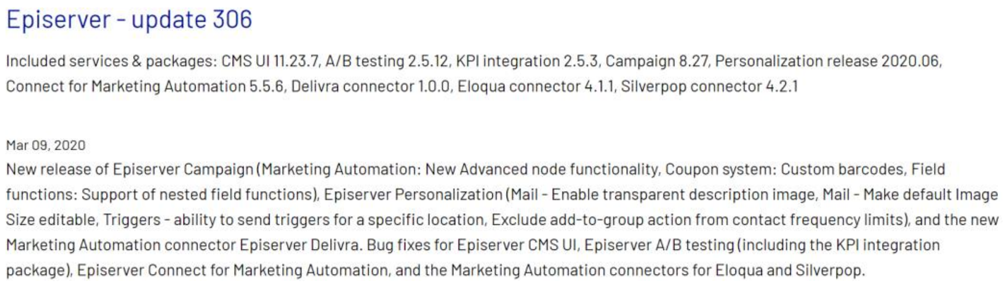
Customizing and Extending Episerver Content Cloud

Spring 2020

Formerly *Episerver CMS – Advanced Development*

Episerver

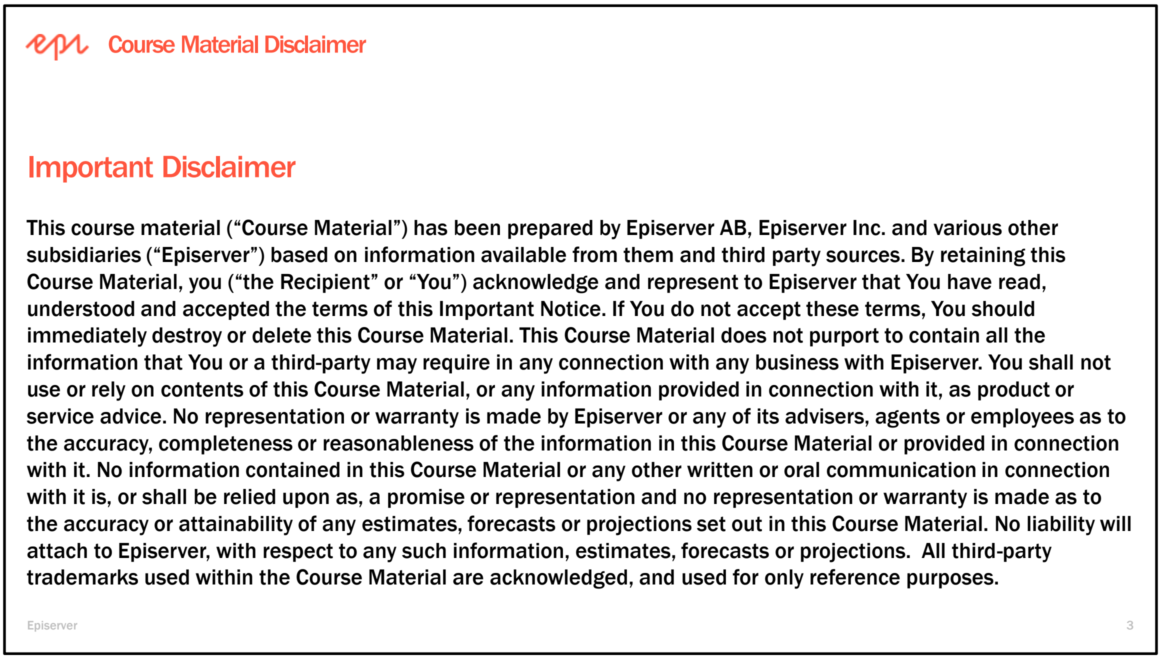
Course title: *Customizing and Extending Episerver Content Cloud* Course code: 170-3030
Course version: 20.03, March 10, 2020 Product version: Update 306, March 9, 2020
Episerver CMS Visual Studio Extension version: 11.6.0.421 (includes `EPiServer.CMS` 11.12.0)
Episerver packages: `EPiServer.CMS` 11.14.2, `EPiServer.CMS.UI` 11.23.7, `EPiServer.Forms` 4.27.1, `EPiServer.Search` 9.0.3, `EPiServer.Find` 13.2.5, `EPiServer.Marketing.Testing` 2.5.12
https://world.episerver.com/releases/



Episerver - update 306

Included services & packages: CMS UI 11.23.7, A/B testing 2.5.12, KPI integration 2.5.3, Campaign 8.27, Personalization release 2020.06, Connect for Marketing Automation 5.5.6, Delivra connector 1.0.0, Eloqua connector 4.1.1, Silverpop connector 4.2.1

Mar 09, 2020
New release of Episerver Campaign (Marketing Automation: New Advanced node functionality, Coupon system: Custom barcodes, Field functions: Support of nested field functions), Episerver Personalization (Mail - Enable transparent description image, Mail - Make default Image Size editable, Triggers - ability to send triggers for a specific location, Exclude add-to-group action from contact frequency limits), and the new Marketing Automation connector Episerver Delivra. Bug fixes for Episerver CMS UI, Episerver A/B testing (including the KPI integration package), Episerver Connect for Marketing Automation, and the Marketing Automation connectors for Eloqua and Silverpop.



Course Material Disclaimer

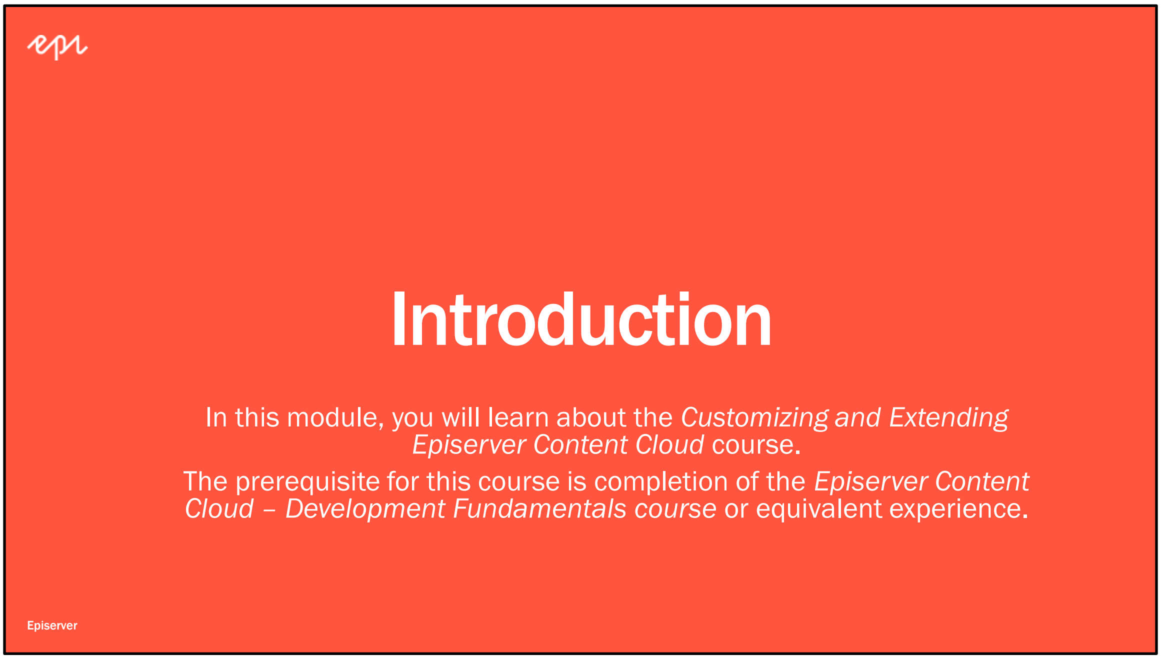
Important Disclaimer

This course material ("Course Material") has been prepared by Episerver AB, Episerver Inc. and various other subsidiaries ("Episerver") based on information available from them and third party sources. By retaining this Course Material, you ("the Recipient" or "You") acknowledge and represent to Episerver that You have read, understood and accepted the terms of this Important Notice. If You do not accept these terms, You should immediately destroy or delete this Course Material. This Course Material does not purport to contain all the information that You or a third-party may require in any connection with any business with Episerver. You shall not use or rely on contents of this Course Material, or any information provided in connection with it, as product or service advice. No representation or warranty is made by Episerver or any of its advisers, agents or employees as to the accuracy, completeness or reasonableness of the information in this Course Material or provided in connection with it. No information contained in this Course Material or any other written or oral communication in connection with it is, or shall be relied upon as, a promise or representation and no representation or warranty is made as to the accuracy or attainability of any estimates, forecasts or projections set out in this Course Material. No liability will attach to Episerver, with respect to any such information, estimates, forecasts or projections. All third-party trademarks used within the Course Material are acknowledged, and used for only reference purposes.

Episerver 3

Episerver does not accept responsibility or liability for any loss or damage suffered or incurred by You or any other person or entity however caused (including, without limitation, negligence) relating in any way to this Course Material including, without limitation, the information contained in or provided in connection with it, any errors or omissions from it however caused (including without limitation, where caused by third parties), lack of accuracy, completeness, currency or reliability or You, or any other person or entity, placing any reliance on this Course Material, its accuracy, completeness, currency or reliability. Any liability of Episerver (including advisers, agents and employees) to You or to any other person or entity arising out of this Course Material including any corresponding provision of any territory legislation, or any applicable law is, to the maximum extent permitted by law, expressly disclaimed and excluded.

You agree not to reproduce, print, re-transmit, copy, distribute, publish or sell any of the contents of this Course Material without the prior written consent of Episerver. Reproduction of part or all of the contents of this Course Material in any form is expressly prohibited and may not be recopied and/or shared with a third party. The permission to recopy by an individual does not allow for incorporation of material or any part of it in any work or publication, whether in hard copy, electronic, or any other form.



Introduction

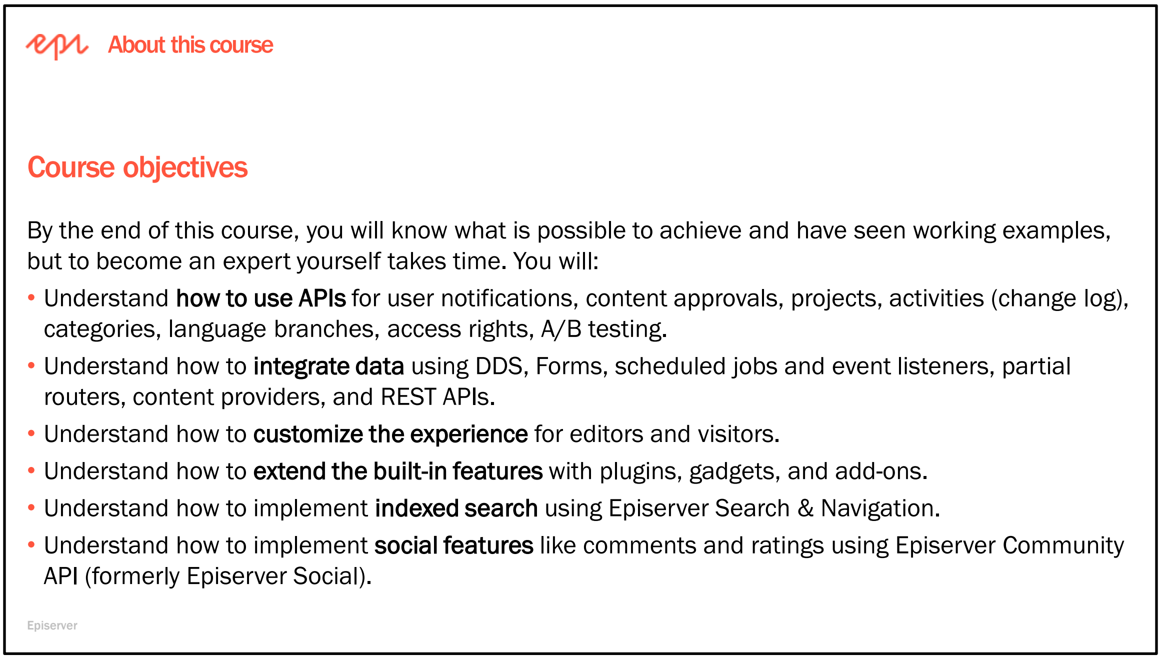
In this module, you will learn about the *Customizing and Extending Episerver Content Cloud* course.

