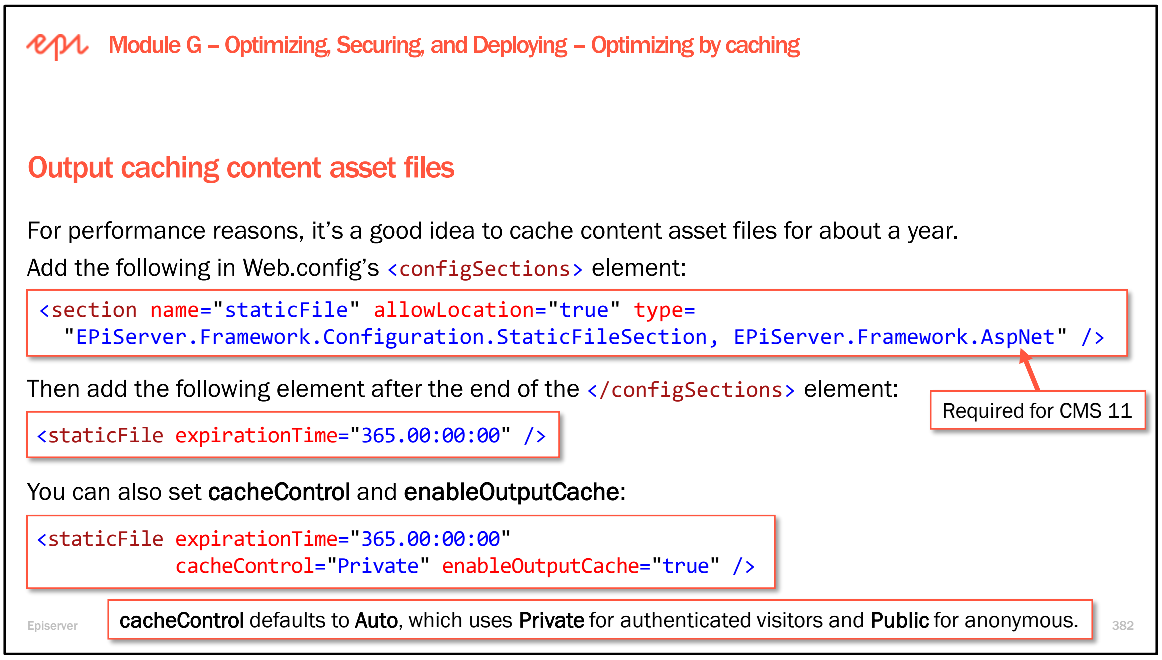
Then modify the `<clientCache>` element to increase the max age from one day to one year, as shown:

```
<staticContent>
  <clientCache cacheControlMode="UseMaxAge" cacheControlMaxAge="365.00:00:00" />
```

But what happens when you want to change the contents of a static file that does not include a version number or date in its name, like **site.css**? We need to "bust" the cached version.

You can write some code that adds a "fingerprint" to each file automatically. This blog article shows an example of how: http://madskristensen.net/post/cache-busting-in-aspnet

Episerver 381



Output caching content asset files

For performance reasons, it's a good idea to cache content asset files for about a year.

Add the following in Web.config's `<configSections>` element:

```
<section name="staticFile" allowLocation="true" type=
  "EPiServer.Framework.Configuration.StaticFileSection, EPiServer.Framework.AspNet" />
```

Required for CMS 11

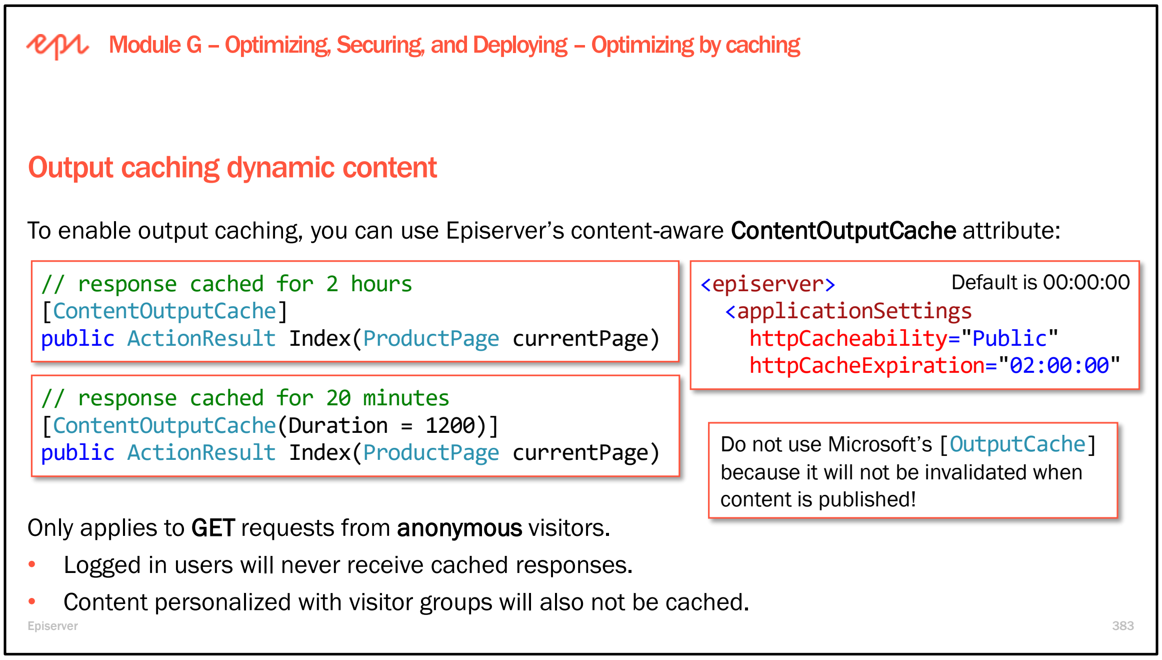
Then add the following element after the end of the `</configSections>` element:

```
<staticFile expirationTime="365.00:00:00" />
```

You can also set **cacheControl** and **enableOutputCache**:

```
<staticFile expirationTime="365.00:00:00"
            cacheControl="Private" enableOutputCache="true" />
```

cacheControl defaults to **Auto**, which uses **Private** for authenticated visitors and **Public** for anonymous.



Module G – Optimizing, Securing, and Deploying – Optimizing by caching

Output caching dynamic content

To enable output caching, you can use Episerver's content-aware **ContentOutputCache** attribute:

```
// response cached for 2 hours
[ContentOutputCache]
public ActionResult Index(ProductPage currentPage)
```

```
// response cached for 20 minutes
[ContentOutputCache(Duration = 1200)]
public ActionResult Index(ProductPage currentPage)
```

```
<episerver>
  <applicationSettings
    httpCacheability="Public"
    httpCacheExpiration="02:00:00"
```

Default is 00:00:00

Do not use Microsoft's [OutputCache] because it will not be invalidated when content is published!

Only applies to **GET** requests from **anonymous** visitors.

- Logged in users will never receive cached responses.
- Content personalized with visitor groups will also not be cached.

Episerver 383

Using Response.Cache to control output caching of dynamic content

Set cacheability and `max-age` in the HTTP response headers just before returning an action result (`TimeSpan` will automatically convert into seconds):

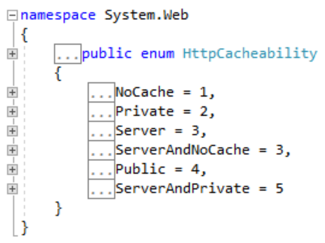
```
Response.Cache.SetCacheability(HttpCacheability.Public);
Response.Cache.SetMaxAge(TimeSpan.FromDays(1));
return View();
```

To set a sliding expiration so each request renews the cache:

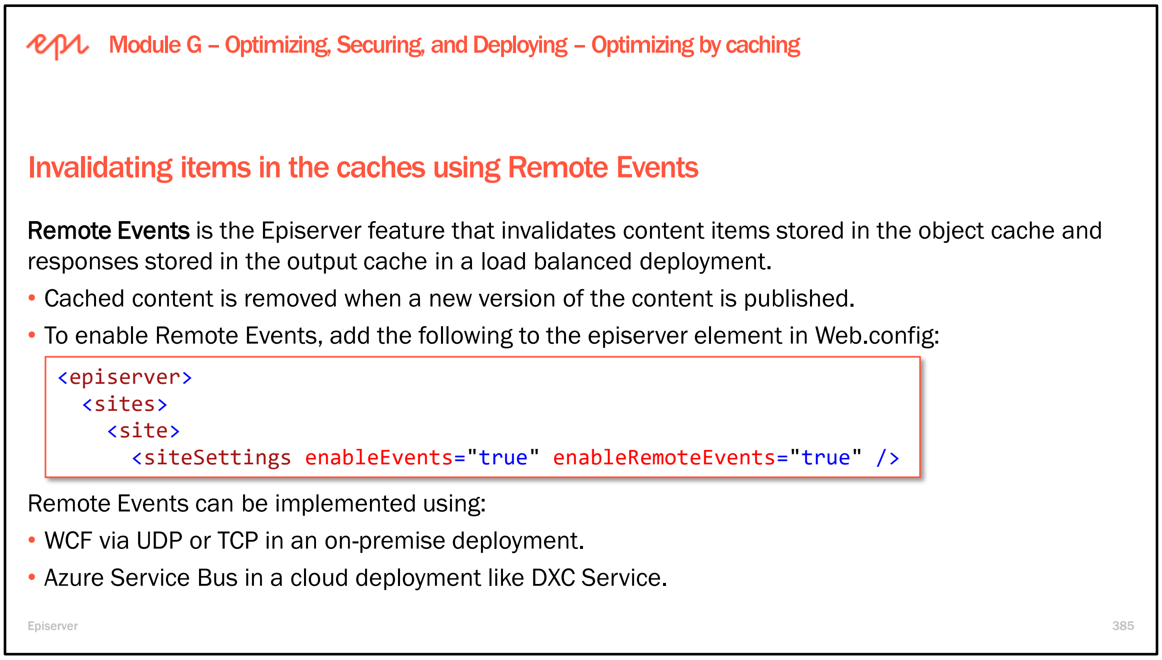
```
Response.Cache.SetSlidingExpiration(true);
```

To set `Expires` header (the older HTTP 1.0 standard):

```
Response.Cache.SetExpires(DateTime.Now.AddDays(1));
```



```
namespace System.Web
{
    ...public enum HttpCacheability
    {
        ...NoCache = 1,
        ...Private = 2,
        ...Server = 3,
        ...ServerAndNoCache = 3,
        ...Public = 4,
        ...ServerAndPrivate = 5
    }
}
```



Module G – Optimizing, Securing, and Deploying – Optimizing by caching

Invalidating items in the caches using Remote Events

Remote Events is the Episerver feature that invalidates content items stored in the object cache and responses stored in the output cache in a load balanced deployment.

- Cached content is removed when a new version of the content is published.
- To enable Remote Events, add the following to the episerver element in Web.config:

```
<episerver>
  <sites>
    <site>
      <siteSettings enableEvents="true" enableRemoteEvents="true" />
```

Remote Events can be implemented using:

- WCF via UDP or TCP in an on-premise deployment.
- Azure Service Bus in a cloud deployment like DXC Service.

Episerver 385

Configuring Remote Events

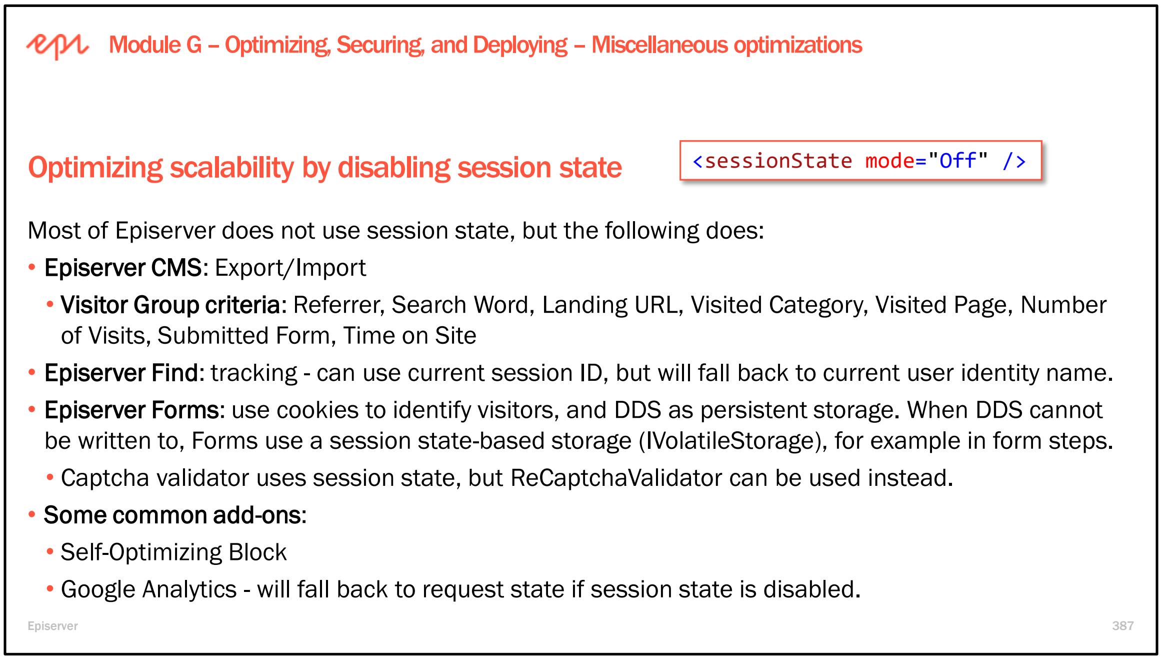
```
<bindings>
  <netTcpBinding>
    <binding name="RemoteEventsBinding"
             portSharingEnabled="false">
      <security mode="None"/>
    </binding>
  </netTcpBinding>
</bindings>
```

```
<client>
  <endpoint name="customer-10.11.12.14"
            address="net.tcp://10.11.12.14:5000/RemoteEventService"
            binding="netTcpBinding" bindingConfiguration="RemoteEventsBinding"
            contract="EPiServer.Events.ServiceModel.IEventReplication"/>
</client>
```

Connect to server with IP address: 10.11.12.14

```
<services>
  <service name="EPiServer.Events.Remote.EventReplication">
    <endpoint name="RemoteEventServiceEndPoint"
              contract="EPiServer.Events.ServiceModel.IEventReplication"
              binding="netTcpBinding" bindingConfiguration="RemoteEventsBinding"
              address="net.tcp://localhost:5000/RemoteEventService" />
  </service>
</services>
```

On server with IP address: 10.11.12.13



Optimizing scalability by disabling session state

```
<sessionState mode="Off" />
```

Most of Episerver does not use session state, but the following does:

- **Episerver CMS**: Export/Import
  - **Visitor Group criteria**: Referrer, Search Word, Landing URL, Visited Category, Visited Page, Number of Visits, Submitted Form, Time on Site
- **Episerver Find**: tracking - can use current session ID, but will fall back to current user identity name.
- **Episerver Forms**: use cookies to identify visitors, and DDS as persistent storage. When DDS cannot be written to, Forms use a session state-based storage (IVolatileStorage), for example in form steps.
  - Captcha validator uses session state, but ReCaptchaValidator can be used instead.
- **Some common add-ons**:
  - Self-Optimizing Block
  - Google Analytics - will fall back to request state if session state is disabled.