The prerequisite for this course is completion of the *Episerver Content Cloud – Development Fundamentals course* or equivalent experience.

Episerver

Prerequisites are:

- Experience with Microsoft Visual Studio 2015 or later, ASP.NET MVC, and web front end technologies.
- Experience with Episerver CMS equivalent to our *Episerver Content Cloud – Development Fundamentals* training course.



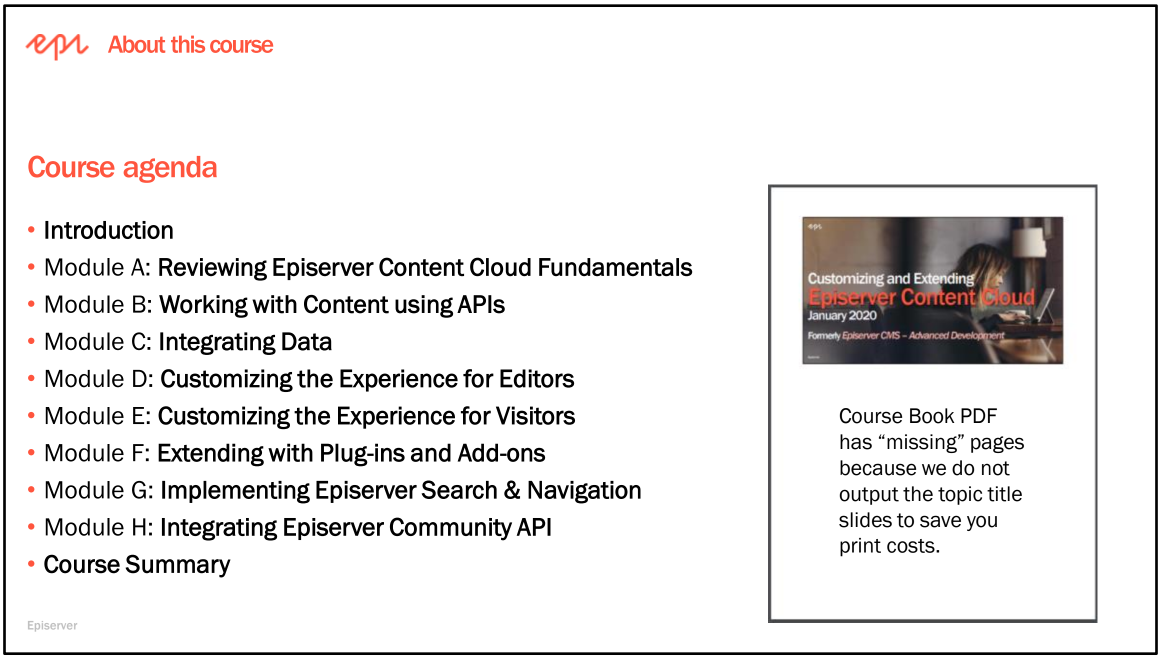
About this course

Course objectives

By the end of this course, you will know what is possible to achieve and have seen working examples, but to become an expert yourself takes time. You will:

- Understand **how to use APIs** for user notifications, content approvals, projects, activities (change log), categories, language branches, access rights, A/B testing.
- Understand how to **integrate data** using DDS, Forms, scheduled jobs and event listeners, partial routers, content providers, and REST APIs.
- Understand how to **customize the experience** for editors and visitors.
- Understand how to **extend the built-in features** with plugins, gadgets, and add-ons.
- Understand how to implement **indexed search** using Episerver Search & Navigation.
- Understand how to implement **social features** like comments and ratings using Episerver Community API (formerly Episerver Social).

Episerver



About this course

Course agenda

- Introduction
- Module A: **Reviewing Episerver Content Cloud Fundamentals**
- Module B: **Working with Content using APIs**
- Module C: **Integrating Data**
- Module D: **Customizing the Experience for Editors**
- Module E: **Customizing the Experience for Visitors**
- Module F: **Extending with Plug-ins and Add-ons**
- Module G: **Implementing Episerver Search & Navigation**
- Module H: **Integrating Episerver Community API**
- Course Summary

Customizing and Extending
Episerver Content Cloud
January 2020
Formerly Episerver CMS – Advanced Development

Course Book PDF has “missing” pages because we do not output the topic title slides to save you print costs.

Episerver

Module A: Reviewing Episerver Content Cloud Fundamentals
In this module, you will review topics you should already know.

Module B: Working with Content using APIs
In this module, you will learn about some advanced APIs including working with Content Approvals, Projects, and Notifications.

Module C: Integrating Data
In this module, you will learn about various technologies and techniques for integrating non-content data, including gathering visitor data with Forms and integrating external data systems with partial routers and Service API.

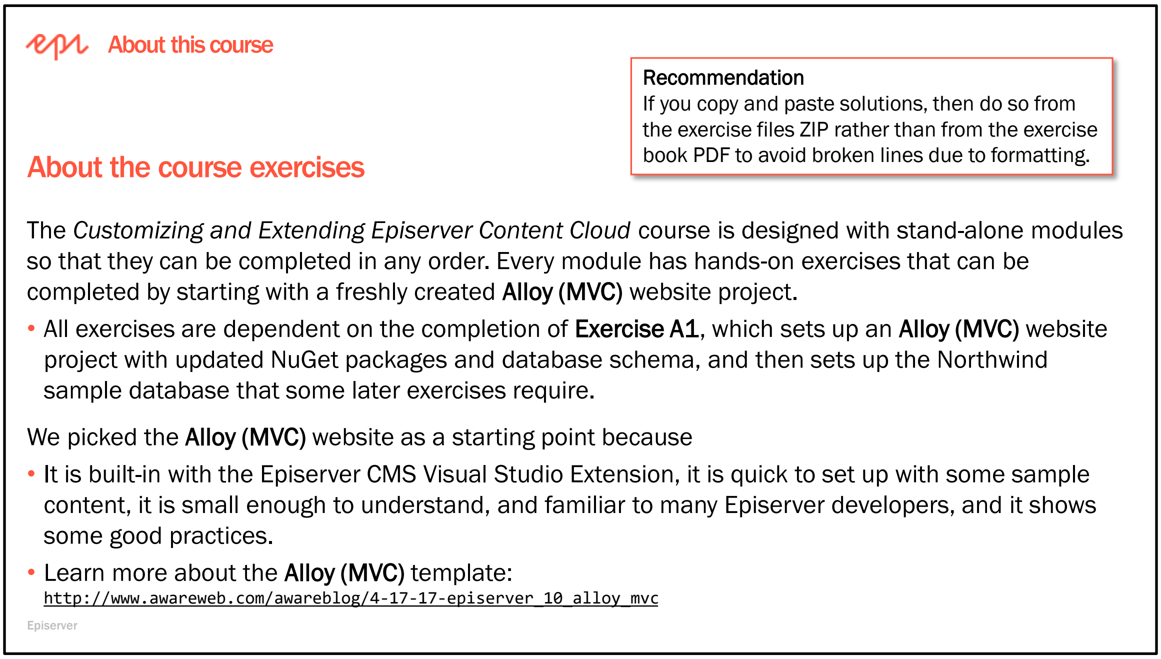
Module D: Customizing the Experience for Editors
In this module, you will learn how to customize the editors experience when setting content properties.

Module E: Customizing the Experience for Visitors
In this module, you will learn how to take control of the visitors experience with custom rendering, personalization with visitor groups, and advanced customization of Episerver Search,.

Module F: Extending with Plug-ins and Add-ons
In this module, you will learn how to extend Episerver with custom plug-ins, gadgets, and add-ons.

Module G: Implementing Episerver Search & Navigation (formerly Find)
In this module, you will learn how to integrate Episerver Content Cloud with Episerver Search & Navigation (formerly Find) to implement advanced search capabilities.

Module H: Integrating Episerver Community API
In this module, you will learn how to integrate Episerver CMS with Episerver **Community API** to implement advanced features like comments, ratings, and managing groups.



About this course

About the course exercises

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.

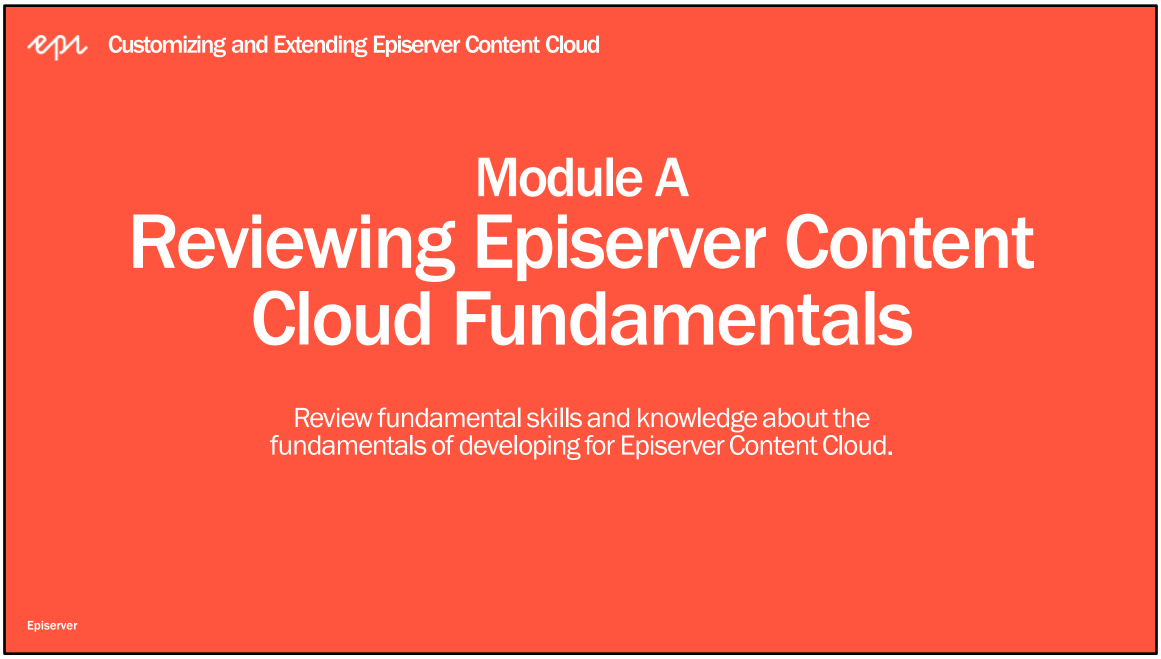
The *Customizing and Extending Episerver Content Cloud* course is designed with stand-alone modules so that they can be completed in any order. Every module has hands-on exercises that can be completed by starting with a freshly created **Alloy (MVC)** website project.

- All exercises are dependent on the completion of **Exercise A1**, which sets up an **Alloy (MVC)** website project with updated NuGet packages and database schema, and then sets up the Northwind sample database that some later exercises require.