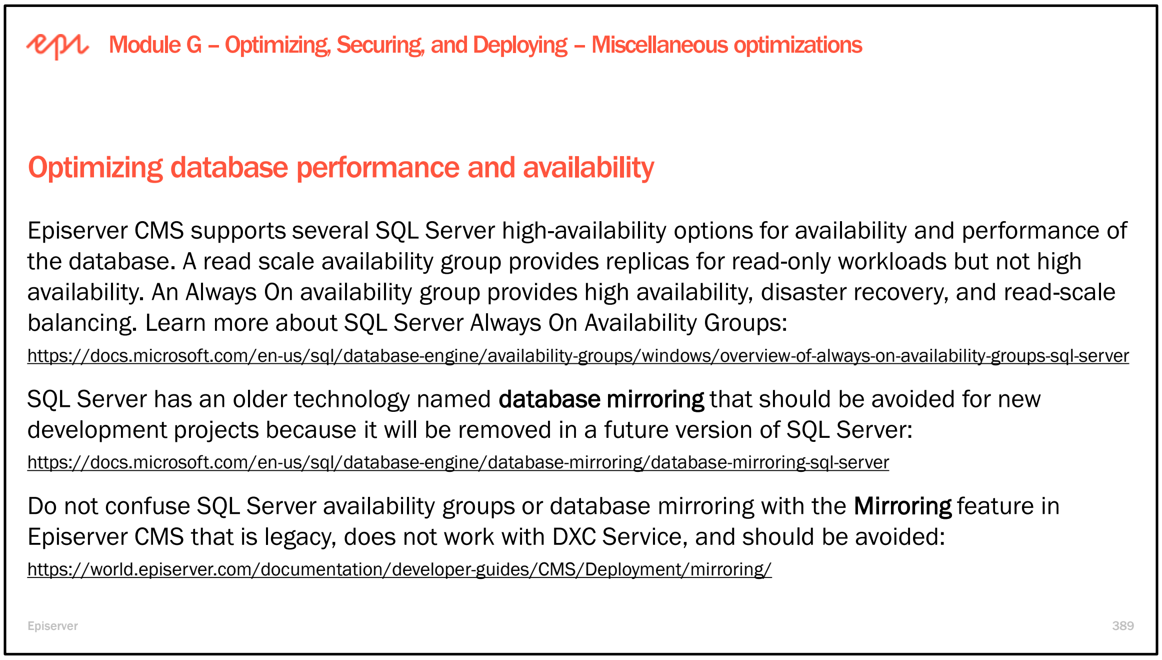
Optimizing tree performance

If the **Pages** tree is slow, you can improve its performance with a configuration setting named uiOptimizeTreeForSpeed. The default value is `false`.

If set to `true`, the **Pages** tree will not evaluate a node's access rights and language availability when it becomes visible. This will increase performance when displaying large tree structures.

```
<episerver>
  <applicationSettings
    uiOptimizeTreeForSpeed="true" ... />
```

This setting also affects the legacy trees that are used in Admin view. All nodes will show the expand icon [+] because the setting disables the checks for children to improve performance.

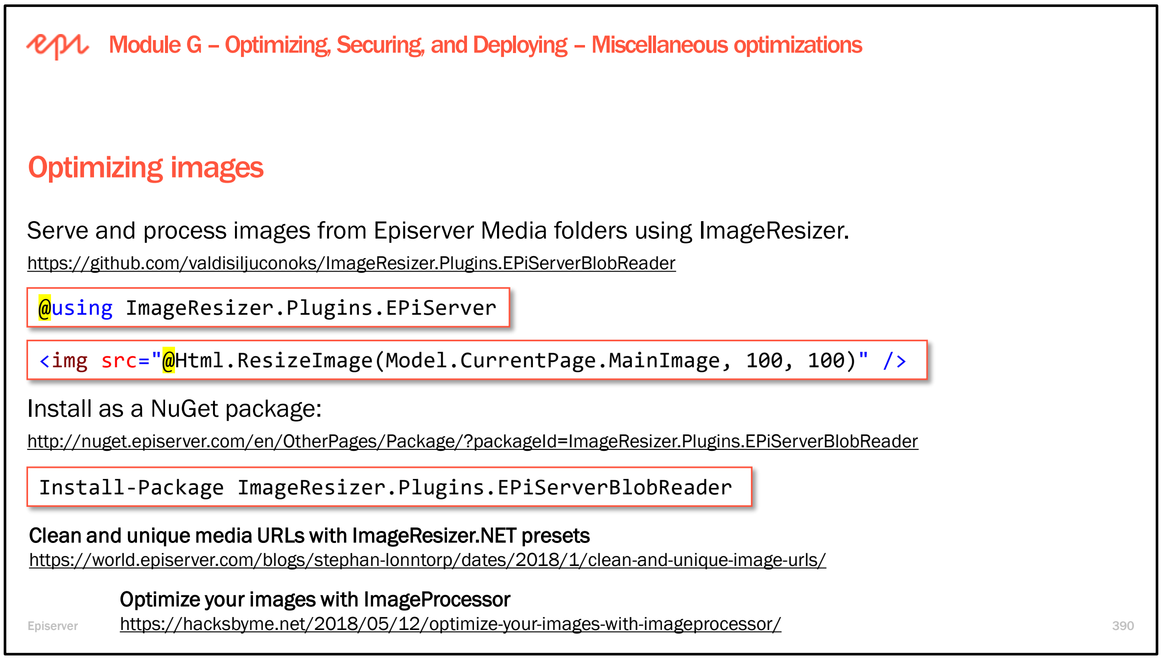


Optimizing database performance and availability

Episerver CMS supports several SQL Server high-availability options for availability and performance of the database. A read scale availability group provides replicas for read-only workloads but not high availability. An Always On availability group provides high availability, disaster recovery, and read-scale balancing. Learn more about SQL Server Always On Availability Groups:
https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/overview-of-always-on-availability-groups-sql-server

SQL Server has an older technology named **database mirroring** that should be avoided for new development projects because it will be removed in a future version of SQL Server:
https://docs.microsoft.com/en-us/sql/database-engine/database-mirroring/database-mirroring-sql-server

Do not confuse SQL Server availability groups or database mirroring with the **Mirroring** feature in Episerver CMS that is legacy, does not work with DXC Service, and should be avoided:
https://world.episerver.com/documentation/developer-guides/CMS/Deployment/mirroring/



Module G – Optimizing, Securing, and Deploying – Miscellaneous optimizations

Optimizing images

Serve and process images from Episerver Media folders using ImageResizer.

https://github.com/valdisiljuconoks/ImageResizer.Plugins.EPiServerBlobReader

```
@using ImageResizer.Plugins.EPiServer
```

```
<img src="@Html.ResizeImage(Model.CurrentPage.MainImage, 100, 100)" />
```

Install as a NuGet package:

http://nuget.episerver.com/en/OtherPages/Package/?packageId=ImageResizer.Plugins.EPiServerBlobReader

```
Install-Package ImageResizer.Plugins.EPiServerBlobReader
```

Clean and unique media URLs with ImageResizer.NET presets
https://world.episerver.com/blogs/stephan-lonntorp/dates/2018/1/clean-and-unique-image-urls/

Optimize your images with ImageProcessor
https://hacksbyme.net/2018/05/12/optimize-your-images-with-imageprocessor/

Episerver 390

Clean and unique URLs for EPiServer, with support for using ImageResizing.NET presets in a cleaner way

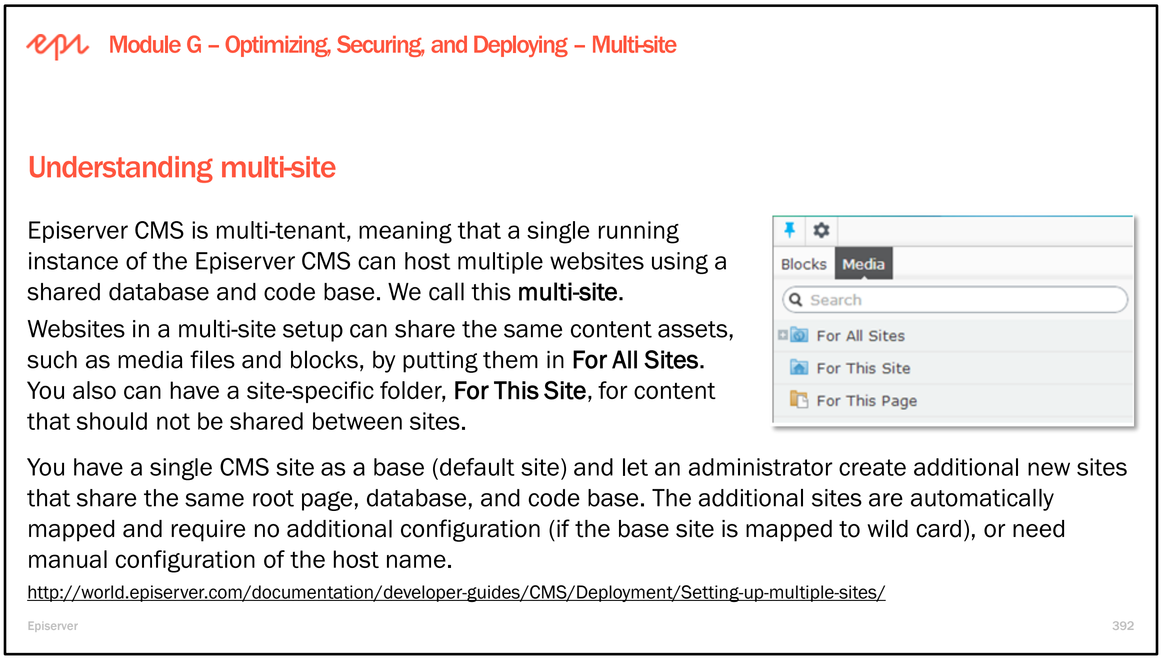
Whenever a change is made to a media item, a new hash is generated. This hash is then used to uniquely identify this version of the item. If the media item is changed, the old url will generate a permanent redirect to the new url. This is done to enable long term caching for media. By default this cache header is set to 365 days, using max-age. This can be changed by adding an appSetting with key "uufp:CacheMaxAgeTimeSpan", and a value of a timespan format string.

What does this have to do with image resizing? Nothing. But the fun doesn't stop here. To top things off, this add-on adds the ability to generate prettier URLs, without all that querystring dirt. By default, the preset keyword is "optimized", but this can be changed by adding an appSetting with key "uufp:BaseSegment", and a string value of your choosing.

If a request is made for a preset that doesn't exist, it will result in a 404.

Given that the defaults are left as-is, if there's an ImageResizer preset with the name "test", and a media item with the url "/globalassets/my-media.png", calling the url "/optimized/test/globalassets/my-media.png" will issue a redirect to "/optimized/test/<hash>/globalassets/my-media.png", where <hash> is an 8 character long calculated hash for that media item, based on its last saved date. That URL will then give you the media item, with the preset applied, and cache headers that will cache the item for a year.

https://github.com/defsteph/UniqueUrlFolderPresets

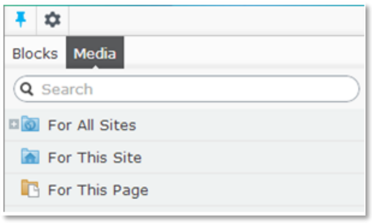


Module G – Optimizing, Securing, and Deploying – Multi-site

Understanding multi-site

Episerver CMS is multi-tenant, meaning that a single running instance of the Episerver CMS can host multiple websites using a shared database and code base. We call this **multi-site.**

Websites in a multi-site setup can share the same content assets, such as media files and blocks, by putting them in **For All Sites.** You also can have a site-specific folder, **For This Site**, for content that should not be shared between sites.



You have a single CMS site as a base (default site) and let an administrator create additional new sites that share the same root page, database, and code base. The additional sites are automatically mapped and require no additional configuration (if the base site is mapped to wild card), or need manual configuration of the host name.

http://world.episerver.com/documentation/developer-guides/CMS/Deployment/Setting-up-multiple-sites/

Episerver 392

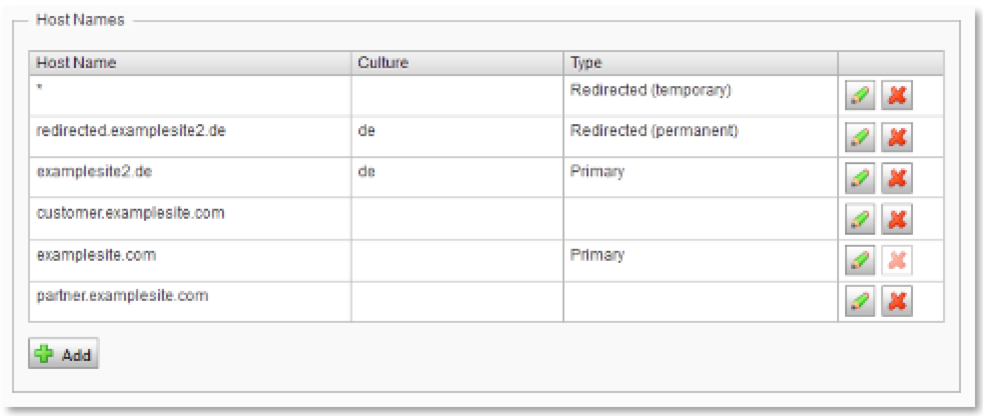
Multi-site – setup requirements

- Each site must have a unique URL and start page in the content tree.
- You cannot nest start pages.
- A multi-site license defining the maximum number of sites from which the installation is licensed.
- IIS must be configured without host headers if you add new sites without making changes to the server (because that would require an administrator to manually add new host headers when you add new sites).
- All sites must have the same root path, which must be identical to what is configured in IIS. You cannot have one site as a virtual directory and another as an IIS site.

If you configure the IIS application to respond to any host name, then you can launch new sites from the CMS admin view without additional configuration.

You need only a start page and a URL for the new site. The URL and start page are stored in the database, and new sites are automatically provisioned.

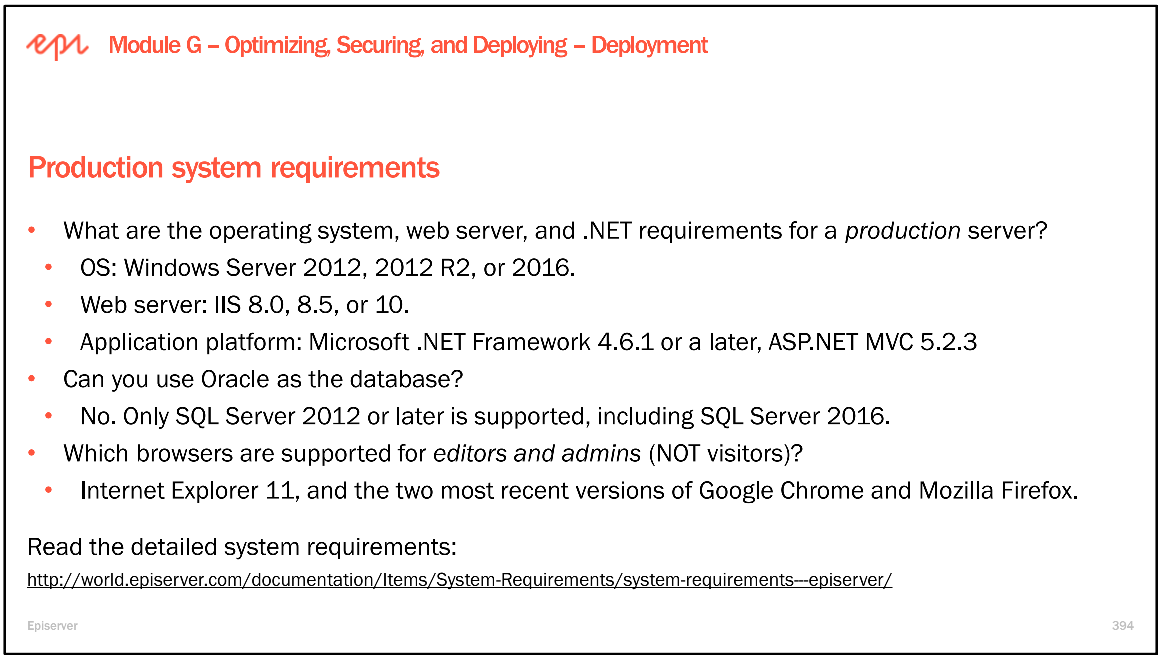
By default, one of the installed sites has the * (wild card) host mapping. You can also add additional hosts mappings, such as partner.examplesite.com or customer.examplesite.com, optionally bound to a specific language.



Host Names

Host Name	Culture	Type	
*		Redirected (temporary)	✎ ✖
redirected.examplesite2.de	de	Redirected (permanent)	✎ ✖
examplesite2.de	de	Primary	✎ ✖
customer.examplesite.com			✎ ✖
examplesite.com		Primary	✎ ✖
partner.examplesite.com			✎ ✖

Add



Module G – Optimizing, Securing, and Deploying – Deployment

Production system requirements

- What are the operating system, web server, and .NET requirements for a *production* server?
  - OS: Windows Server 2012, 2012 R2, or 2016.
  - Web server: IIS 8.0, 8.5, or 10.
  - Application platform: Microsoft .NET Framework 4.6.1 or a later, ASP.NET MVC 5.2.3
- Can you use Oracle as the database?
  - No. Only SQL Server 2012 or later is supported, including SQL Server 2016.
- Which browsers are supported for *editors and admins* (NOT visitors)?
  - Internet Explorer 11, and the two most recent versions of Google Chrome and Mozilla Firefox.

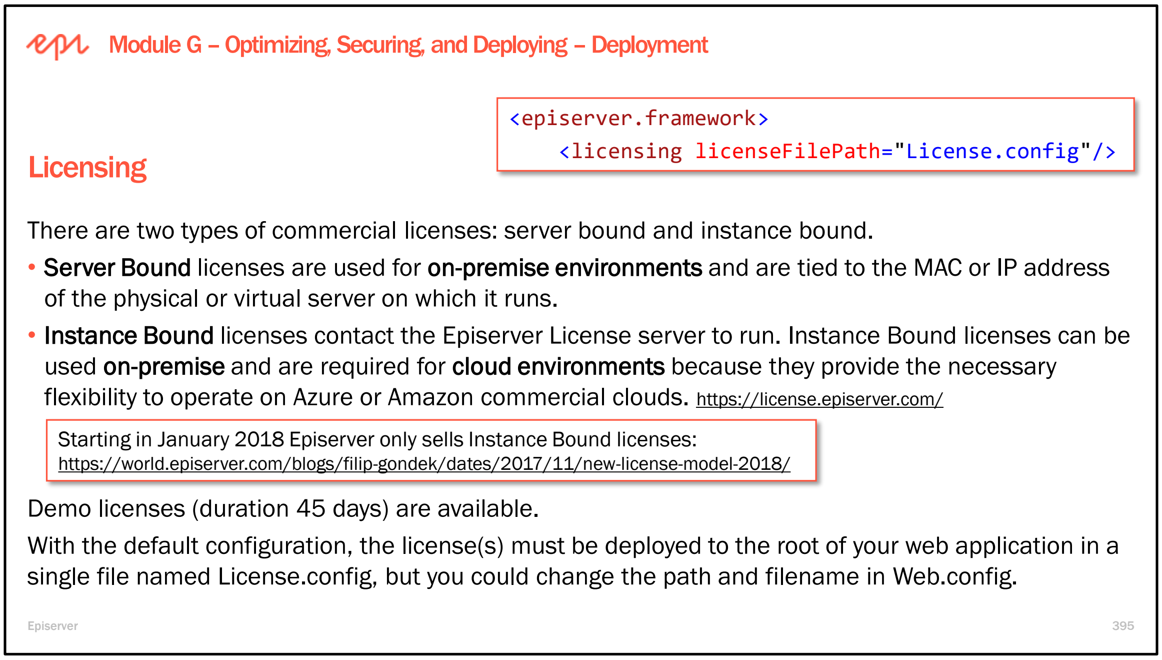
Read the detailed system requirements:
http://world.episerver.com/documentation/Items/System-Requirements/system-requirements---episerver/

Episerver 394

When configuring IIS for production deployment, include the following:

- **Web Server**
  - Common HTTP Features
    - Static Content
    - Default Document
    - HTTP Errors
    - HTTP Redirection
  - Application Development
    - ASP.NET
    - .NET Extensibility
    - ISAPI Extensions
    - ISAPI Filters
  - Security
    - Windows Authentication
    - URL Authorization
    - Request Filtering
- **Management Tools**
  - IIS Management Console
  - IIS Management Scripts and Tools
  - Management Service
  - Health and Diagnostics
    - HTTP Logging
    - Request Monitoring

Installing Commonly Used IIS Features Using Powershell
http://world.episerver.com/kb/176156



Module G – Optimizing, Securing, and Deploying – Deployment

Licensing

```
<episerver.framework>
    <licensing licenseFilePath="License.config"/>
```

There are two types of commercial licenses: server bound and instance bound.

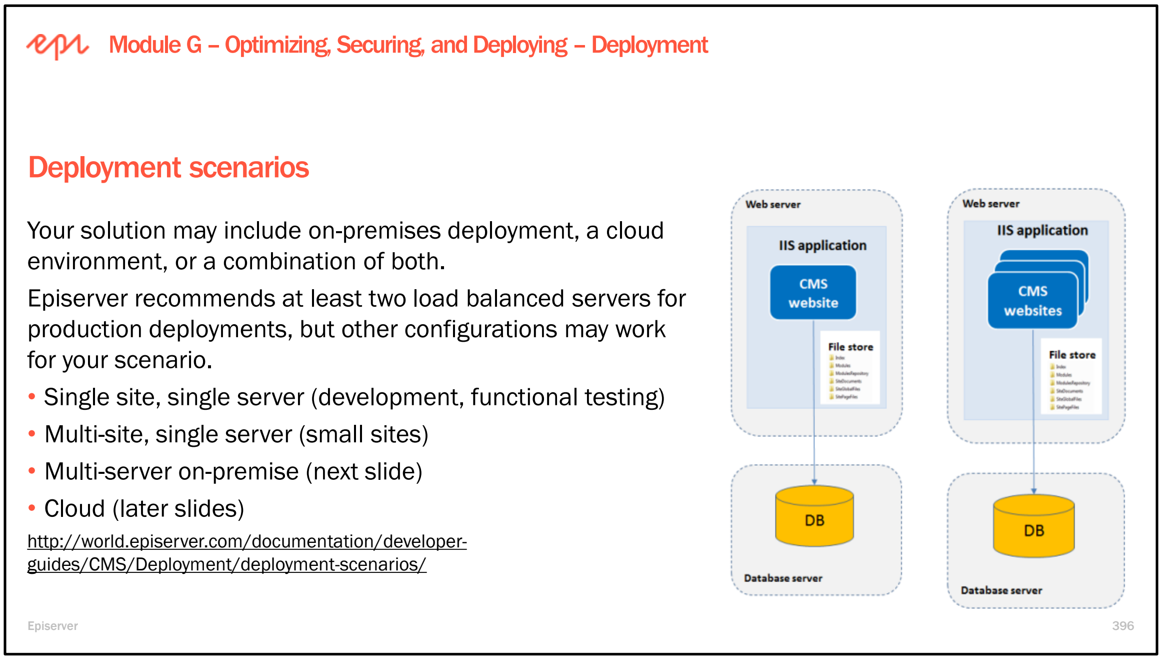
- **Server Bound** licenses are used for **on-premise environments** and are tied to the MAC or IP address of the physical or virtual server on which it runs.
- **Instance Bound** licenses contact the Episerver License server to run. Instance Bound licenses can be used **on-premise** and are required for **cloud environments** because they provide the necessary flexibility to operate on Azure or Amazon commercial clouds. https://license.episerver.com/

Starting in January 2018 Episerver only sells Instance Bound licenses:
https://world.episerver.com/blogs/filip-gondek/dates/2017/11/new-license-model-2018/

Demo licenses (duration 45 days) are available.

With the default configuration, the license(s) must be deployed to the root of your web application in a single file named License.config, but you could change the path and filename in Web.config.

Episerver 395



Module G – Optimizing, Securing, and Deploying – Deployment

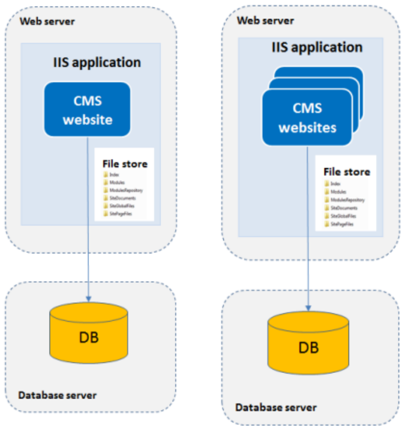
Deployment scenarios

Your solution may include on-premises deployment, a cloud environment, or a combination of both.

Episerver recommends at least two load balanced servers for production deployments, but other configurations may work for your scenario.

- Single site, single server (development, functional testing)
- Multi-site, single server (small sites)
- Multi-server on-premise (next slide)
- Cloud (later slides)

http://world.episerver.com/documentation/developer-guides/CMS/Deployment/deployment-scenarios/



Episerver

396

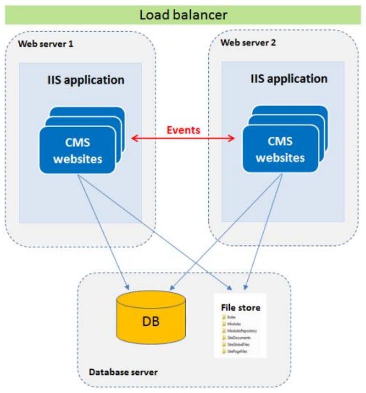
Deployment scenarios – Multi-server on-premise

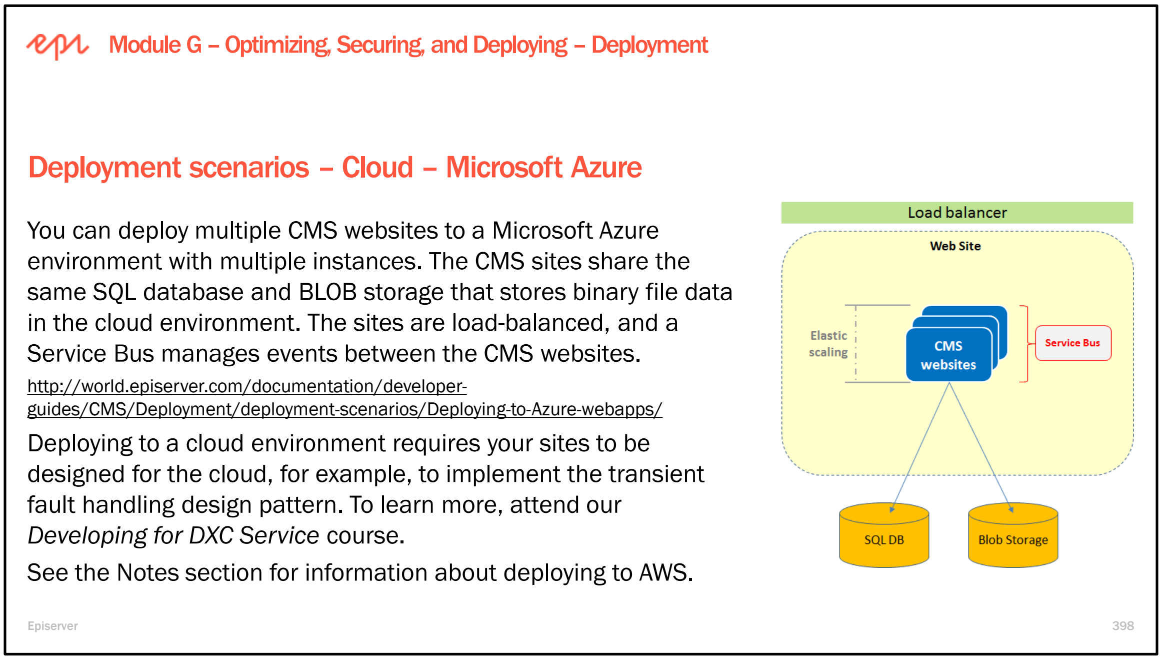
Load-balancing manages multiple CMS websites that run on two or more separate web servers. All the websites share the same database and file store for media. The websites may or may not be multisite.

When an event occurs, such as a cache removal notification triggered by content publishing, that server updates the local cache and configured servers in the setup, using Remote Events based on WCF on-premise or Azure Service Bus in DXC Service.

Deploying to Windows Servers

http://world.episerver.com/documentation/developer-guides/CMS/Deployment/deployment-scenarios/Deploying-to-Windows-Servers/





Module G – Optimizing, Securing, and Deploying – Deployment

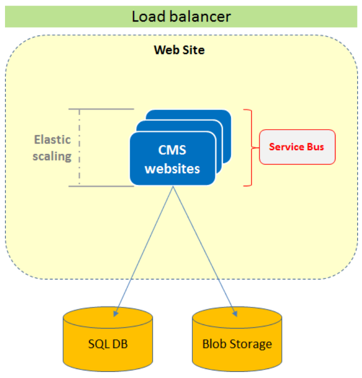
Deployment scenarios – Cloud – Microsoft Azure

You can deploy multiple CMS websites to a Microsoft Azure environment with multiple instances. The CMS sites share the same SQL database and BLOB storage that stores binary file data in the cloud environment. The sites are load-balanced, and a Service Bus manages events between the CMS websites.

http://world.episerver.com/documentation/developer-guides/CMS/Deployment/deployment-scenarios/Deploying-to-Azure-webapps/

Deploying to a cloud environment requires your sites to be designed for the cloud, for example, to implement the transient fault handling design pattern. To learn more, attend our *Developing for DXC Service* course.

See the Notes section for information about deploying to AWS.



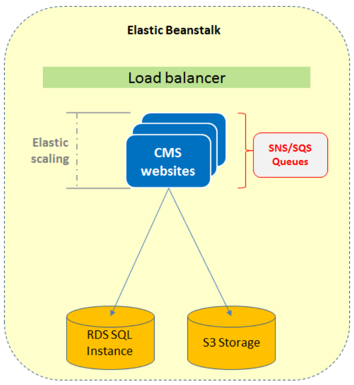
Episerver 398

Deploying an Episerver site to Azure

To deploy an Episerver CMS site to Azure, you must:

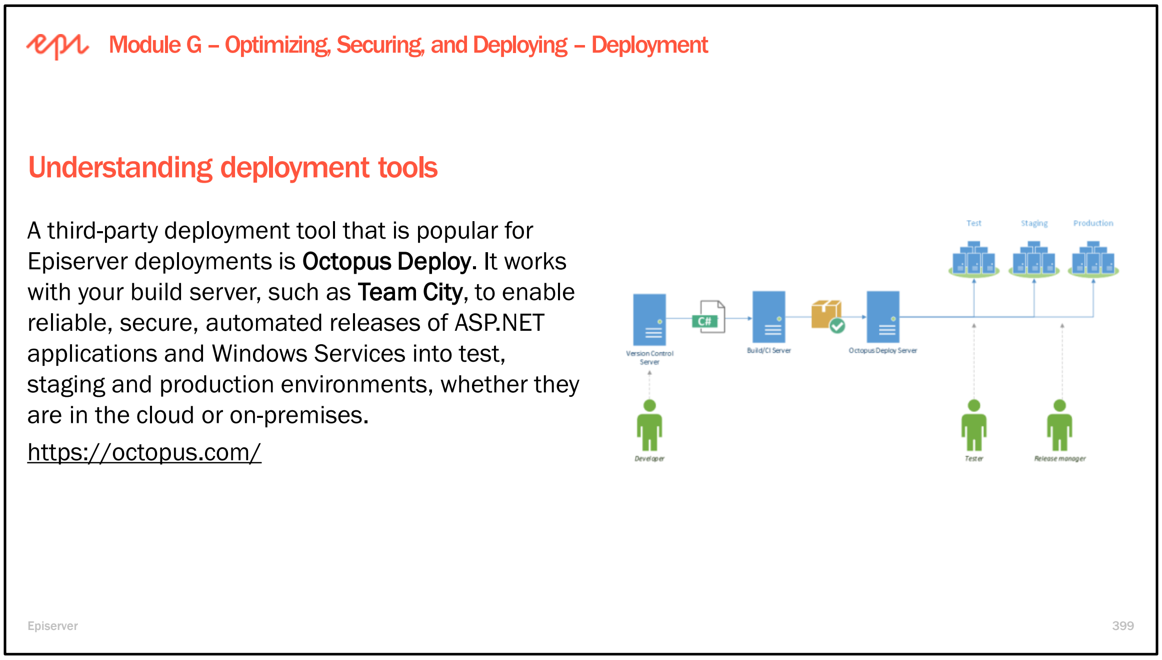
1. Install the **Episerver.Azure** NuGet package in your Episerver website project.
2. Create **Azure resources** (with DXC Service this is done for you).
3. **Transform** the Episerver website's configuration:
   a. Database connection string (from local database to SQL Database)
   b. Blob provider (from local file system to Azure Storage)
   c. Remote Events provider (from WCF to Azure Service Bus)
   d. Indexed search provider (for example, from Search to Find)
4. **Deploy** code, data, and blobs to Azure resources.