We picked the **Alloy (MVC)** website as a starting point because

- It is built-in with the Episerver CMS Visual Studio Extension, it is quick to set up with some sample content, it is small enough to understand, and familiar to many Episerver developers, and it shows some good practices.
- Learn more about the **Alloy (MVC)** template: `http://www.awareweb.com/awareblog/4-17-17-episerver_10_alloy_mvc`

Episerver



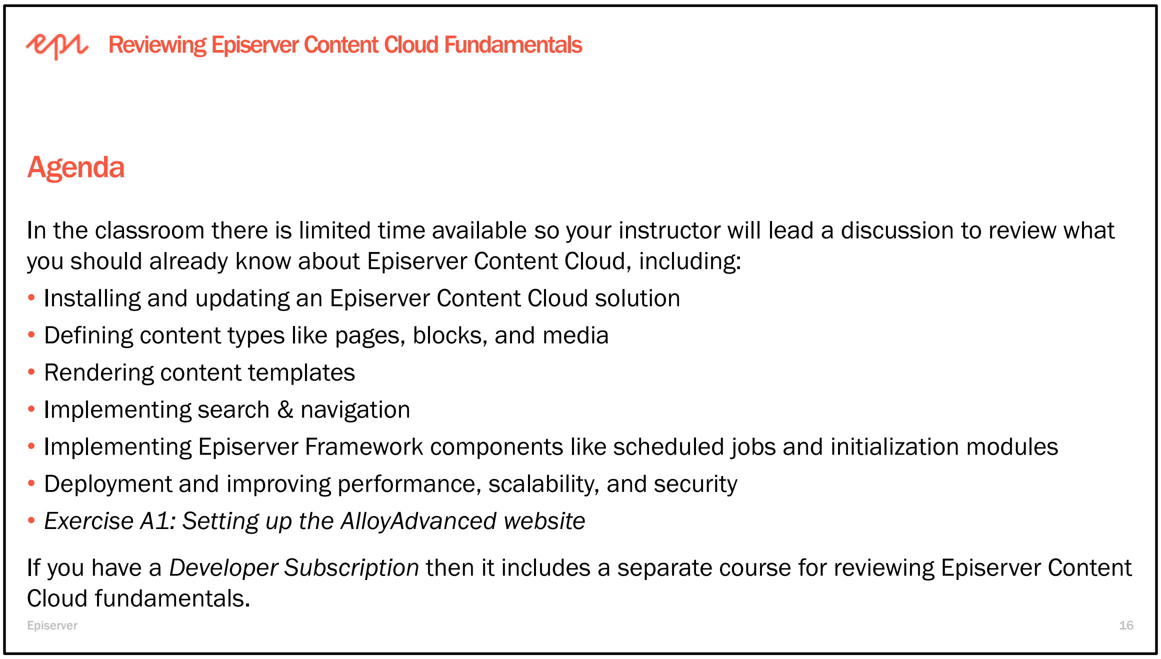
Customizing and Extending Episerver Content Cloud

Module A

Reviewing Episerver Content Cloud Fundamentals

Review fundamental skills and knowledge about the fundamentals of developing for Episerver Content Cloud.

Episerver



Reviewing Episerver Content Cloud Fundamentals

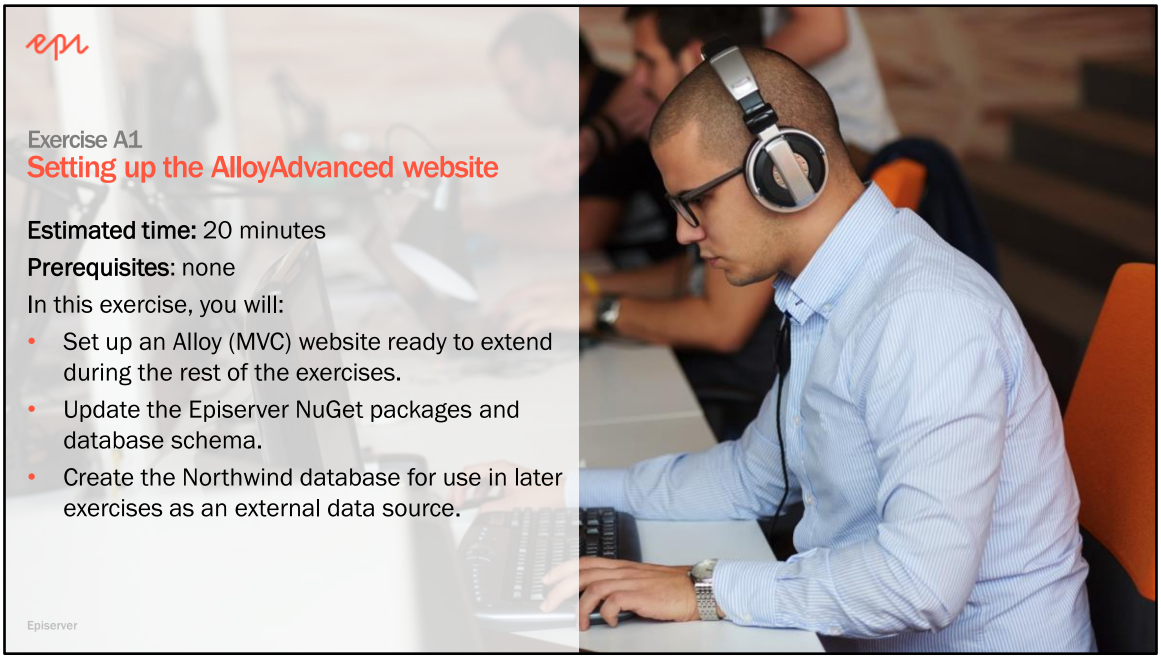
Agenda

In the classroom there is limited time available so your instructor will lead a discussion to review what you should already know about Episerver Content Cloud, including:

- Installing and updating an Episerver Content Cloud solution
- Defining content types like pages, blocks, and media
- Rendering content templates
- Implementing search & navigation
- Implementing Episerver Framework components like scheduled jobs and initialization modules
- Deployment and improving performance, scalability, and security
- *Exercise A1: Setting up the AlloyAdvanced website*

If you have a *Developer Subscription* then it includes a separate course for reviewing Episerver Content Cloud fundamentals.

Episerver 16



Exercise A1

Setting up the AlloyAdvanced website

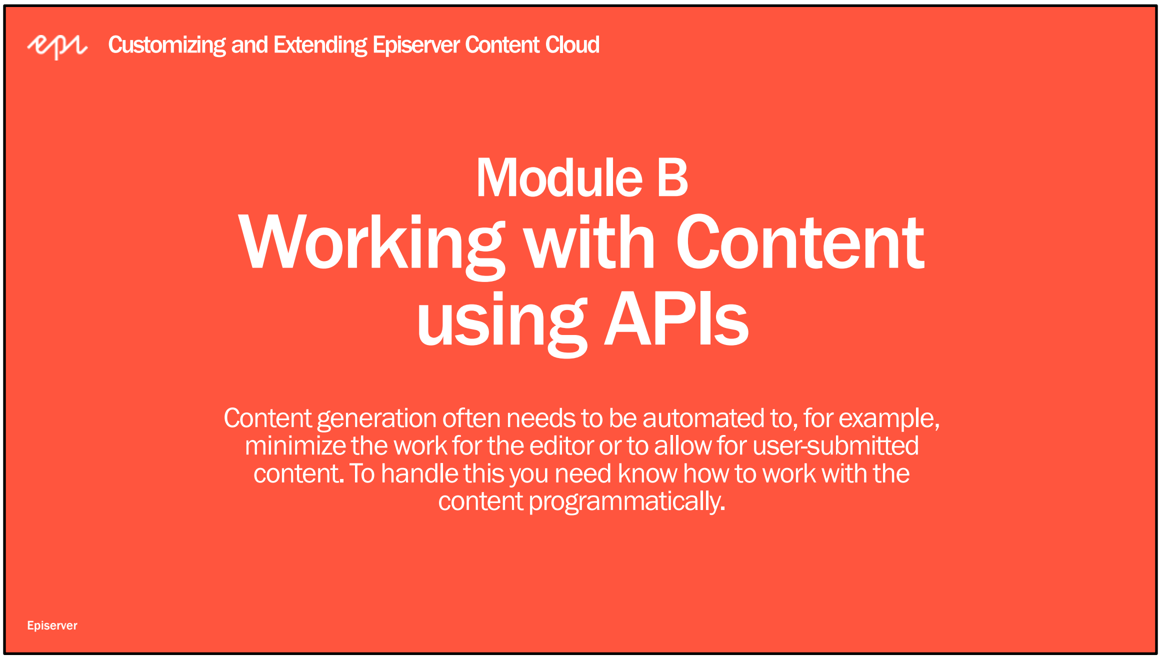
Estimated time: 20 minutes

Prerequisites: none

In this exercise, you will:

- Set up an Alloy (MVC) website ready to extend during the rest of the exercises.
- Update the Episerver NuGet packages and database schema.
- Create the Northwind database for use in later exercises as an external data source.

Episerver



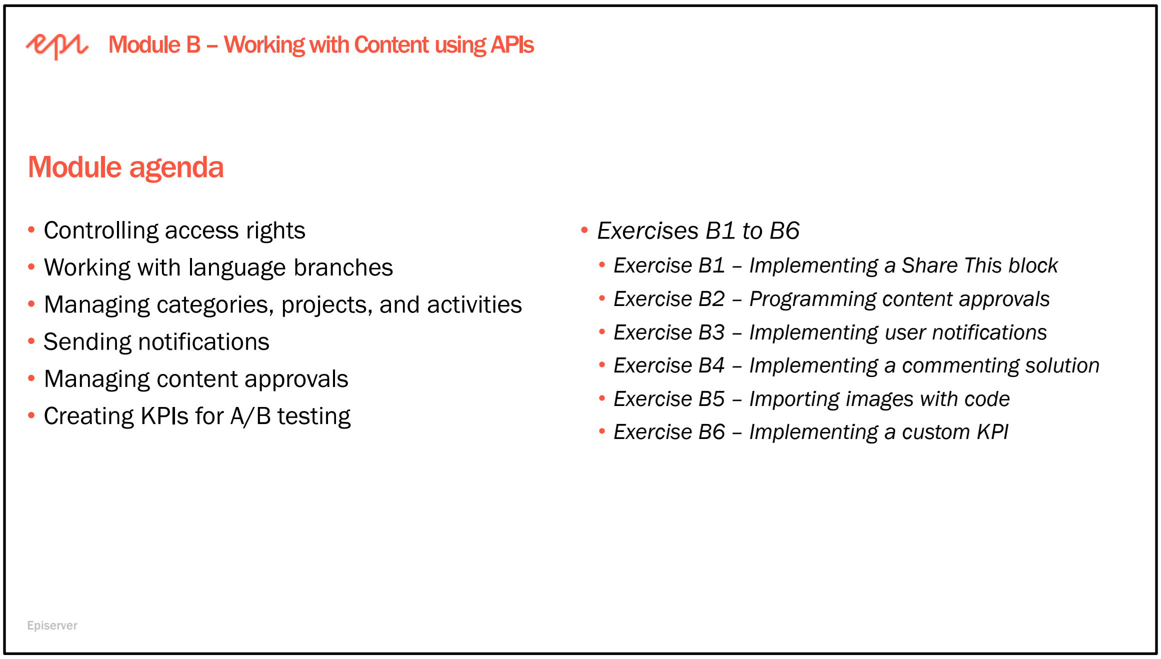
Customizing and Extending Episerver Content Cloud

Module B

Working with Content using APIs

Content generation often needs to be automated to, for example, minimize the work for the editor or to allow for user-submitted content. To handle this you need know how to work with the content programmatically.

Episerver



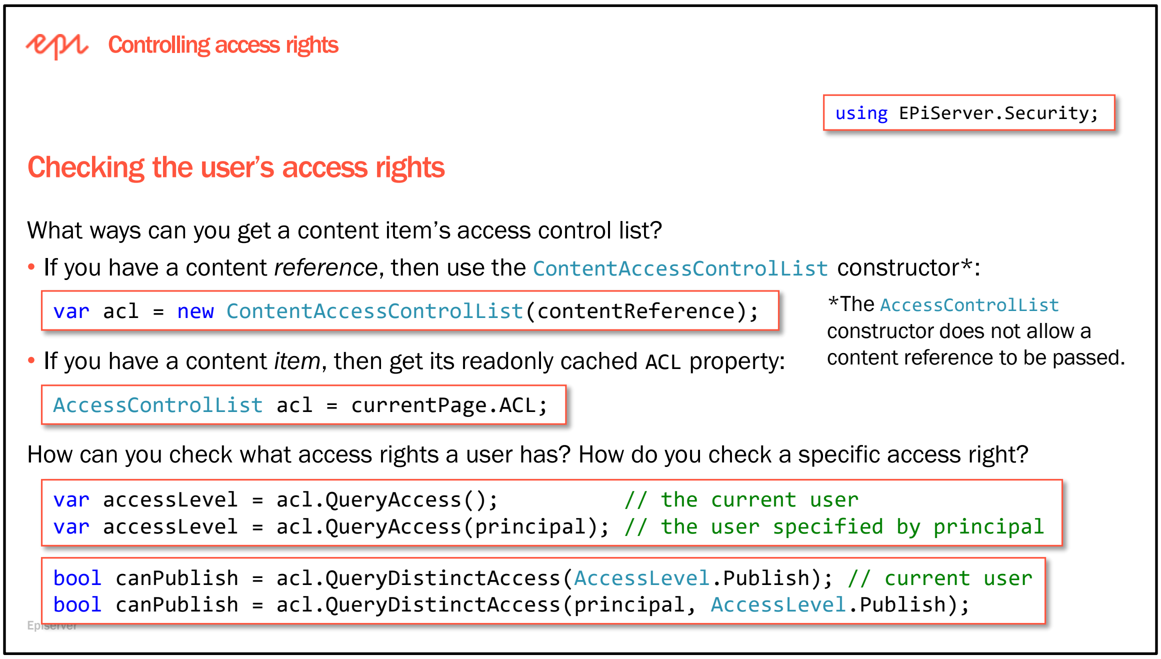
Module B – Working with Content using APIs

Module agenda

- Controlling access rights
- Working with language branches
- Managing categories, projects, and activities
- Sending notifications
- Managing content approvals
- Creating KPIs for A/B testing

- *Exercises B1 to B6*
  - *Exercise B1 – Implementing a Share This block*
  - *Exercise B2 – Programming content approvals*
  - *Exercise B3 – Implementing user notifications*
  - *Exercise B4 – Implementing a commenting solution*
  - *Exercise B5 – Importing images with code*
  - *Exercise B6 – Implementing a custom KPI*

Episerver



Controlling access rights

```
using EPiServer.Security;
```

Checking the user's access rights

What ways can you get a content item's access control list?