Deployment scenarios – Cloud – Amazon Web Services



The website instances share the same Amazon RDS (Relational Database Service) SQL instance, and the S3 storage in Amazon that stores binary file data in the cloud environment. The sites are load-balanced, and the SNS (Simple Notification Service)/SQS (Simple Queue Service) message queues manage events between the CMS websites. Elastic scaling lets you increase or reduce the number of CMS sites from the Elastic Beanstalk administration interface in the AWS management console.

http://world.episerver.com/documentation/developer-guides/CMS/Deployment/deployment-scenarios/Deploying-to-Amazon/

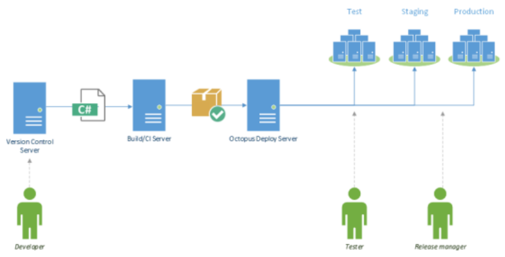


Module G – Optimizing, Securing, and Deploying – Deployment

Understanding deployment tools

A third-party deployment tool that is popular for Episerver deployments is **Octopus Deploy**. It works with your build server, such as **Team City**, to enable reliable, secure, automated releases of ASP.NET applications and Windows Services into test, staging and production environments, whether they are in the cloud or on-premises.

https://octopus.com/



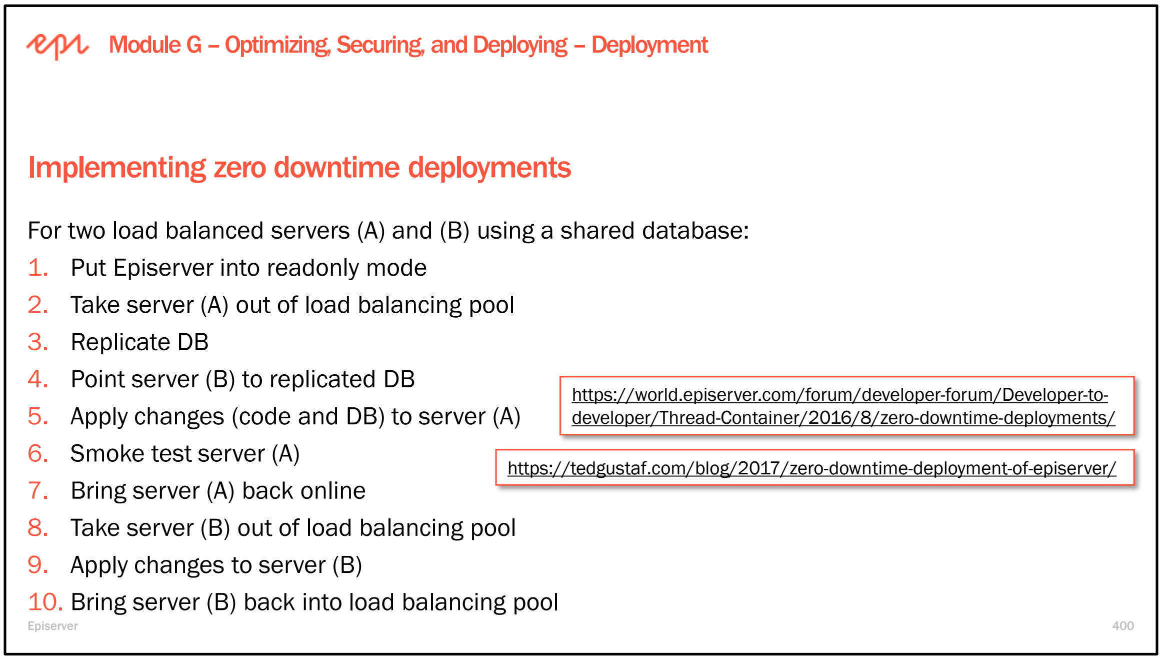
Episerver 399

XCOPY deployment

You can install Episerver CMS through XCOPY deployment by copying application files.
An XCOPY-style file transfer simplifies deployment and maintenance of Episerver sites because it does not create registry entries, and it does not register shared components.

Another benefit of the XCOPY architecture in Episerver CMS is that it does not store machine- or site-specific information in configuration files, so you can use a shared folder for multiple servers.

During development in a shared environment with source control, developers can keep configuration files checked-in and still create a site that can be duplicated in a multi-site deployment.



Module G – Optimizing, Securing, and Deploying – Deployment

Implementing zero downtime deployments

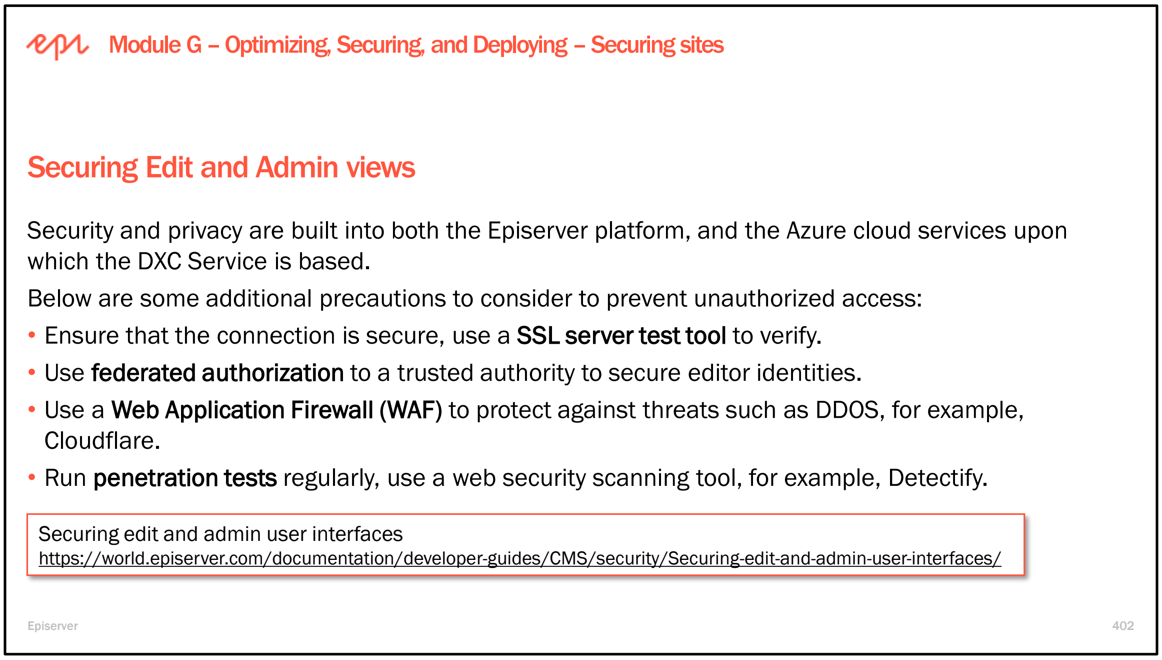
For two load balanced servers (A) and (B) using a shared database:

1. Put Episerver into readonly mode
2. Take server (A) out of load balancing pool
3. Replicate DB
4. Point server (B) to replicated DB
5. Apply changes (code and DB) to server (A)
6. Smoke test server (A)
7. Bring server (A) back online
8. Take server (B) out of load balancing pool
9. Apply changes to server (B)
10. Bring server (B) back into load balancing pool

https://world.episerver.com/forum/developer-forum/Developer-to-developer/Thread-Container/2016/8/zero-downtime-deployments/

https://tedgustaf.com/blog/2017/zero-downtime-deployment-of-episerver/

Episerver 400



Module G – Optimizing, Securing, and Deploying – Securing sites

Securing Edit and Admin views

Security and privacy are built into both the Episerver platform, and the Azure cloud services upon which the DXC Service is based.

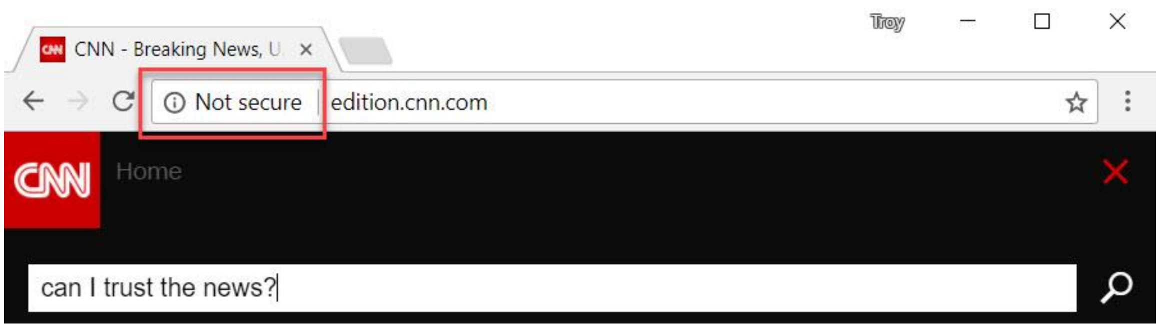
Below are some additional precautions to consider to prevent unauthorized access:

- Ensure that the connection is secure, use a **SSL server test tool** to verify.
- Use **federated authorization** to a trusted authority to secure editor identities.
- Use a **Web Application Firewall (WAF)** to protect against threats such as DDOS, for example, Cloudflare.
- Run **penetration tests** regularly, use a web security scanning tool, for example, Detectify.

Securing edit and admin user interfaces
https://world.episerver.com/documentation/developer-guides/CMS/security/Securing-edit-and-admin-user-interfaces/

Episerver 402

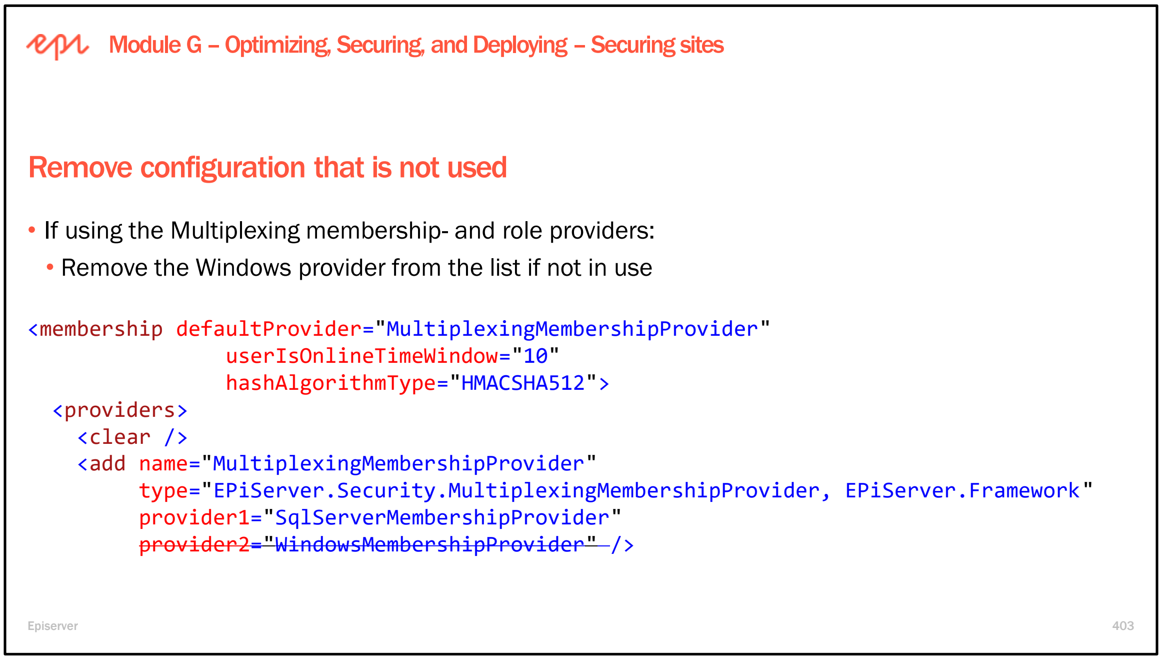
All websites should now use HTTPS. If not, Chrome shows them as **Not secure** if the website has any input boxes:



Set HTTP Only
Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side script accessing the protected cookie (if the browser supports it).
https://www.owasp.org/index.php/HttpOnly#Using_.NET_to_Set_HttpOnly

The 6-Step "Happy Path" to HTTPS
https://www.troyhunt.com/the-6-step-happy-path-to-https/

Free eBook: OWASP Top 10 for .NET developers
https://www.troyhunt.com/free-ebook-owasp-top-10-for-net/



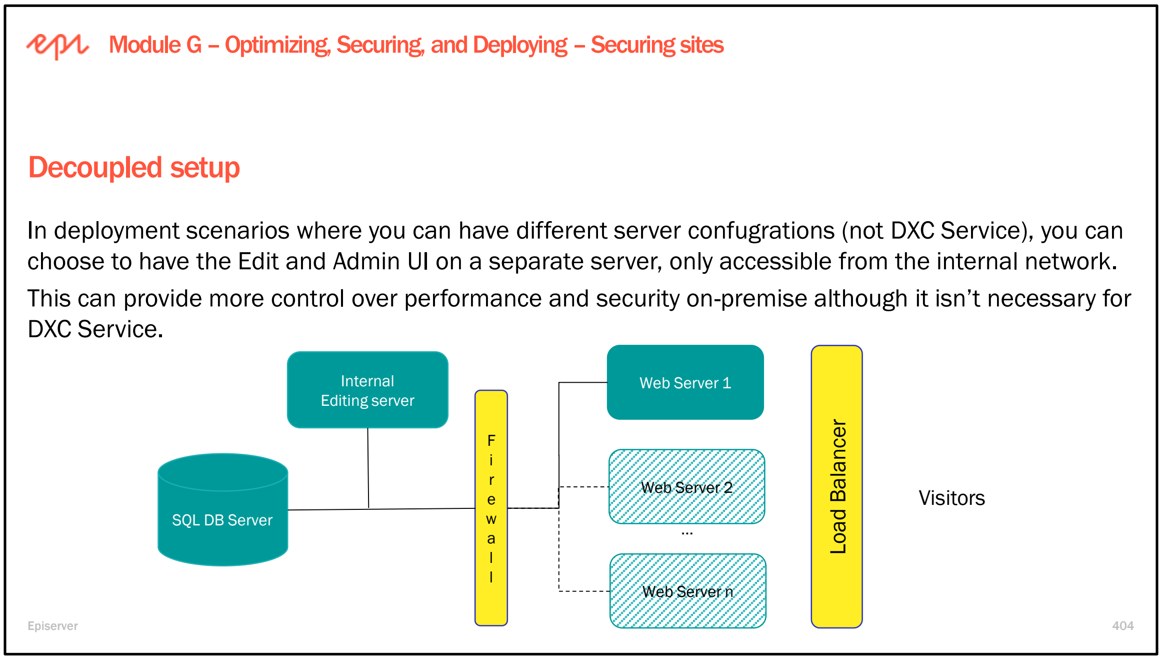
Module G – Optimizing, Securing, and Deploying – Securing sites

Remove configuration that is not used

- If using the Multiplexing membership- and role providers:
  - Remove the Windows provider from the list if not in use

```
<membership defaultProvider="MultiplexingMembershipProvider"
                userIsOnlineTimeWindow="10"
                hashAlgorithmType="HMACSHA512">
  <providers>
    <clear />
    <add name="MultiplexingMembershipProvider"
         type="EPiServer.Security.MultiplexingMembershipProvider, EPiServer.Framework"
         provider1="SqlServerMembershipProvider"
         ~~provider2="WindowsMembershipProvider"~~ />
```

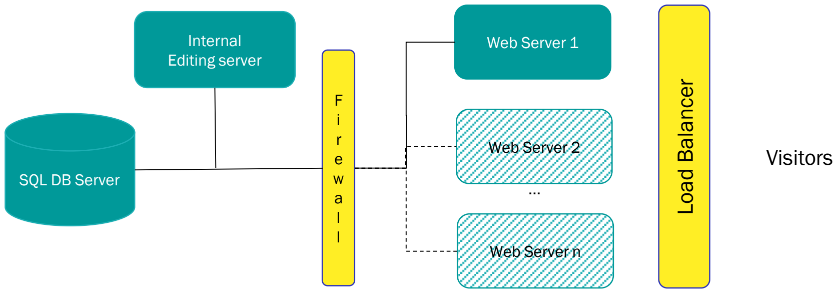
Episerver 403

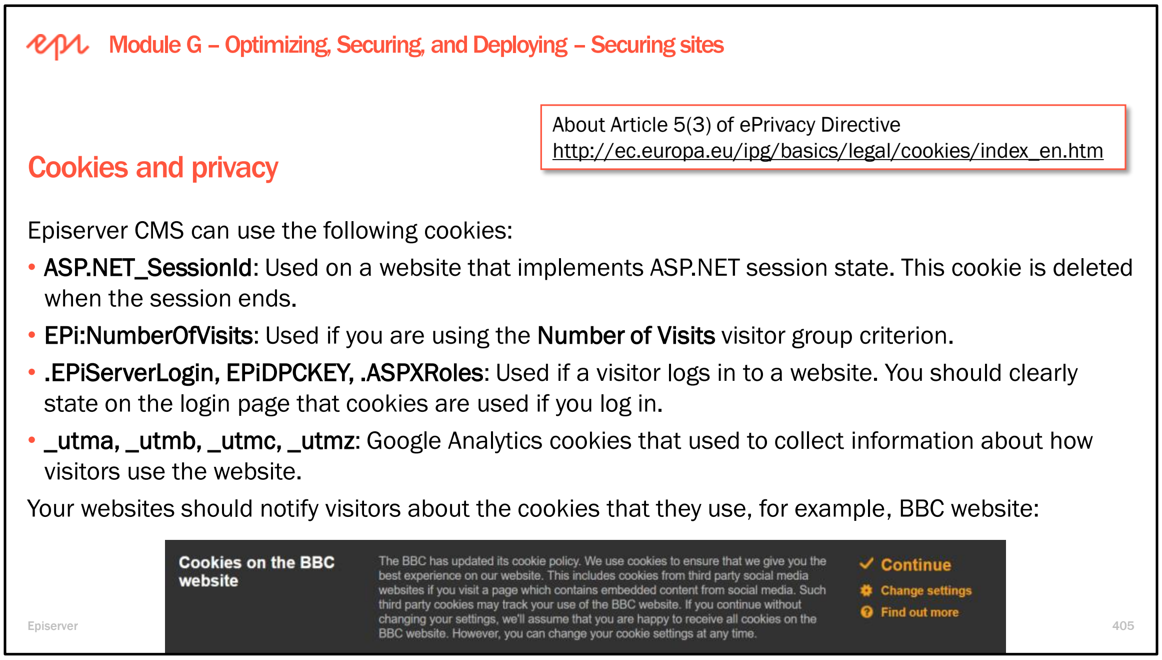


Decoupled setup

In deployment scenarios where you can have different server confugrations (not DXC Service), you can choose to have the Edit and Admin UI on a separate server, only accessible from the internal network.

This can provide more control over performance and security on-premise although it isn't necessary for DXC Service.





Module G – Optimizing, Securing, and Deploying – Securing sites

About Article 5(3) of ePrivacy Directive
http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm

Cookies and privacy

Episerver CMS can use the following cookies:

- **ASP.NET_SessionId**: Used on a website that implements ASP.NET session state. This cookie is deleted when the session ends.
- **EPi:NumberOfVisits**: Used if you are using the **Number of Visits** visitor group criterion.
- **.EPiServerLogin, EPiDPCKEY, .ASPXRoles**: Used if a visitor logs in to a website. You should clearly state on the login page that cookies are used if you log in.
- **_utma, _utmb, _utmc, _utmz**: Google Analytics cookies that used to collect information about how visitors use the website.

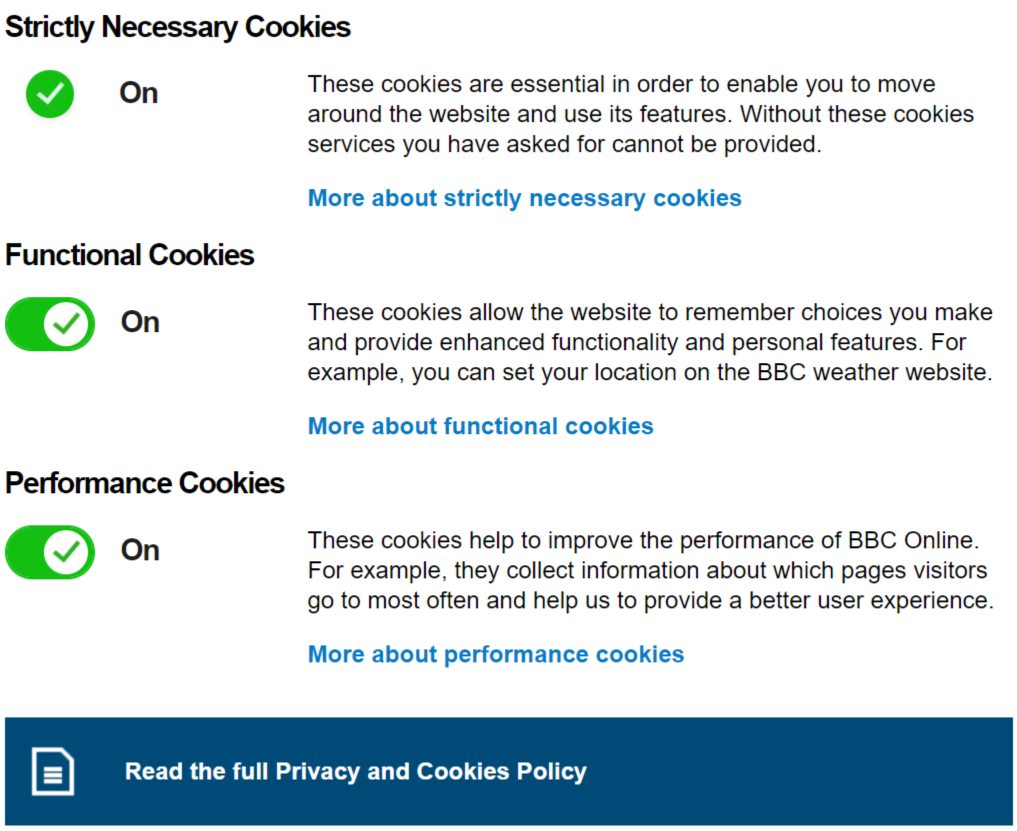
Your websites should notify visitors about the cookies that they use, for example, BBC website:

Cookies on the BBC website

The BBC has updated its cookie policy. We use cookies to ensure that we give you the best experience on our website. This includes cookies from third party social media websites if you visit a page which contains embedded content from social media. Such third party cookies may track your use of the BBC website. If you continue without changing your settings, we'll assume that you are happy to receive all cookies on the BBC website. However, you can change your cookie settings at any time.

Episerver 405

Example of cookie settings from BBC News website



Strictly Necessary Cookies

✓ On — These cookies are essential in order to enable you to move around the website and use its features. Without these cookies services you have asked for cannot be provided.

More about strictly necessary cookies

Functional Cookies

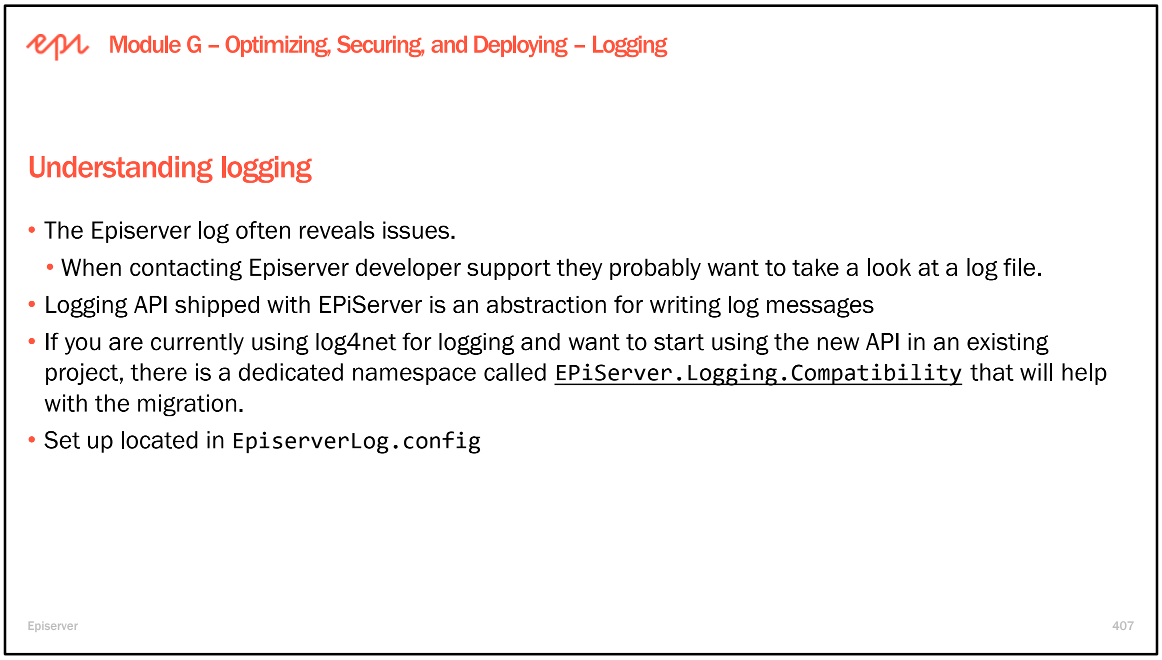
✓ On — These cookies allow the website to remember choices you make and provide enhanced functionality and personal features. For example, you can set your location on the BBC weather website.

More about functional cookies

Performance Cookies

✓ On — These cookies help to improve the performance of BBC Online. For example, they collect information about which pages visitors go to most often and help us to provide a better user experience.

More about performance cookies



Module G – Optimizing, Securing, and Deploying – Logging

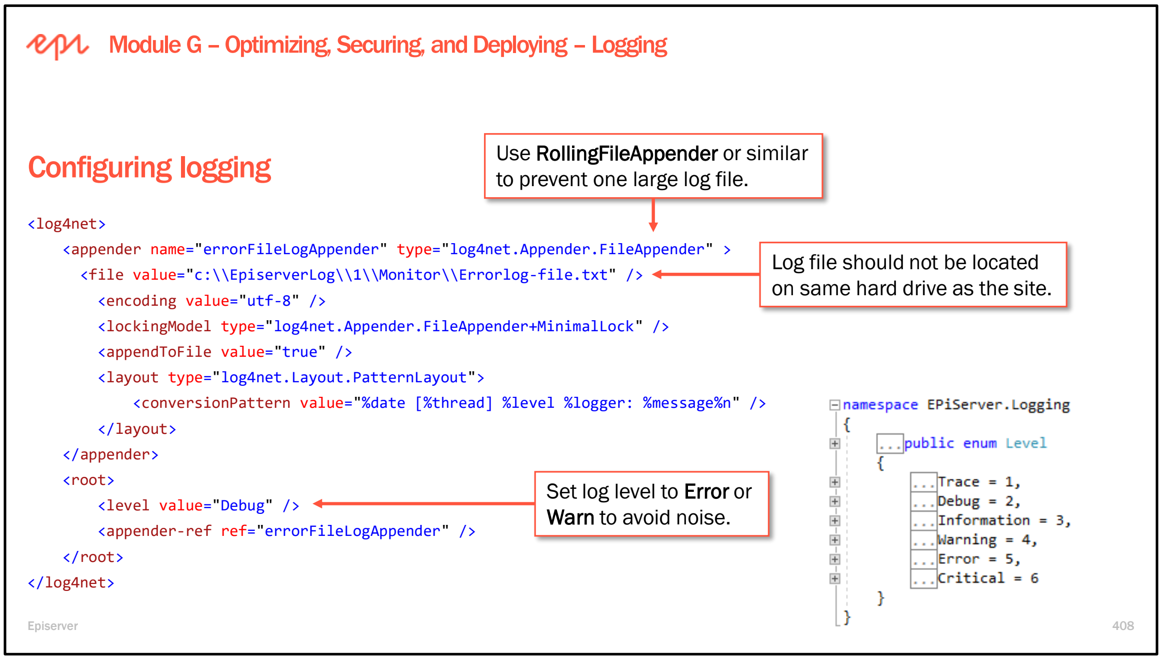
Understanding logging

- The Episerver log often reveals issues.
  - When contacting Episerver developer support they probably want to take a look at a log file.
- Logging API shipped with EPiServer is an abstraction for writing log messages
- If you are currently using log4net for logging and want to start using the new API in an existing project, there is a dedicated namespace called `EPiServer.Logging.Compatibility` that will help with the migration.
- Set up located in `EpiserverLog.config`

Episerver 407

Log level: Error, Debug, Info or All
When logging too much it reduces performance, makes the log files large and hard to read. Log only what you need.
Store log files on separate drive
You might run out of space
Split up into log files each day
One good way to make the files more easy to read
log4net.Appender.RollingFileAppender

https://world.episerver.com/documentation/developer-guides/CMS/logging/



Module G – Optimizing, Securing, and Deploying – Logging

Configuring logging

```
<log4net>
    <appender name="errorFileLogAppender" type="log4net.Appender.FileAppender" >
      <file value="c:\\EpiserverLog\\1\\Monitor\\Errorlog-file.txt" />
        <encoding value="utf-8" />
        <lockingModel type="log4net.Appender.FileAppender+MinimalLock" />
        <appendToFile value="true" />
        <layout type="log4net.Layout.PatternLayout">
            <conversionPattern value="%date [%thread] %level %logger: %message%n" />
        </layout>
    </appender>
    <root>
        <level value="Debug" />
        <appender-ref ref="errorFileLogAppender" />
    </root>
</log4net>
```

Use **RollingFileAppender** or similar to prevent one large log file.

Log file should not be located on same hard drive as the site.