- If you have a content *reference*, then use the `ContentAccessControlList` constructor*:

```
var acl = new ContentAccessControlList(contentReference);
```

*The `AccessControlList` constructor does not allow a content reference to be passed.

- If you have a content *item*, then get its readonly cached `ACL` property:

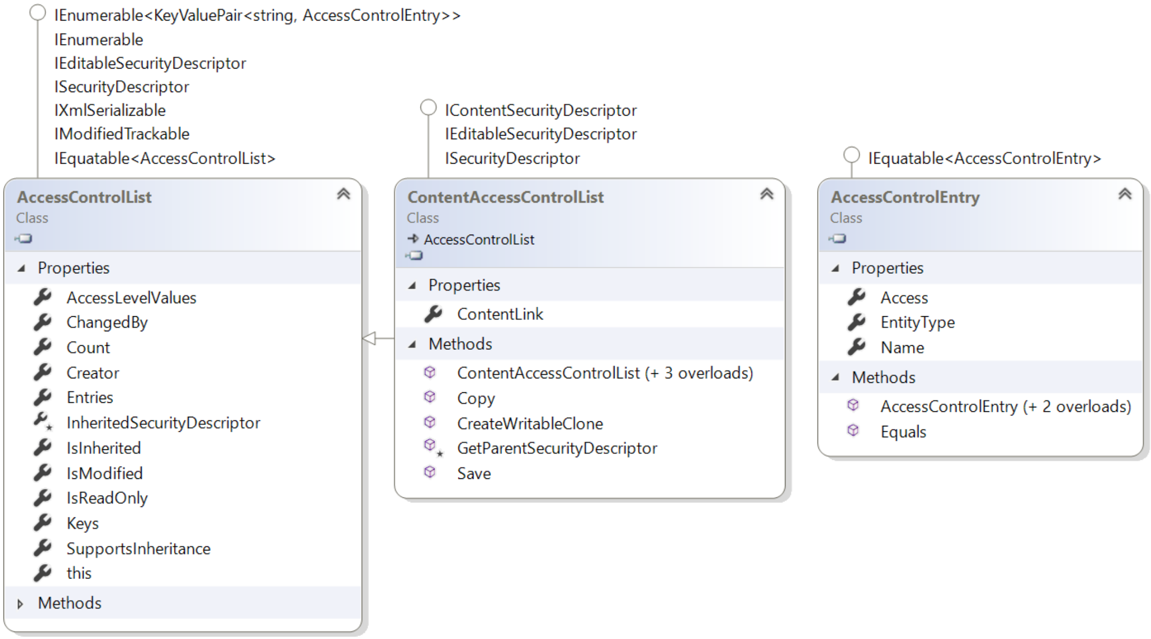
```
AccessControlList acl = currentPage.ACL;
```

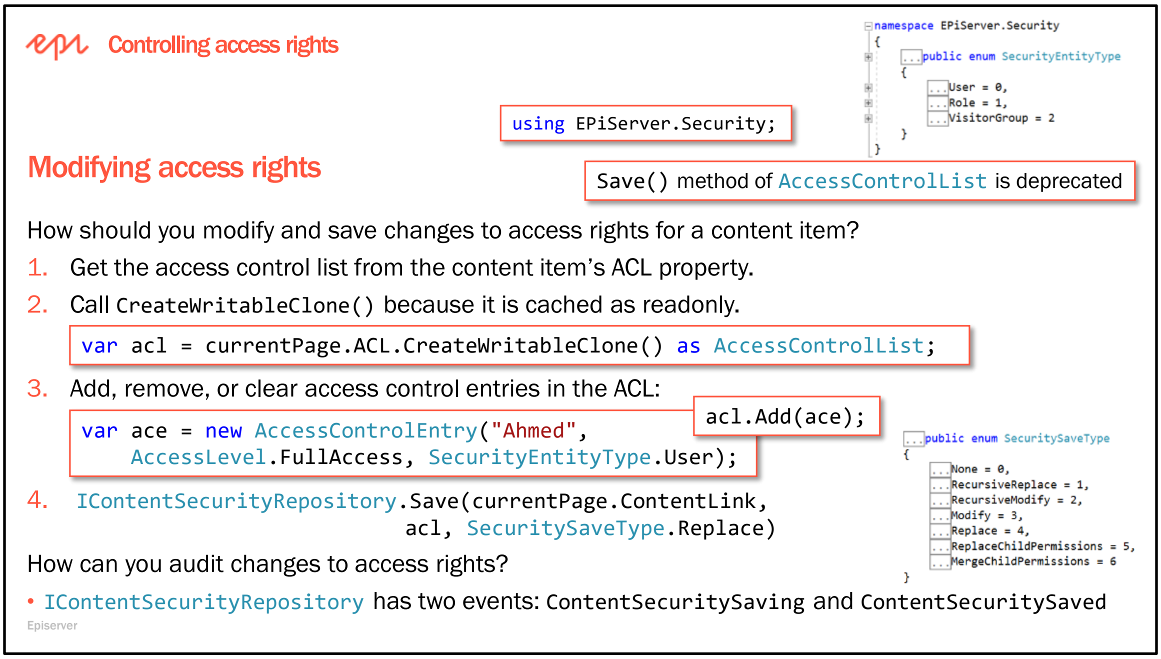
How can you check what access rights a user has? How do you check a specific access right?

```
var accessLevel = acl.QueryAccess();          // the current user
var accessLevel = acl.QueryAccess(principal); // the user specified by principal
```

```
bool canPublish = acl.QueryDistinctAccess(AccessLevel.Publish); // current user
bool canPublish = acl.QueryDistinctAccess(principal, AccessLevel.Publish);
```

Episerver





Controlling access rights

Modifying access rights

```
namespace EPiServer.Security
{
    public enum SecurityEntityType
    {
        User = 0,
        Role = 1,
        VisitorGroup = 2
    }
}
```

```
using EPiServer.Security;
```

`Save()` method of `AccessControlList` is deprecated

How should you modify and save changes to access rights for a content item?

1. Get the access control list from the content item's ACL property.
2. Call `CreateWritableClone()` because it is cached as readonly.

```
var acl = currentPage.ACL.CreateWritableClone() as AccessControlList;
```

3. Add, remove, or clear access control entries in the ACL:

```
var ace = new AccessControlEntry("Ahmed",
    AccessLevel.FullAccess, SecurityEntityType.User);
```

```
acl.Add(ace);
```

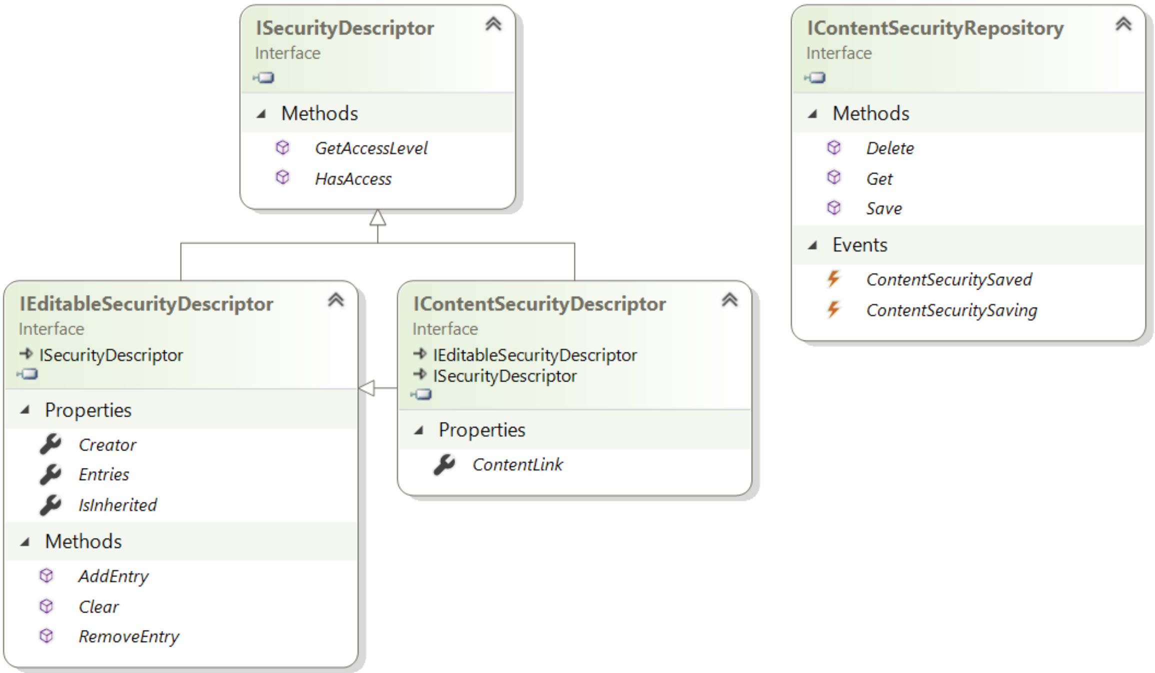
```
public enum SecuritySaveType
{
    None = 0,
    RecursiveReplace = 1,
    RecursiveModify = 2,
    Modify = 3,
    Replace = 4,
    ReplaceChildPermissions = 5,
    MergeChildPermissions = 6
}
```

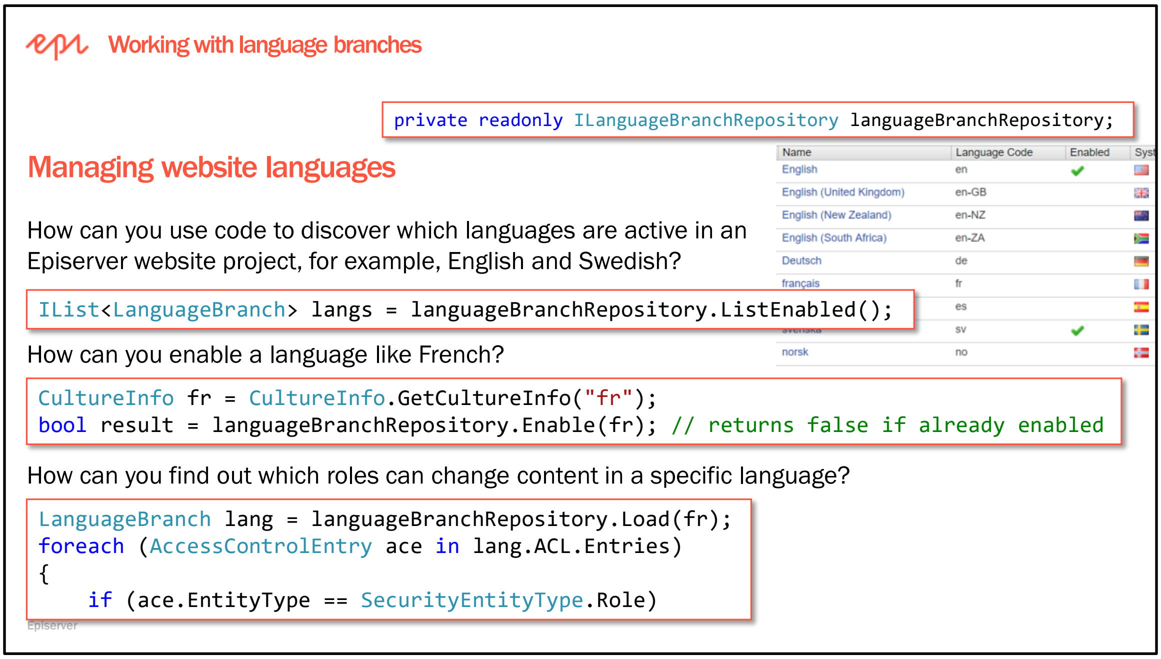
4.
```
IContentSecurityRepository.Save(currentPage.ContentLink,
                                acl, SecuritySaveType.Replace)
```

How can you audit changes to access rights?