Set log level to **Error** or **Warn** to avoid noise.

```
namespace EPiServer.Logging
{
    public enum Level
    {
        Trace = 1,
        Debug = 2,
        Information = 3,
        Warning = 4,
        Error = 5,
        Critical = 6
    }
}
```

Episerver

408

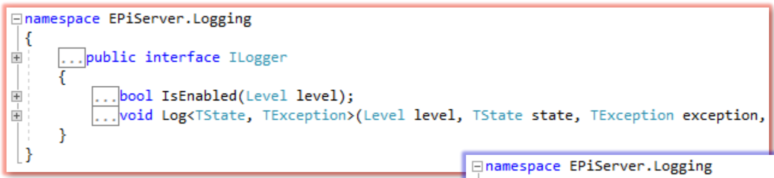
Module G – Optimizing, Securing, and Deploying – Logging

Writing to the log

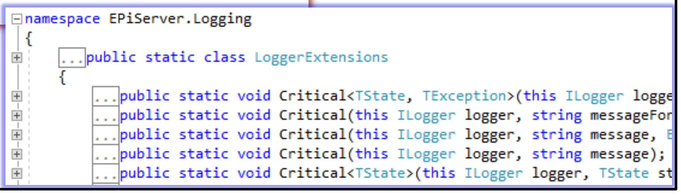
Get the logger service with `LogManager`:

```
using EPiServer.Logging;
private readonly ILogger logger = LogManager.GetLogger();
```

CMS 11 breaking change
It is no longer supported to get an `ILogger` instance from IOC container.



```
namespace EPiServer.Logging
{
    ...public interface ILogger
    {
        ...bool IsEnabled(Level level);
        ...void Log<TState, TException>(Level level, TState state, TException exception,
    }
}
```

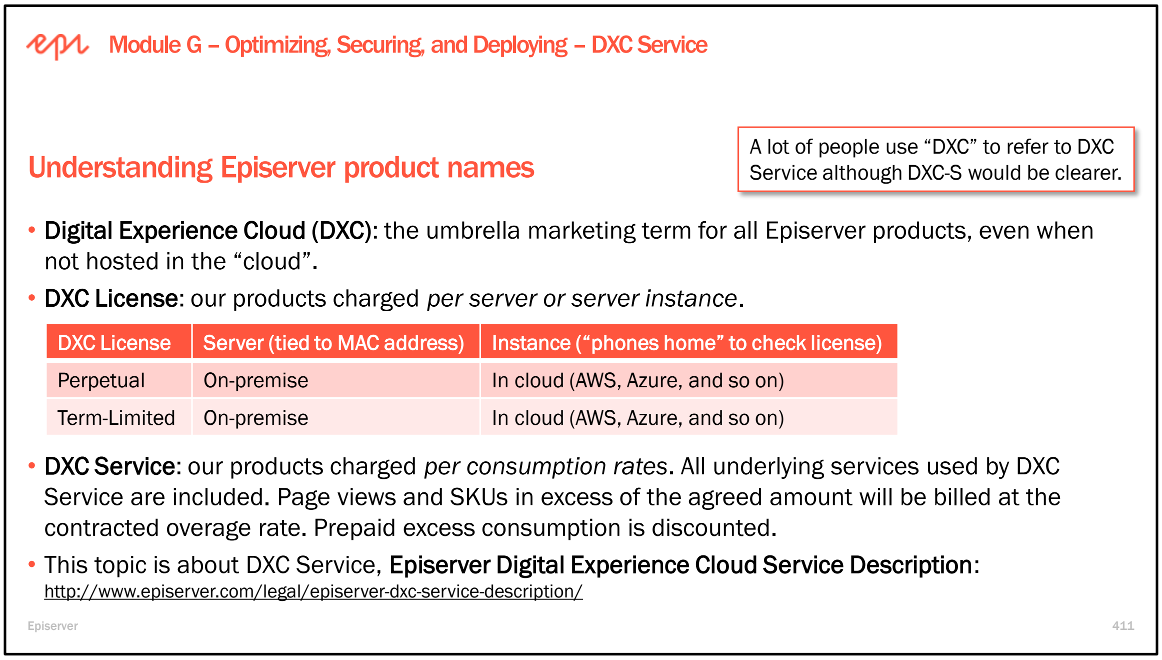


```
namespace EPiServer.Logging
{
    ...public static class LoggerExtensions
    {
        ...public static void Critical<TState, TException>(this ILogger logge
        ...public static void Critical(this ILogger logger, string messageFor
        ...public static void Critical(this ILogger logger, string message, E
        ...public static void Critical(this ILogger logger, string message);
        ...public static void Critical<TState>(this ILogger logger, TState st
```

Write to the log:

```
logger.Critical("My message");
```

Episerver



Module G – Optimizing, Securing, and Deploying – DXC Service

Understanding Episerver product names

A lot of people use "DXC" to refer to DXC Service although DXC-S would be clearer.

- **Digital Experience Cloud (DXC):** the umbrella marketing term for all Episerver products, even when not hosted in the "cloud".
- **DXC License**: our products charged *per server or server instance*.

DXC License	Server (tied to MAC address)	Instance ("phones home" to check license)
Perpetual	On-premise	In cloud (AWS, Azure, and so on)
Term-Limited	On-premise	In cloud (AWS, Azure, and so on)

- **DXC Service**: our products charged *per consumption rates*. All underlying services used by DXC Service are included. Page views and SKUs in excess of the agreed amount will be billed at the contracted overage rate. Prepaid excess consumption is discounted.
- This topic is about DXC Service, **Episerver Digital Experience Cloud Service Description**: http://www.episerver.com/legal/episerver-dxc-service-description/

Episerver 411

Module G – Optimizing, Securing, and Deploying – DXC Service

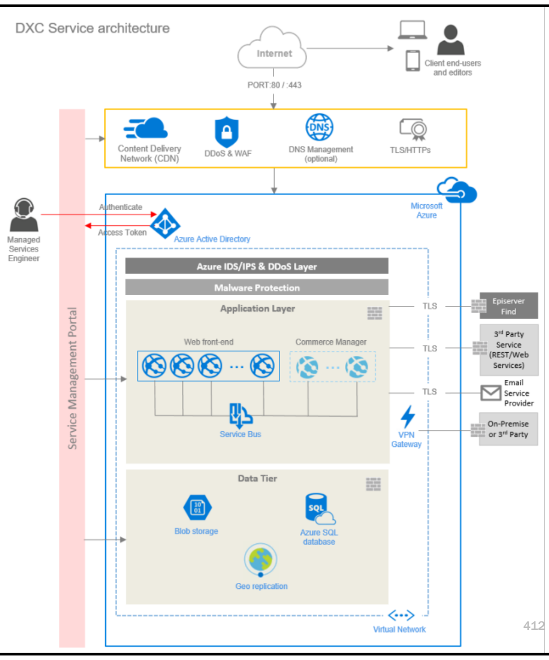
Understanding DXC-S architecture

DXC-S **Digital Commerce** package includes everything in this diagram.

DXC-S **Digital Marketing** package is the same but without the Commerce parts.

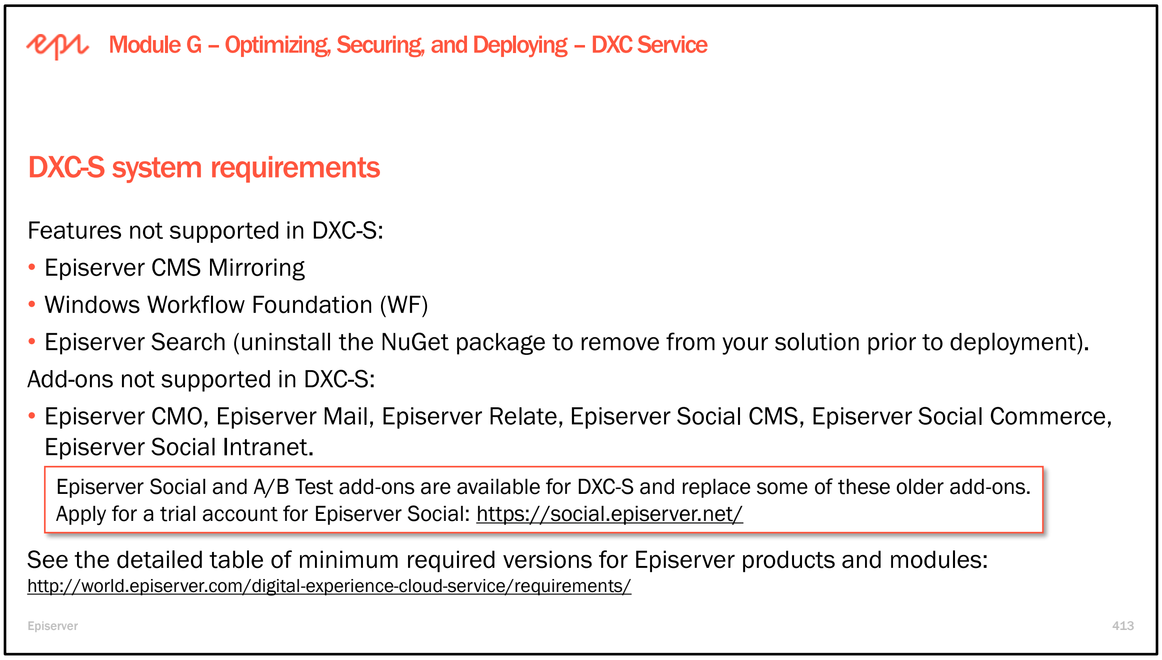
Specific technologies can be replaced with better alternatives in the future. Your code should target Episerver APIs, not a specific platform-as-a-service like Microsoft Azure Service Bus.

http://world.episerver.com/digital-experience-cloud-service/introduction/



Episerver

412



Module G – Optimizing, Securing, and Deploying – DXC Service

DXC-S system requirements

Features not supported in DXC-S:

- Episerver CMS Mirroring
- Windows Workflow Foundation (WF)
- Episerver Search (uninstall the NuGet package to remove from your solution prior to deployment).

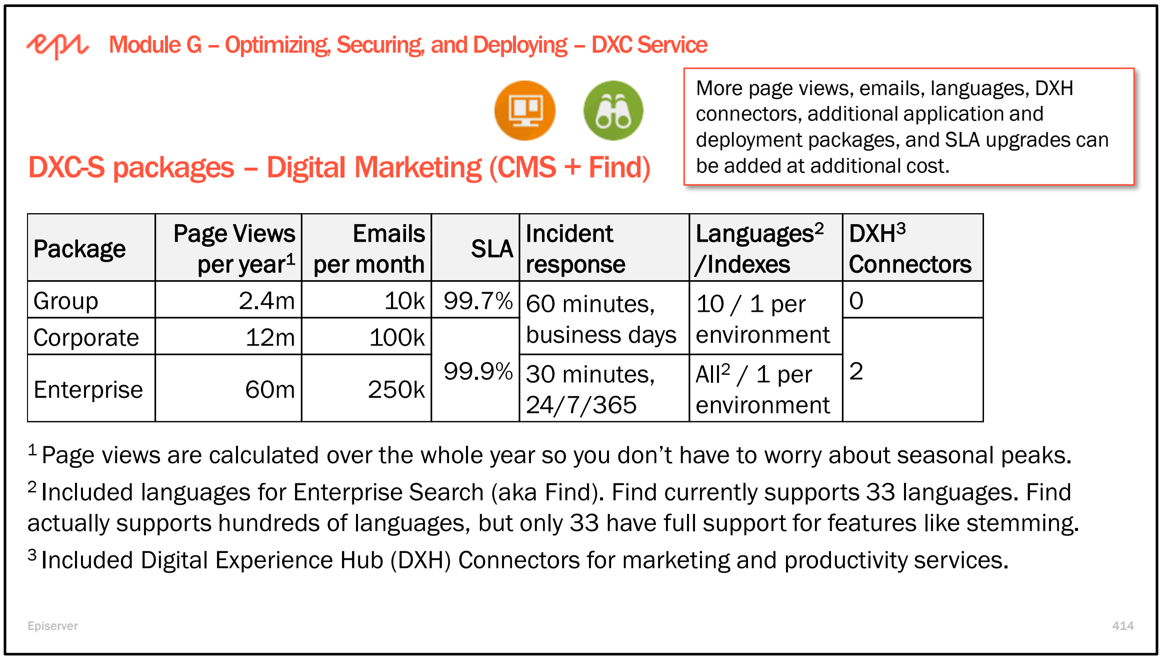
Add-ons not supported in DXC-S:

- Episerver CMO, Episerver Mail, Episerver Relate, Episerver Social CMS, Episerver Social Commerce, Episerver Social Intranet.

Episerver Social and A/B Test add-ons are available for DXC-S and replace some of these older add-ons. Apply for a trial account for Episerver Social: https://social.episerver.net/

See the detailed table of minimum required versions for Episerver products and modules: http://world.episerver.com/digital-experience-cloud-service/requirements/

Episerver 413



Module G – Optimizing, Securing, and Deploying – DXC Service

DXC-S packages – Digital Marketing (CMS + Find)

More page views, emails, languages, DXH connectors, additional application and deployment packages, and SLA upgrades can be added at additional cost.

Package	Page Views per year[1]	Emails per month	SLA	Incident response	Languages[2] /Indexes	DXH[3] Connectors
Group	2.4m	10k	99.7%	60 minutes, business days	10 / 1 per environment	0
Corporate	12m	100k	99.9%			2
Enterprise	60m	250k		30 minutes, 24/7/365	All[2] / 1 per environment	

[1] Page views are calculated over the whole year so you don't have to worry about seasonal peaks.

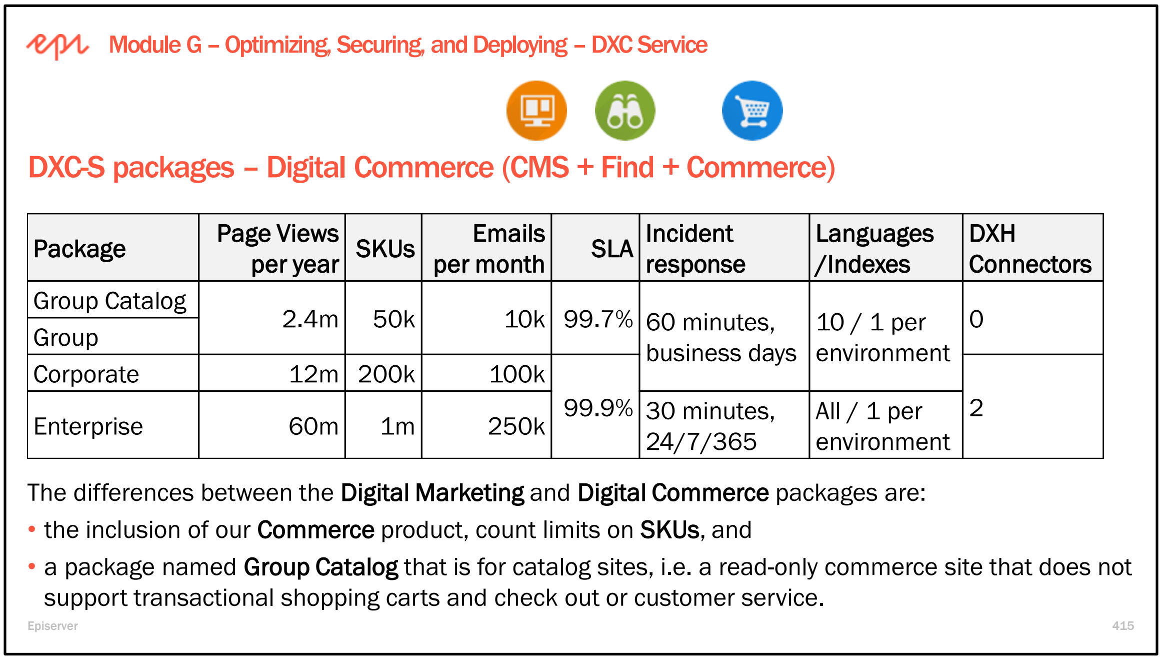
[2] Included languages for Enterprise Search (aka Find). Find currently supports 33 languages. Find actually supports hundreds of languages, but only 33 have full support for features like stemming.

[3] Included Digital Experience Hub (DXH) Connectors for marketing and productivity services.

Episerver 414

Understanding the Master Package and Additional Packages
With DXC-S, you pay for a **Master Package** and one or more **Additional Packages** and **Add-Ons**.
A **Master Package** includes the following:
A number of **page views per year** (so that you don't overpay for seasonal peak traffic).
A number of **SKUs per environment** that are managed and indexed for search.
A number of **transactional emails per month**.
An advanced **firewall** for attack prevention.
A Content Delivery Network (**CDN**) for caching, improving scalability and responsiveness.
A multi-domain **SSL** certificate.

Additional Packages can be added for a fraction of a Master Package cost. Additional numbers of page views and SKUs are merged into a single "bucket". Prepay and overage charges are per 25k page views and 20k SKUs. Prepay is half the cost of overage.



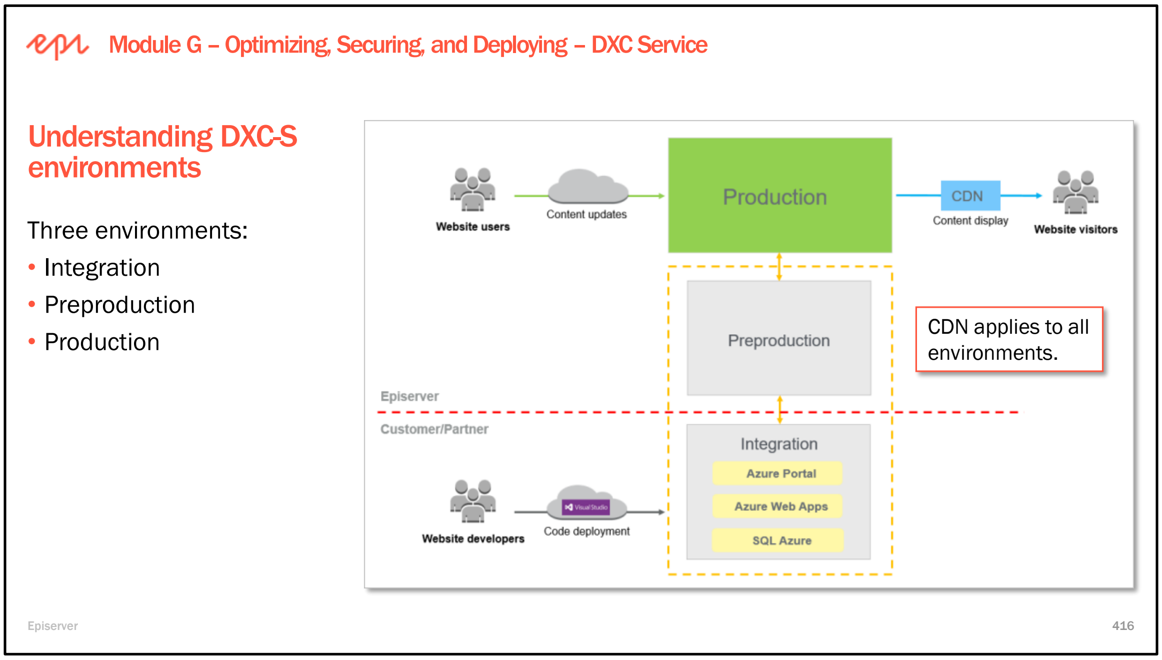
DXC-S packages – Digital Commerce (CMS + Find + Commerce)

Package	Page Views per year	SKUs	Emails per month	SLA	Incident response	Languages /Indexes	DXH Connectors
Group Catalog	2.4m	50k	10k	99.7%	60 minutes, business days	10 / 1 per environment	0
Group							
Corporate	12m	200k	100k	99.9%			2
Enterprise	60m	1m	250k		30 minutes, 24/7/365	All / 1 per environment	

The differences between the **Digital Marketing** and **Digital Commerce** packages are:

- the inclusion of our **Commerce** product, count limits on **SKUs**, and
- a package named **Group Catalog** that is for catalog sites, i.e. a read-only commerce site that does not support transactional shopping carts and check out or customer service.

Episerver 415

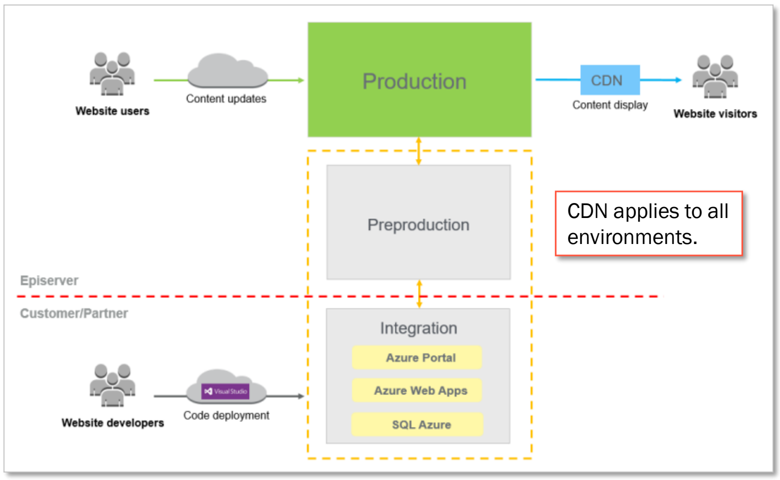


Module G – Optimizing, Securing, and Deploying – DXC Service

Understanding DXC-S environments

Three environments:

- Integration
- Preproduction
- Production



Episerver 416

Partners and customer deploy the full solution to the **Integration** environment, as daily builds or continuous releases. The Integration environment has fixed configuration and no automatic scaling.
Here developers can:

- validate integrations with external systems,
- perform functional testing, and
- add initial content in the case of a first-time deployment.

Episerver uses the **Preproduction** environment to:

- test Production deployment,
- verify performance and operational functionality.

Developers may also use the Preproduction environment for:

- User Acceptance Testing (UAT),
- performance and load testing,
- approved penetration testing.

The Preproduction environment scales automatically, and deployment is done by Episerver.

In the **Production** environment:

- **content editors** will author content, using the Episerver content publishing flow or Projects, and
- **visitors** can access public content.

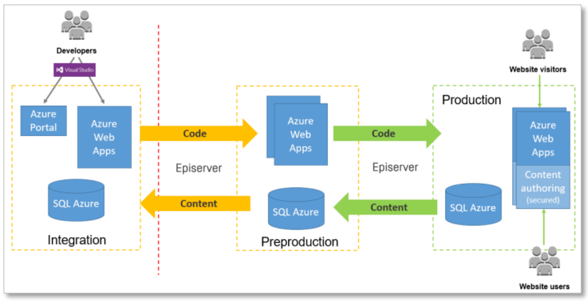
The **Production** environment scales automatically, and deployment is done by Episerver.

You need to contact Episerver to register a ticket to initiate deployment to Preproduction and Production environments, since this can only be done by Episerver.

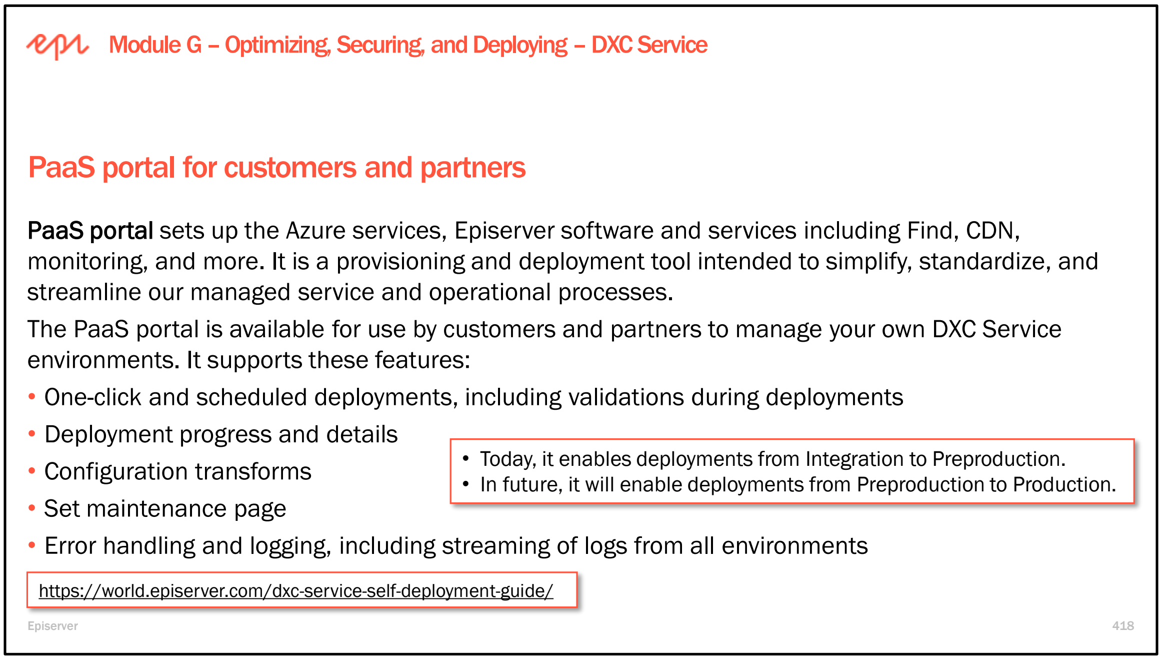
All instances in the Production environment are identical, so you cannot follow Episerver's recommended practice for on premise deployment, i.e., for extra security, create a server for editors and admins that is separate from the load balanced servers used by visitors.

Understanding the role of Managed Services

- Set up of environments
- First-time deployment of code, content and configuration to Preproduction and Production.
- Initial troubleshooting and roll-back if issues arise.
- Continuous deployment of code to Preproduction and Production after go-live.
- Deployment of production content back to Preproduction and Integration.



You can purchase additional Integration environments but they won't be part of the deployment chain.



Module G – Optimizing, Securing, and Deploying – DXC Service

PaaS portal for customers and partners

PaaS portal sets up the Azure services, Episerver software and services including Find, CDN, monitoring, and more. It is a provisioning and deployment tool intended to simplify, standardize, and streamline our managed service and operational processes.

The PaaS portal is available for use by customers and partners to manage your own DXC Service environments. It supports these features:

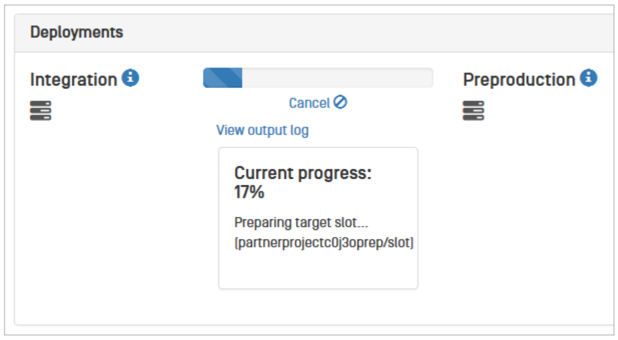
- One-click and scheduled deployments, including validations during deployments
- Deployment progress and details
- Configuration transforms
- Set maintenance page
- Error handling and logging, including streaming of logs from all environments

- Today, it enables deployments from Integration to Preproduction.
- In future, it will enable deployments from Preproduction to Production.

https://world.episerver.com/dxc-service-self-deployment-guide/

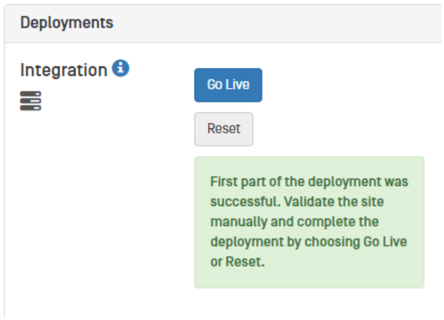
Episerver 418

Performing deployments



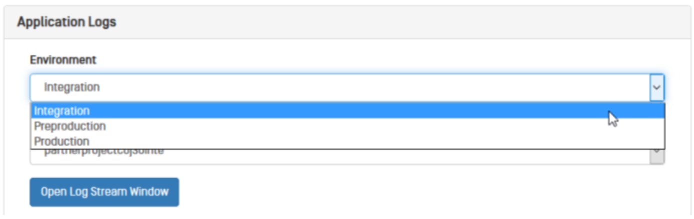
Current progress: 17%

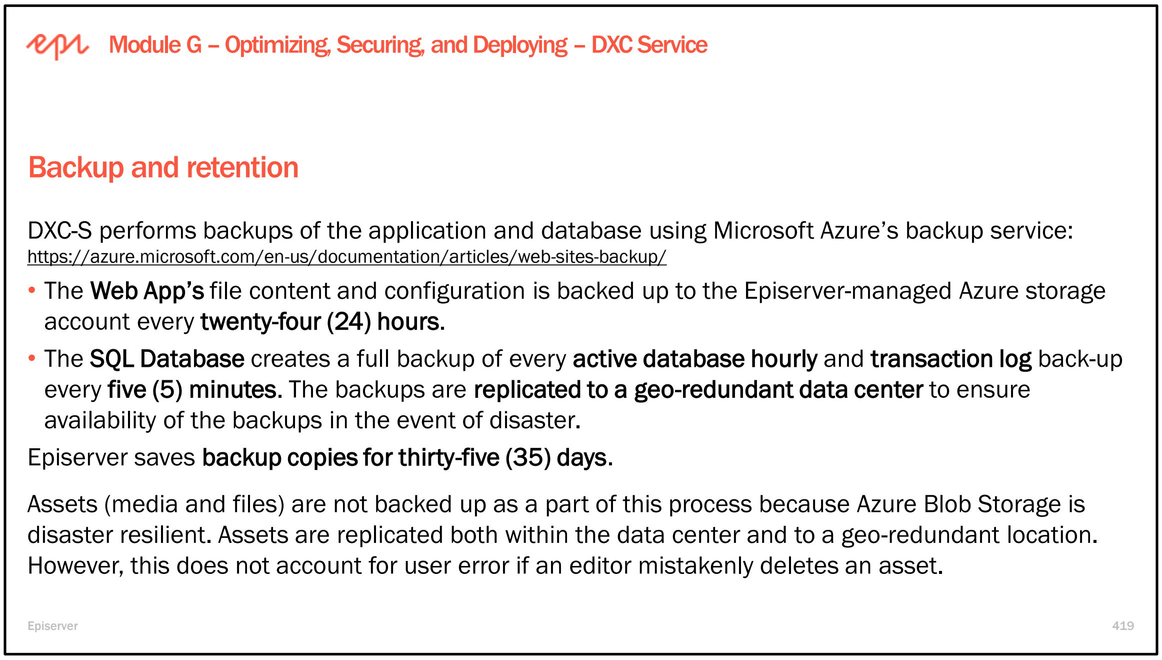
Preparing target slot... [partnerprojectc0j3oprep/slot]



First part of the deployment was successful. Validate the site manually and complete the deployment by choosing Go Live or Reset.

Viewing application logs





Module G – Optimizing, Securing, and Deploying – DXC Service

Backup and retention

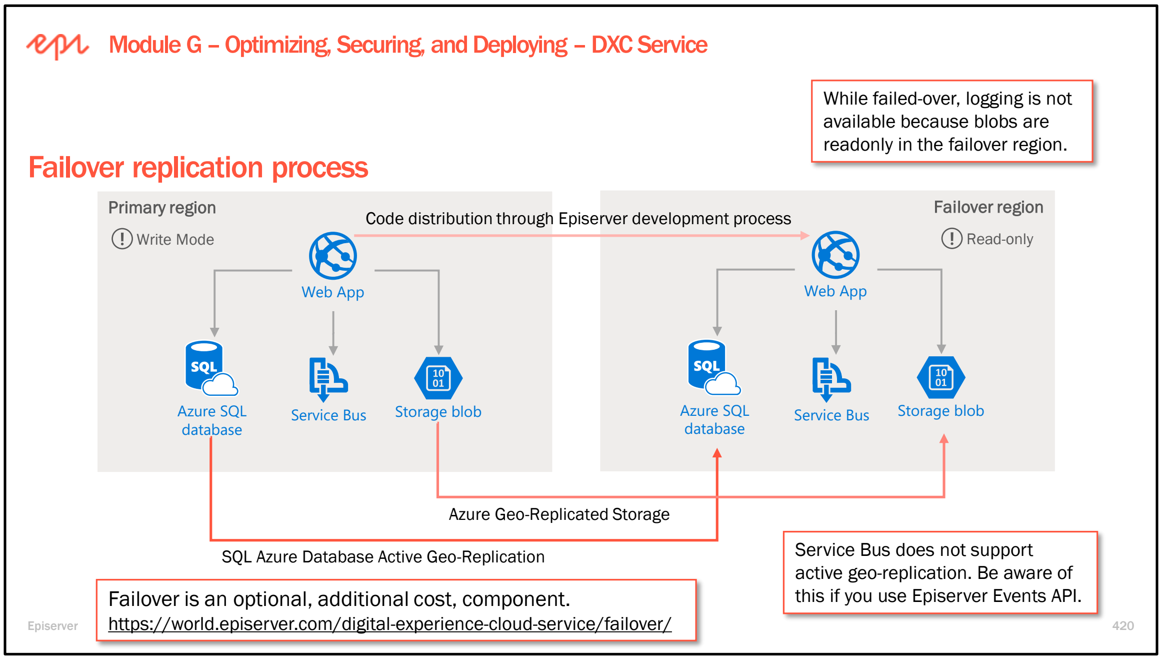
DXC-S performs backups of the application and database using Microsoft Azure's backup service: https://azure.microsoft.com/en-us/documentation/articles/web-sites-backup/

- The **Web App's** file content and configuration is backed up to the Episerver-managed Azure storage account every **twenty-four (24) hours.**
- The **SQL Database** creates a full backup of every **active database hourly** and **transaction log** back-up every **five (5) minutes.** The backups are **replicated to a geo-redundant data center** to ensure availability of the backups in the event of disaster.

Episerver saves **backup copies for thirty-five (35) days**.

Assets (media and files) are not backed up as a part of this process because Azure Blob Storage is disaster resilient. Assets are replicated both within the data center and to a geo-redundant location. However, this does not account for user error if an editor mistakenly deletes an asset.

Episerver 419



Module G – Optimizing, Securing, and Deploying – DXC Service

Failover replication process

While failed-over, logging is not available because blobs are readonly in the failover region.

Failover is an optional, additional cost, component.
https://world.episerver.com/digital-experience-cloud-service/failover/

Service Bus does not support active geo-replication. Be aware of this if you use Episerver Events API.