- `IContentSecurityRepository` has two events: `ContentSecuritySaving` and `ContentSecuritySaved`

Episerver





Working with language branches

```
private readonly ILanguageBranchRepository languageBranchRepository;
```

Managing website languages

Name	Language Code	Enabled	Syst
English	en	✓	🇺🇸
English (United Kingdom)	en-GB		🇬🇧
English (New Zealand)	en-NZ		🇳🇿
English (South Africa)	en-ZA		🇿🇦
Deutsch	de		🇩🇪
français	fr		🇫🇷
	es		🇪🇸
svenska	sv	✓	🇸🇪
norsk	no		🇳🇴

How can you use code to discover which languages are active in an Episerver website project, for example, English and Swedish?

```
IList<LanguageBranch> langs = languageBranchRepository.ListEnabled();
```

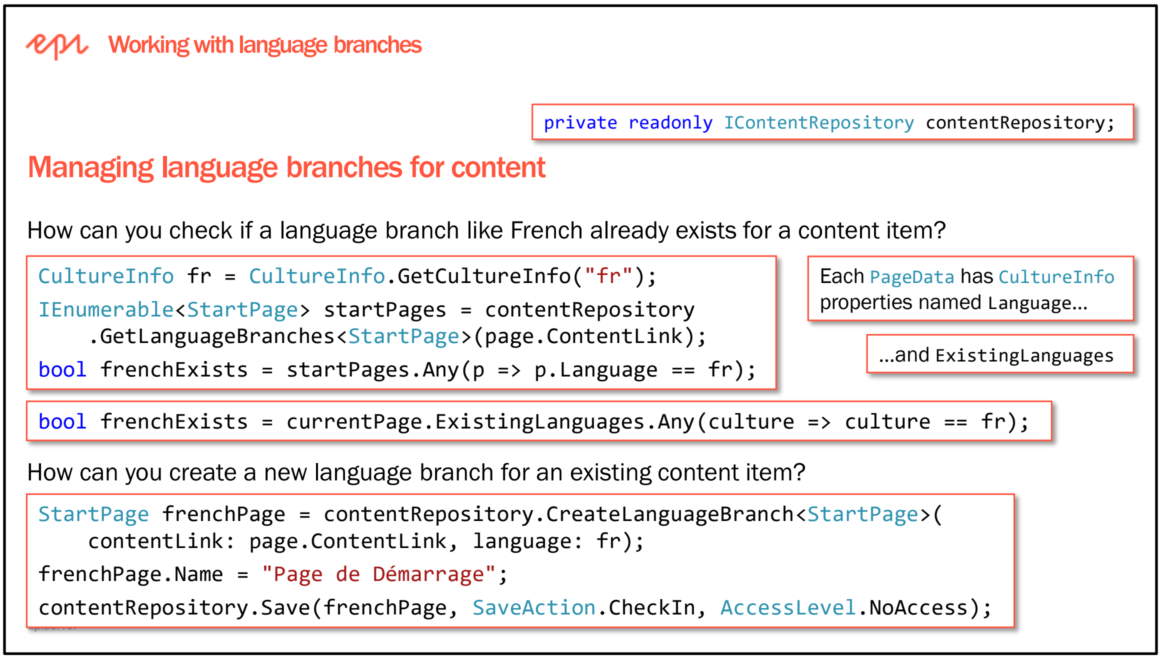
How can you enable a language like French?

```
CultureInfo fr = CultureInfo.GetCultureInfo("fr");
bool result = languageBranchRepository.Enable(fr); // returns false if already enabled
```

How can you find out which roles can change content in a specific language?

```
LanguageBranch lang = languageBranchRepository.Load(fr);
foreach (AccessControlEntry ace in lang.ACL.Entries)
{
    if (ace.EntityType == SecurityEntityType.Role)
```

Episerver



Working with language branches

```
private readonly IContentRepository contentRepository;
```

Managing language branches for content

How can you check if a language branch like French already exists for a content item?

```
CultureInfo fr = CultureInfo.GetCultureInfo("fr");
IEnumerable<StartPage> startPages = contentRepository
    .GetLanguageBranches<StartPage>(page.ContentLink);
bool frenchExists = startPages.Any(p => p.Language == fr);
```

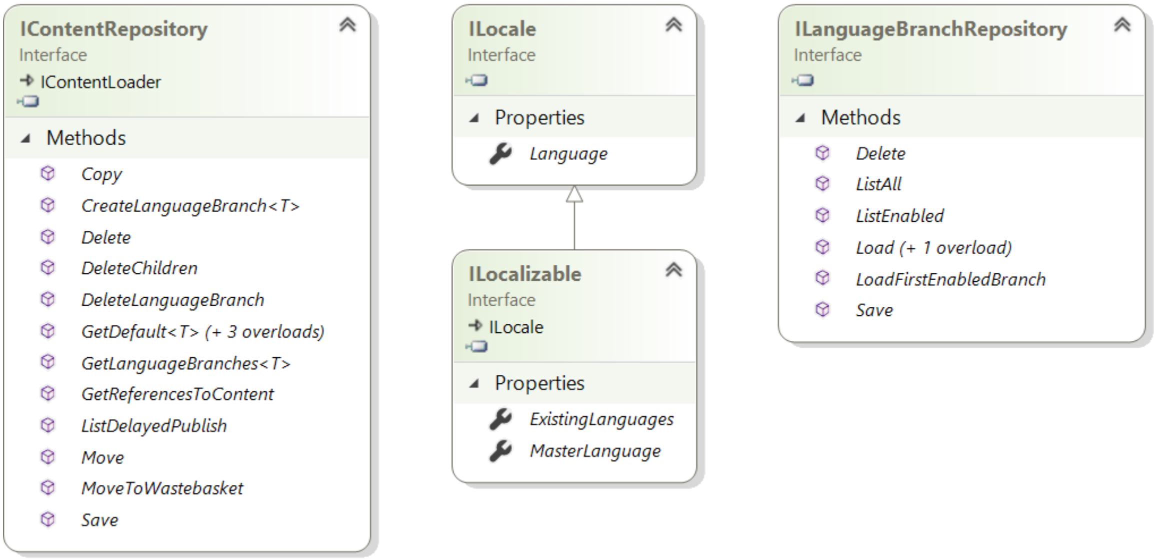
Each PageData has CultureInfo properties named Language…

…and ExistingLanguages

```
bool frenchExists = currentPage.ExistingLanguages.Any(culture => culture == fr);
```

How can you create a new language branch for an existing content item?

```
StartPage frenchPage = contentRepository.CreateLanguageBranch<StartPage>(
    contentLink: page.ContentLink, language: fr);
frenchPage.Name = "Page de Démarrage";
contentRepository.Save(frenchPage, SaveAction.CheckIn, AccessLevel.NoAccess);
```



Managing categories, projects, and activities

Episerver CMS 8 or later

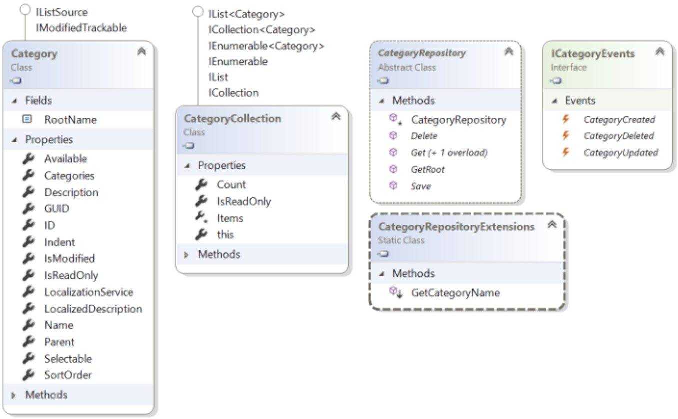
Programmatically working with categories

Use `CategoryRepository` to work with categories programmatically.

- `Get()`: by ID or name
- `GetRoot()`
- `Save()`
- `Delete()`

Use `ICategoryEvents` to log when categories are changed.

Use `Category` and its `Categories` property (has its child categories as a `CategoryCollection`) to navigate the hierarchy.



Episerver

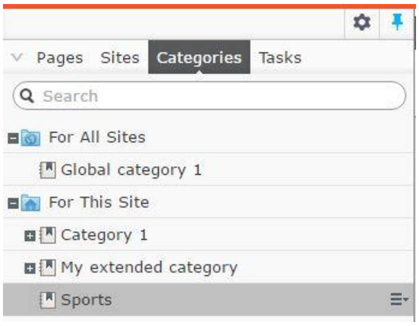
Alternatives to default Episerver categories

An alternative to Episerver's default category functionality, where categories are instead stored as localizable IContent: https://github.com/Geta/EpiCategories

Features

- Localization (no more language XML files)
- More user friendly edit UI
- Access rights support (some editors should perhaps have limited category access)
- Shared and site specific categories in multisite solutions
- Partial routing of category URL segments

```
Install-Package Geta.EpiCategories
```

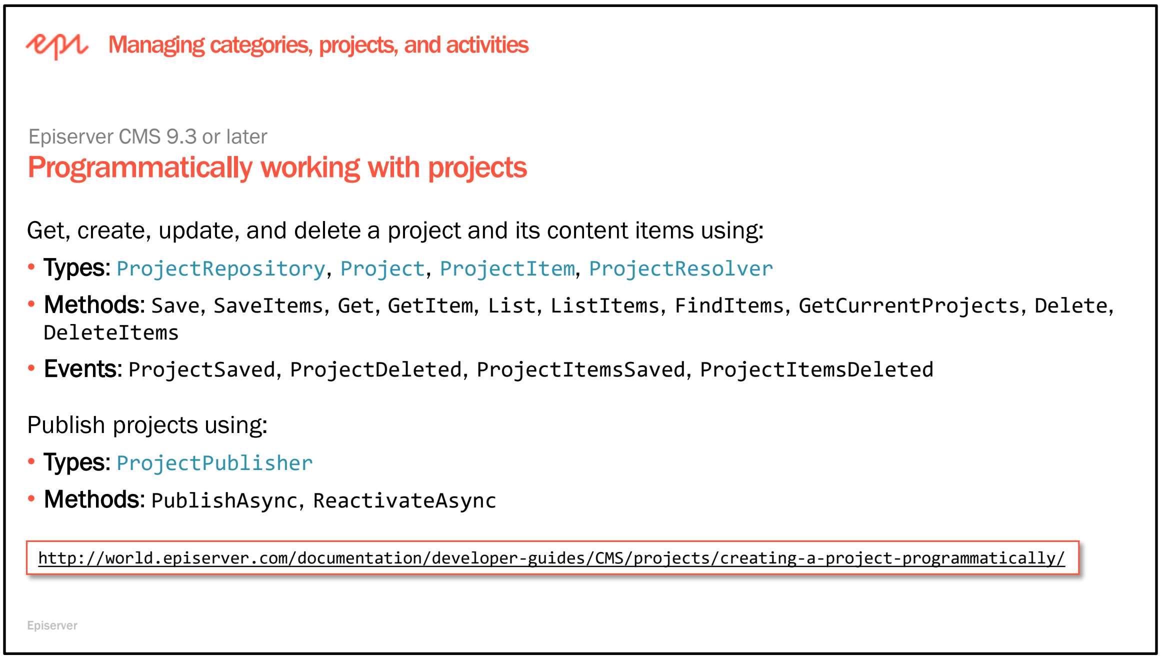


Geta Tags for EPiServer CMS

https://github.com/Geta/Tags

Relations for Episerver, connectable content for better navigation and great relevance

https://github.com/BVNetwork/Relations



Managing categories, projects, and activities

Episerver CMS 9.3 or later

Programmatically working with projects

Get, create, update, and delete a project and its content items using:

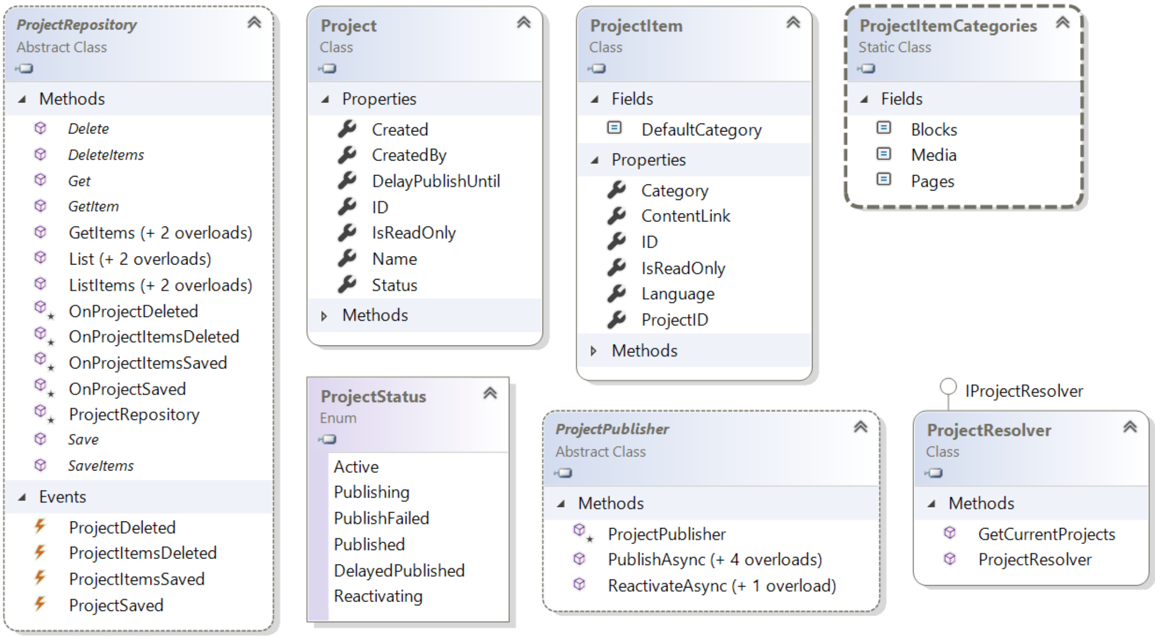
- **Types:** `ProjectRepository`, `Project`, `ProjectItem`, `ProjectResolver`
- **Methods:** `Save`, `SaveItems`, `Get`, `GetItem`, `List`, `ListItems`, `FindItems`, `GetCurrentProjects`, `Delete`, `DeleteItems`
- **Events:** `ProjectSaved`, `ProjectDeleted`, `ProjectItemsSaved`, `ProjectItemsDeleted`

Publish projects using:

- **Types:** `ProjectPublisher`
- **Methods:** `PublishAsync`, `ReactivateAsync`

http://world.episerver.com/documentation/developer-guides/CMS/projects/creating-a-project-programmatically/

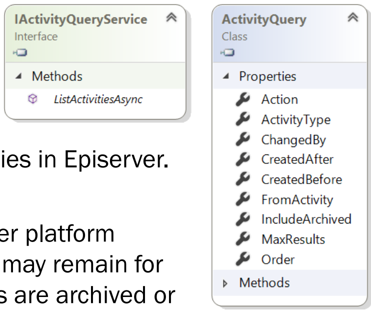
Episerver



Managing categories, projects, and activities

Episerver CMS 10.9 or later

Programmatically working with activities

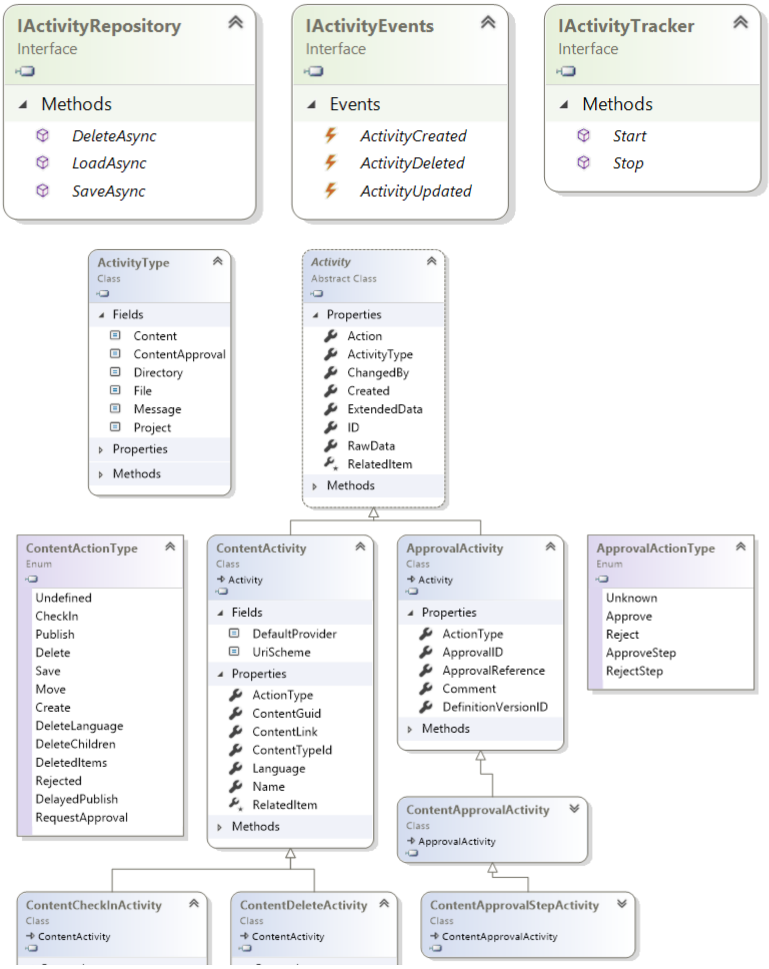


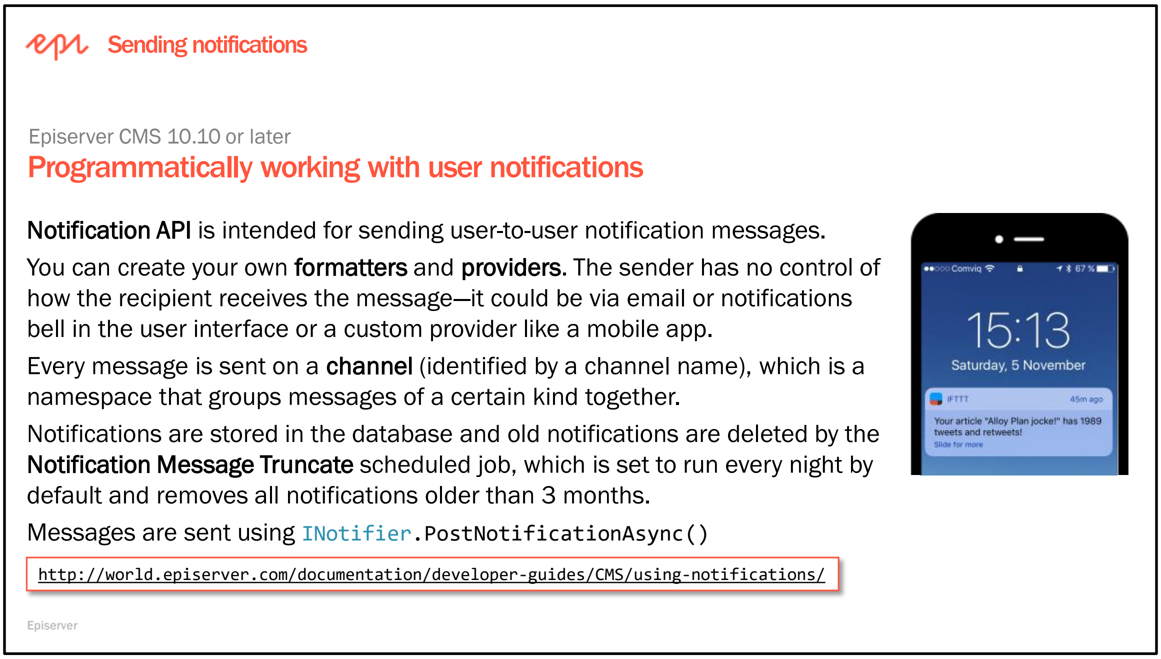
Change Log is a user interface for administrators to list recent activities in Episerver. All changes to content items are logged as an **activity**.

- **Retention:** All activities are stored at least one month unless another platform feature has a dependency to certain activities, in which cases they may remain for an additional period. Activities without any remaining dependencies are archived or deleted by a scheduled job. Archived activities are persisted for 12 months by default.
- **Activities API**: The classes and interfaces for the Activities API can be found in the `EPiServer.DataAbstraction.Activities` namespace. The Activities API supersedes the previous ChangeLog API that is now deprecated. Developers can execute queries to retrieve information from the Activities log.

Episerver

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





Sending notifications

Episerver CMS 10.10 or later

Programmatically working with user notifications

Notification API is intended for sending user-to-user notification messages.

You can create your own **formatters** and **providers.** The sender has no control of how the recipient receives the message—it could be via email or notifications bell in the user interface or a custom provider like a mobile app.

Every message is sent on a **channel** (identified by a channel name), which is a namespace that groups messages of a certain kind together.

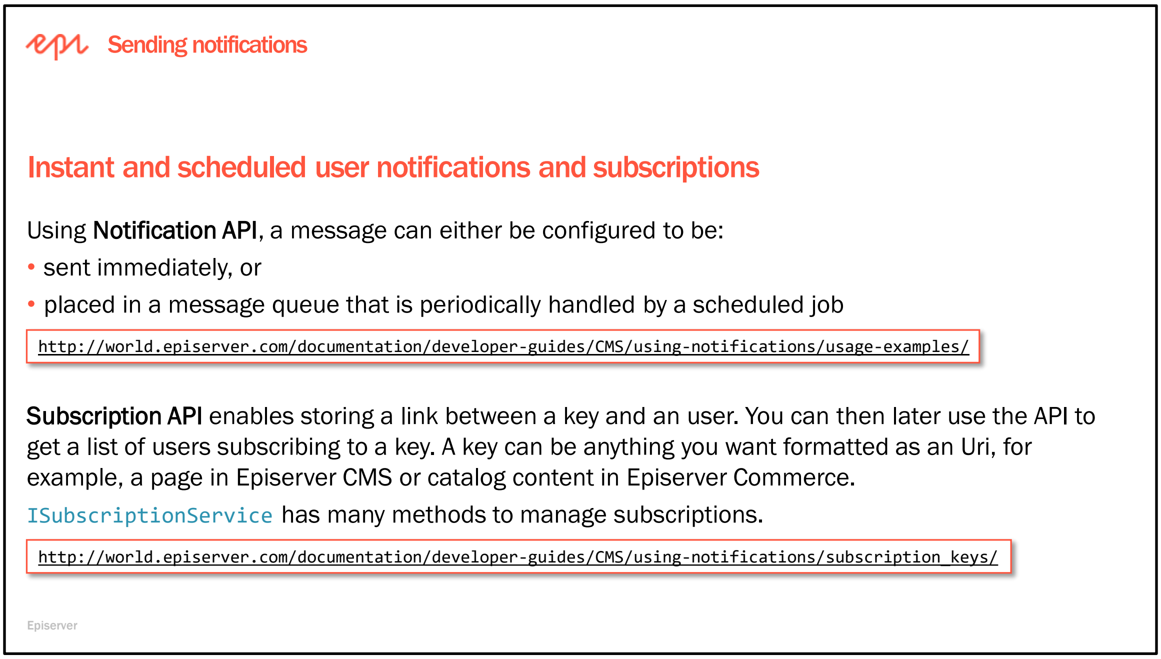
Notifications are stored in the database and old notifications are deleted by the **Notification Message Truncate** scheduled job, which is set to run every night by default and removes all notifications older than 3 months.

Messages are sent using `INotifier.PostNotificationAsync()`

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

Episerver

Comviq
67 %
15:13
Saturday, 5 November
IFTTT
45m ago
Your article "Alloy Plan jocke!" has 1989 tweets and retweets!
Slide for more



Sending notifications

Instant and scheduled user notifications and subscriptions

Using **Notification API**, a message can either be configured to be:

- sent immediately, or
- placed in a message queue that is periodically handled by a scheduled job

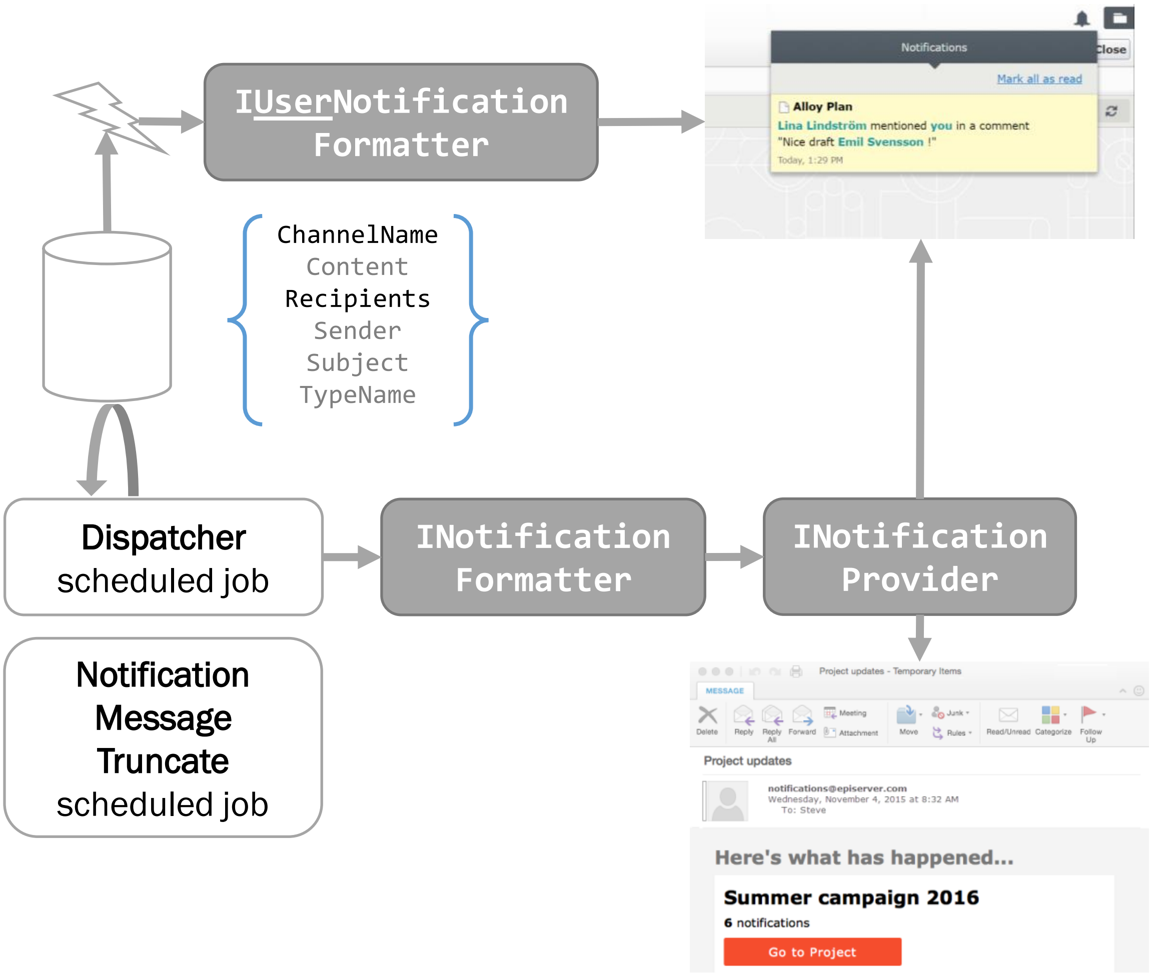
http://world.episerver.com/documentation/developer-guides/CMS/using-notifications/usage-examples/

Subscription API enables storing a link between a key and an user. You can then later use the API to get a list of users subscribing to a key. A key can be anything you want formatted as an Uri, for example, a page in Episerver CMS or catalog content in Episerver Commerce.

`ISubscriptionService` has many methods to manage subscriptions.

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

Episerver



Alloy Plan
Lina Lindström mentioned you in a comment
"Nice draft Emil Svensson !"
Today, 1:29 PM

Project updates - Temporary Items

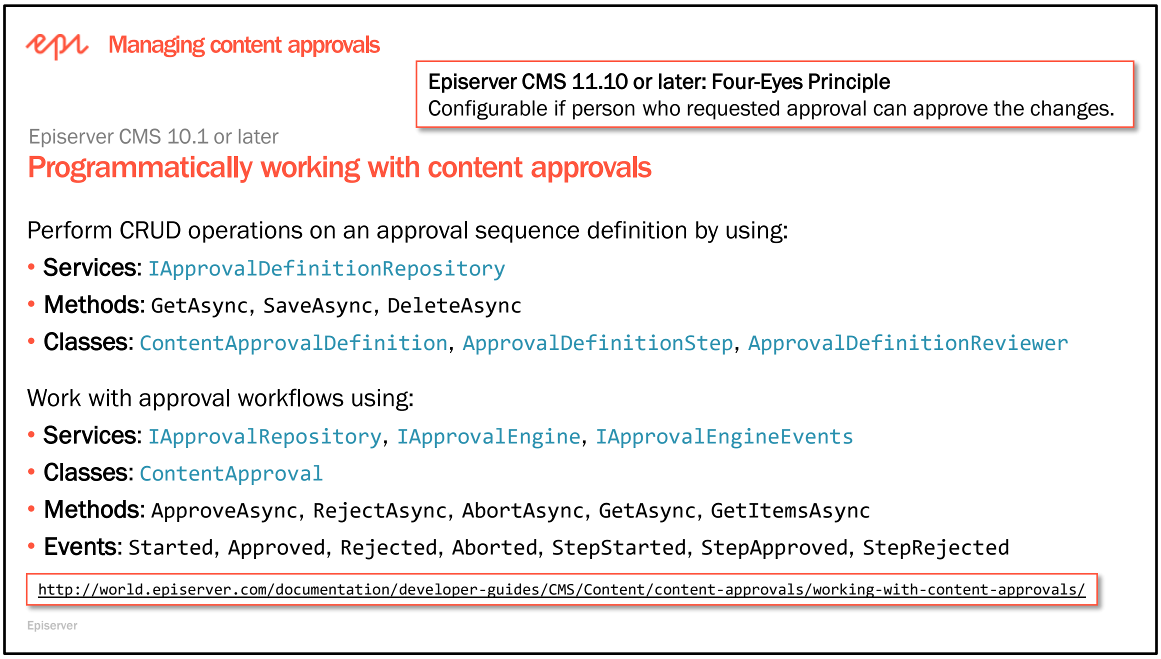
Project updates

notifications@episerver.com
Wednesday, November 4, 2015 at 8:32 AM
To: Steve

Here's what has happened...

Summer campaign 2016

6 notifications



Managing content approvals

Episerver CMS 11.10 or later: Four-Eyes Principle
Configurable if person who requested approval can approve the changes.

Episerver CMS 10.1 or later

Programmatically working with content approvals

Perform CRUD operations on an approval sequence definition by using:

- **Services:** `IApprovalDefinitionRepository`
- **Methods:** `GetAsync`, `SaveAsync`, `DeleteAsync`
- **Classes:** `ContentApprovalDefinition`, `ApprovalDefinitionStep`, `ApprovalDefinitionReviewer`

Work with approval workflows using:

- **Services:** `IApprovalRepository`, `IApprovalEngine`, `IApprovalEngineEvents`
- **Classes:** `ContentApproval`
- **Methods:** `ApproveAsync`, `RejectAsync`, `AbortAsync`, `GetAsync`, `GetItemsAsync`
- **Events:** `Started`, `Approved`, `Rejected`, `Aborted`, `StepStarted`, `StepApproved`, `StepRejected`

http://world.episerver.com/documentation/developer-guides/CMS/Content/content-approvals/working-with-content-approvals/

Episerver

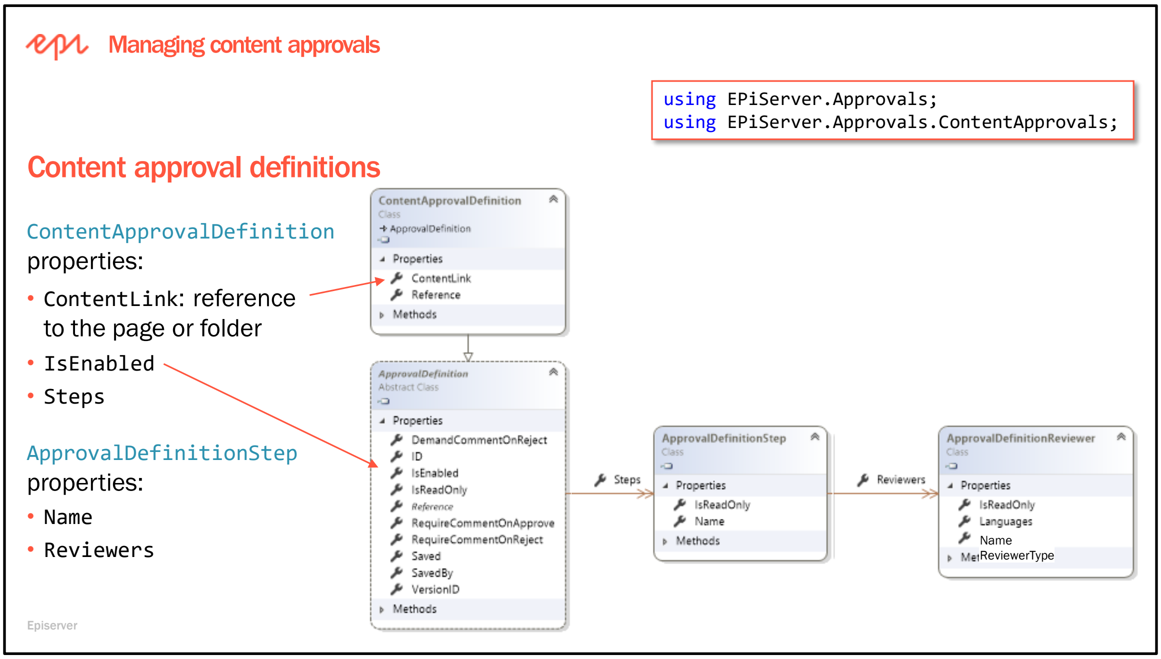
Content approvals is a way to make sure that content is reviewed and approved before it is published.
The reviewers are defined by an administrator in an approval sequence.
One or more appointed reviewers must then approve the content item before it can be published. To review content the user must have **Read** access right and at least one other access right, like **Create** or **Change** or **Delete**.
When an editor has finished working on a content item, the item is set to **Ready for Review**.