Episerver 420

Failover considerations

The main consideration is to ensure your site supports read-only mode:

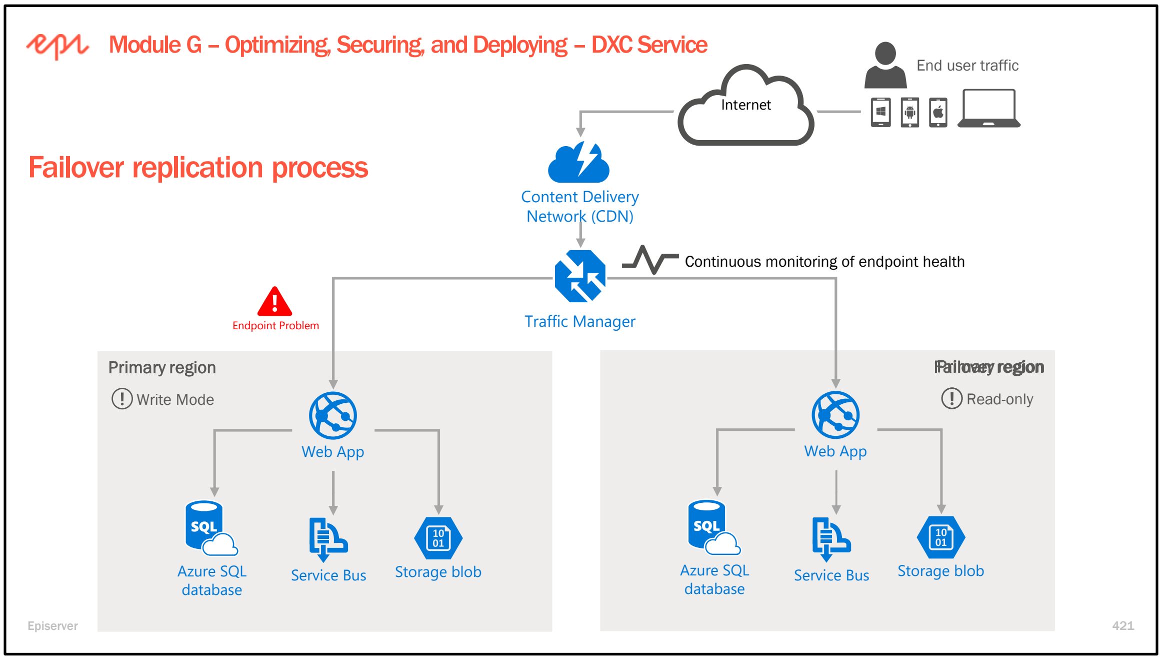
CMS and Commerce versions on your site must support read-only mode (CMS 9.7.0 and Commerce 9.9.0 or higher). Add-ons on the site must also support read-only mode.

Ensure that you configure warnings in your solution to handle read-only mode, for example by using application state. For database transactions features, such as saving a posted form, or storage transaction features like image resizing, these features must be aware that the application is in read-only mode, to not throw write exceptions.

Optionally, you can configure if you want to display an information message to end-users on the failover site when in read-only mode during a failure. Set the episerver:ReadOnlyInfoUrl appSetting to override the default of ~/Util/ReadOnly.aspx:

```
<appSettings>
  <add key="episerver:ReadOnlyInfoUrl" value="~/OurCustomReadOnlyPage.html" />
```

https://world.episerver.com/documentation/developer-guides/CMS/Deployment/database-mode/

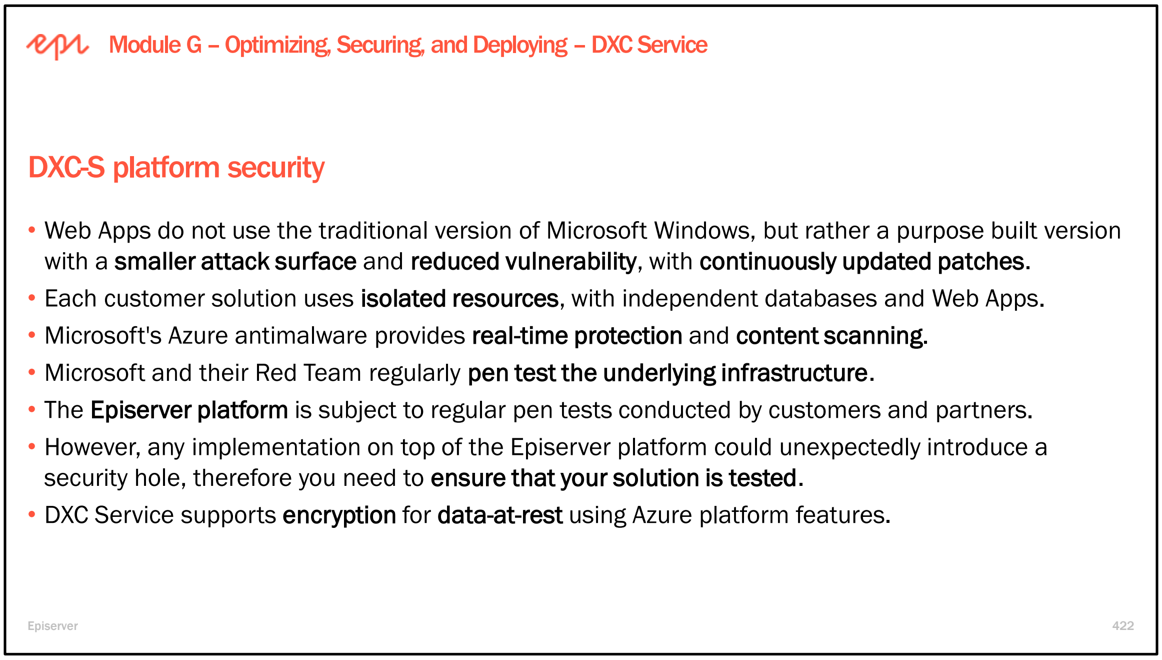


Module G – Optimizing, Securing, and Deploying – DXC Service

Failover replication process

Episerver

421



Module G – Optimizing, Securing, and Deploying – DXC Service

DXC-S platform security

- Web Apps do not use the traditional version of Microsoft Windows, but rather a purpose built version with a **smaller attack surface** and **reduced vulnerability**, with **continuously updated patches.**
- Each customer solution uses **isolated resources**, with independent databases and Web Apps.
- Microsoft's Azure antimalware provides **real-time protection** and **content scanning**.
- Microsoft and their Red Team regularly **pen test the underlying infrastructure.**
- The **Episerver platform** is subject to regular pen tests conducted by customers and partners.
- However, any implementation on top of the Episerver platform could unexpectedly introduce a security hole, therefore you need to **ensure that your solution is tested.**
- DXC Service supports **encryption** for **data-at-rest** using Azure platform features.

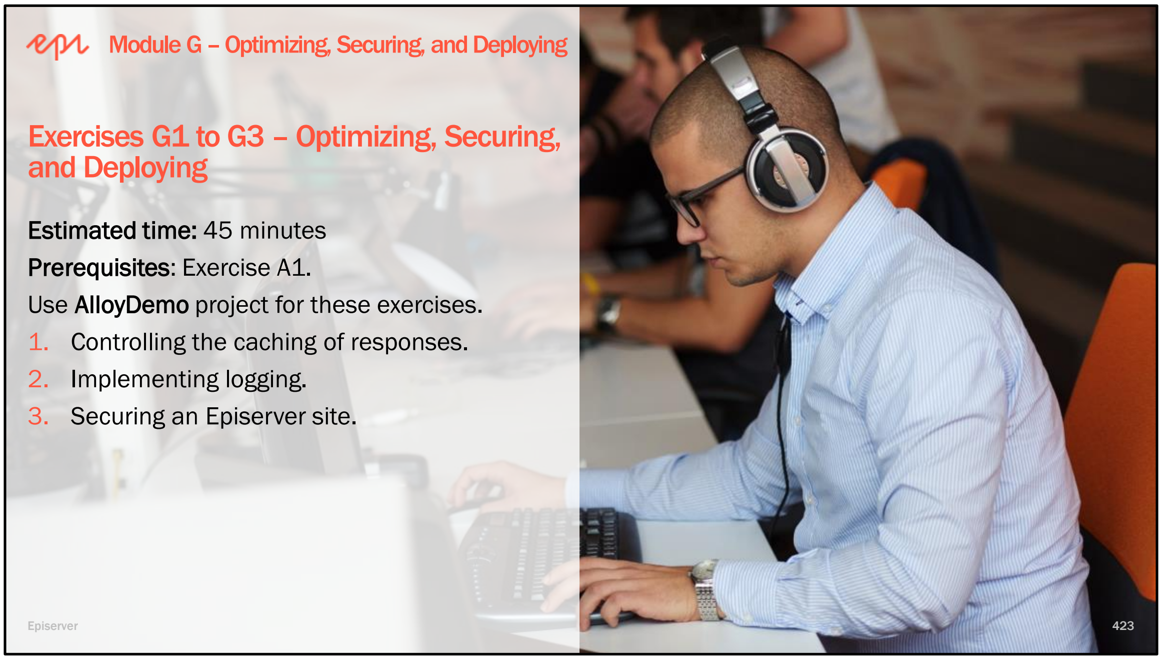
Episerver 422

WAF protects against Distributed Denial of Service (DDoS) attacks

DDoS attacks are common and complex, and traditional on premise solutions cannot handle these.

DXC-S WAF offers **advanced protection** at the network edge through its CDN provider including UDP and ICMP protocols, DNS amplification, Layer 7 and 3/4, SYN/ACK, and SMURF (refer to information on the net for this terminology).

DXC-S WAF supports **blocking traffic by country** and we will enable this for a customer on request.
Microsoft Azure also protects against attacks generated from outside and inside the platform.



Module G – Optimizing, Securing, and Deploying

Exercises G1 to G3 – Optimizing, Securing, and Deploying

Estimated time: 45 minutes

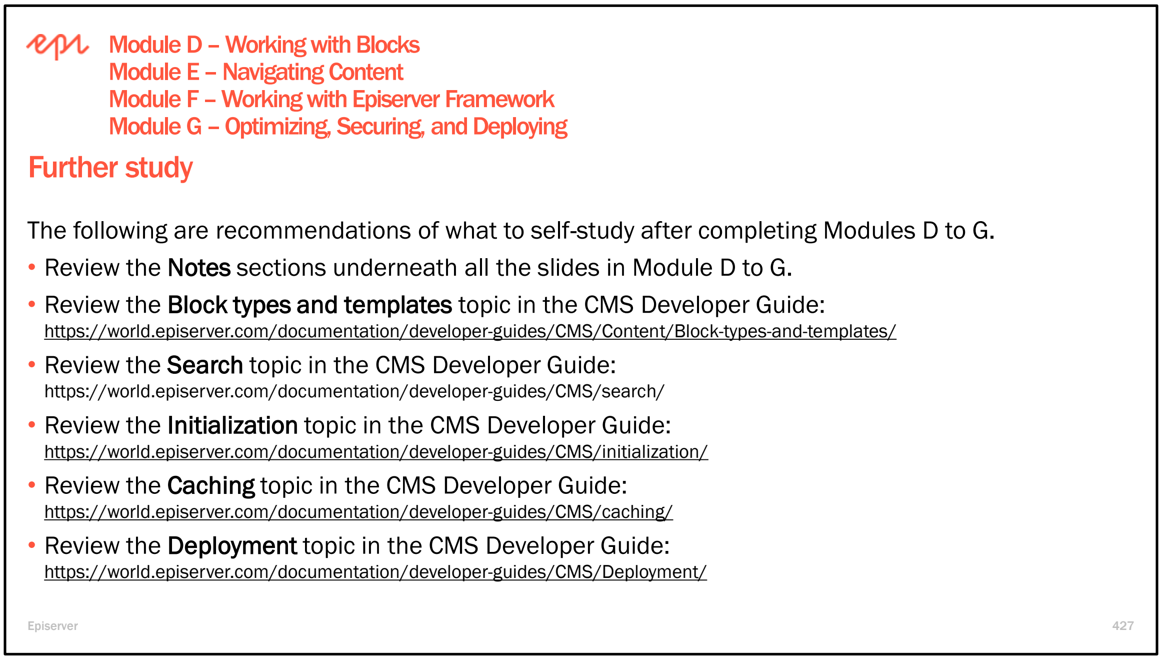
Prerequisites: Exercise A1.

Use **AlloyDemo** project for these exercises.

1. Controlling the caching of responses.
2. Implementing logging.
3. Securing an Episerver site.

Episerver

423



Module D – Working with Blocks
Module E – Navigating Content
Module F – Working with Episerver Framework
Module G – Optimizing, Securing, and Deploying

Further study

The following are recommendations of what to self-study after completing Modules D to G.

- Review the **Notes** sections underneath all the slides in Module D to G.
- Review the **Block types and templates** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/Content/Block-types-and-templates/
- Review the **Search** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/search/
- Review the **Initialization** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/initialization/
- Review the **Caching** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/caching/
- Review the **Deployment** topic in the CMS Developer Guide:
  https://world.episerver.com/documentation/developer-guides/CMS/Deployment/

Episerver 427

Episerver CMS – Development Fundamentals

Course Summary

Episerver

428

Certificate of Completion

Mark Price

Course date:

Course: Episerver CMS - Development Fundamentals :: eLearning

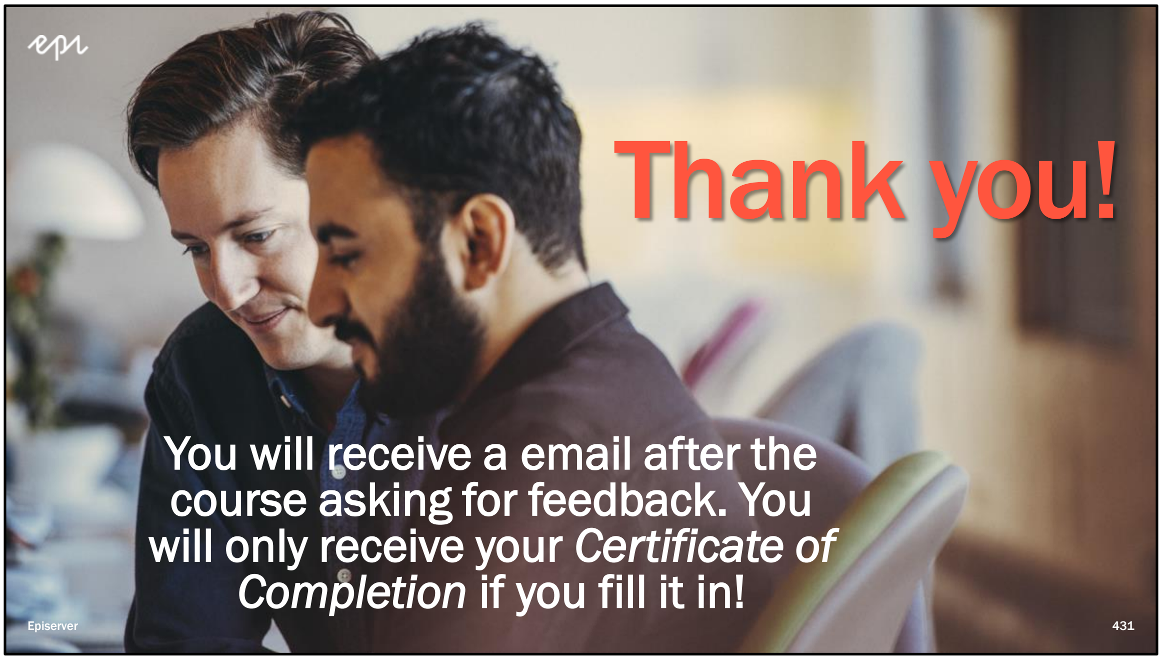
Episerver Education Services provides training courses on how to use Episerver solutions in the most efficient way possible. Participating in a training course gives insight into key knowledge areas and best practices in the Episerver platform. This certifies that the participant has successfully completed this course.

[illegible]

Annika Renestam
VP Global Education

Episerver

430



Thank you!

You will receive a email after the course asking for feedback. You will only receive your *Certificate of Completion* if you fill it in!

Episerver

431