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 as a reviewer 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/



Managing content approvals

```
using EPiServer.Approvals;
using EPiServer.Approvals.ContentApprovals;
```

Content approval definitions

`ContentApprovalDefinition` properties:

- `ContentLink`: reference to the page or folder
- `IsEnabled`
- `Steps`

`ApprovalDefinitionStep` properties:

- `Name`
- `Reviewers`

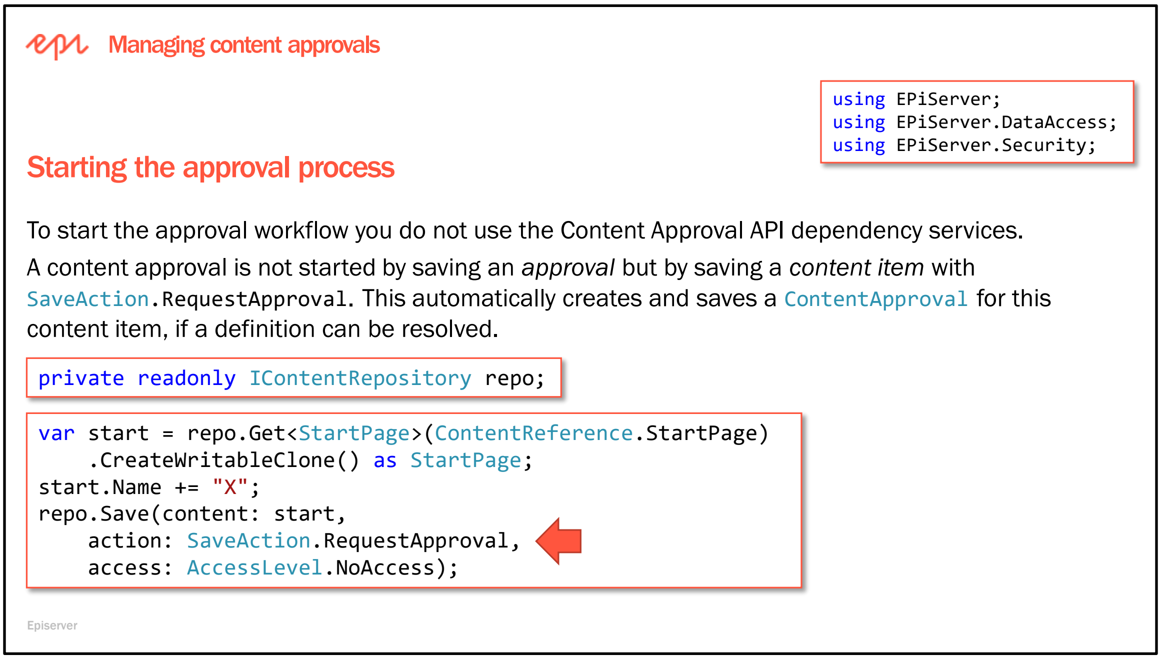
Episerver

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.



Managing content approvals

```
using EPiServer;
using EPiServer.DataAccess;
using EPiServer.Security;
```

Starting the approval process

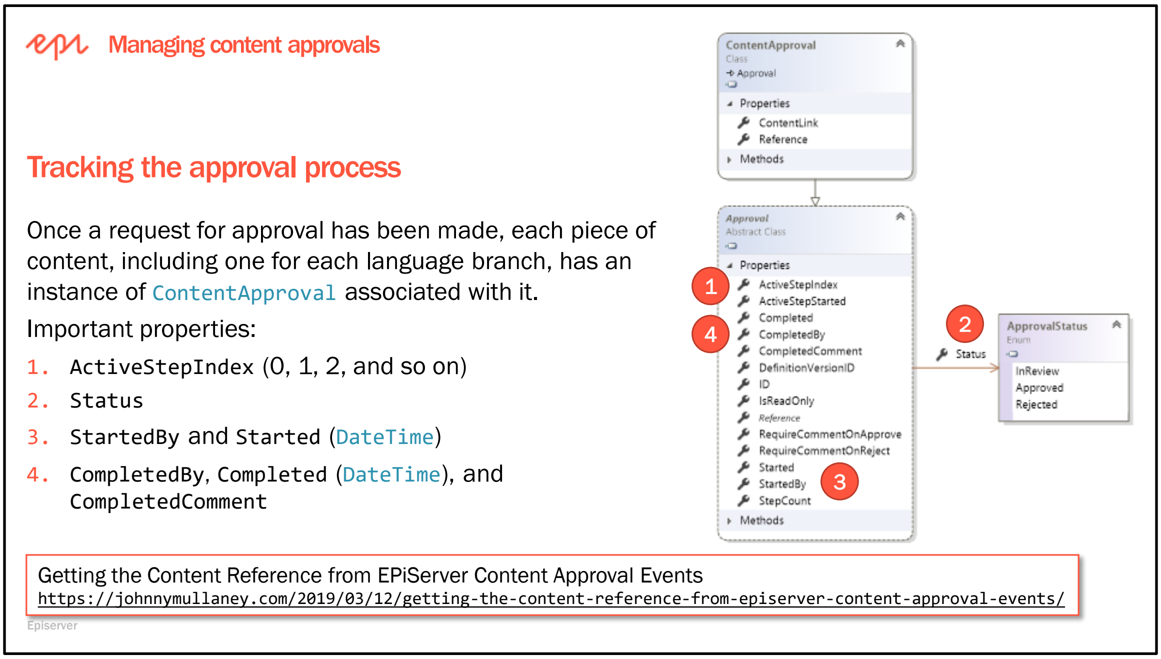
To start the approval workflow you do not use the Content Approval API dependency services.

A content approval is not started by saving an *approval* but by saving a *content item* with `SaveAction.RequestApproval`. This automatically creates and saves a `ContentApproval` for this content item, if a definition can be resolved.

```
private readonly IContentRepository repo;
```

```
var start = repo.Get<StartPage>(ContentReference.StartPage)
    .CreateWritableClone() as StartPage;
start.Name += "X";
repo.Save(content: start,
    action: SaveAction.RequestApproval,
    access: AccessLevel.NoAccess);
```

Episerver



Managing content approvals

Tracking the approval process

Once a request for approval has been made, each piece of content, including one for each language branch, has an instance of `ContentApproval` associated with it.

Important properties:

1. `ActiveStepIndex` (0, 1, 2, and so on)
2. `Status`
3. `StartedBy` and `Started` (`DateTime`)
4. `CompletedBy`, `Completed` (`DateTime`), and `CompletedComment`

Getting the Content Reference from EPiServer Content Approval Events
https://johnnymullaney.com/2019/03/12/getting-the-content-reference-from-episerver-content-approval-events/

Episerver

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.

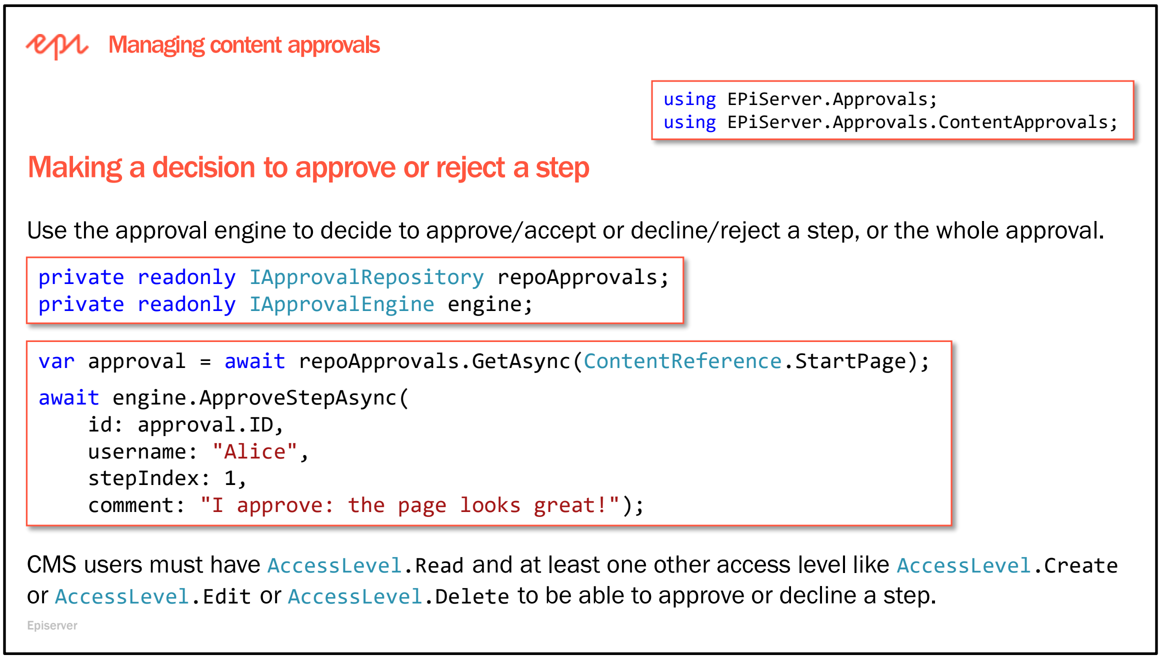
Administrators and the editor who started the approval sequence can cancel the approval sequence at any step.

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/



Managing content approvals

```
using EPiServer.Approvals;
using EPiServer.Approvals.ContentApprovals;
```

Making a decision to approve or reject a step

Use the approval engine to decide to approve/accept or decline/reject a step, or the whole approval.

```
private readonly IApprovalRepository repoApprovals;
private readonly IApprovalEngine engine;
```

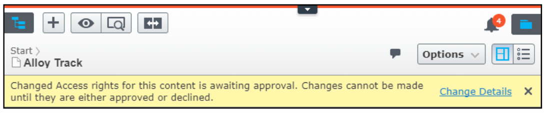
```
var approval = await repoApprovals.GetAsync(ContentReference.StartPage);
await engine.ApproveStepAsync(
    id: approval.ID,
    username: "Alice",
    stepIndex: 1,
    comment: "I approve: the page looks great!");
```

CMS users must have `AccessLevel.Read` and at least one other access level like `AccessLevel.Create` or `AccessLevel.Edit` or `AccessLevel.Delete` to be able to approve or decline a step.

Episerver

Managing content approvals

Change approvals



Start >
Alloy Track

Options

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

Change Details ✕

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 EPiServer.ChangeApproval -ProjectName AlloyAdvanced
```

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.

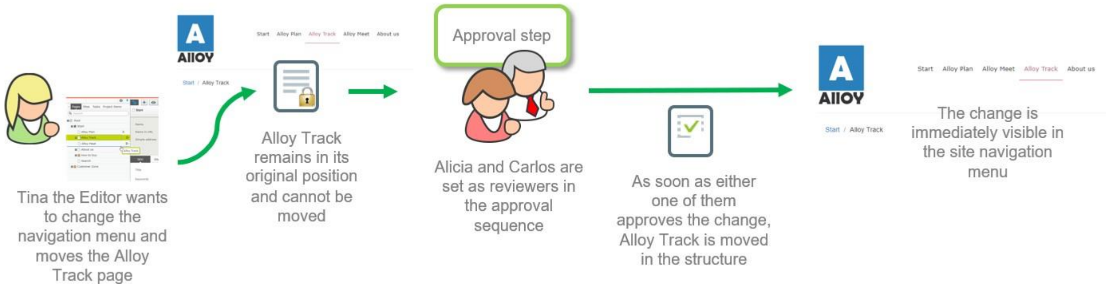
Change approvals affects all versions of the page or block, so while one change is in review, you cannot perform any of the changes that must be approved before being applied.

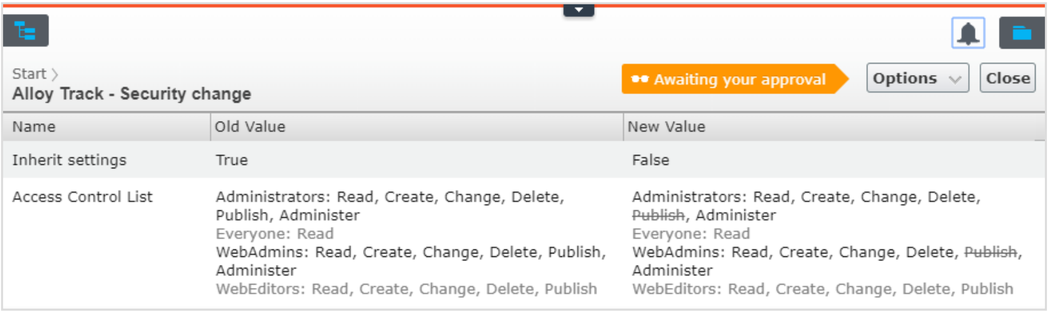
Episerver

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.

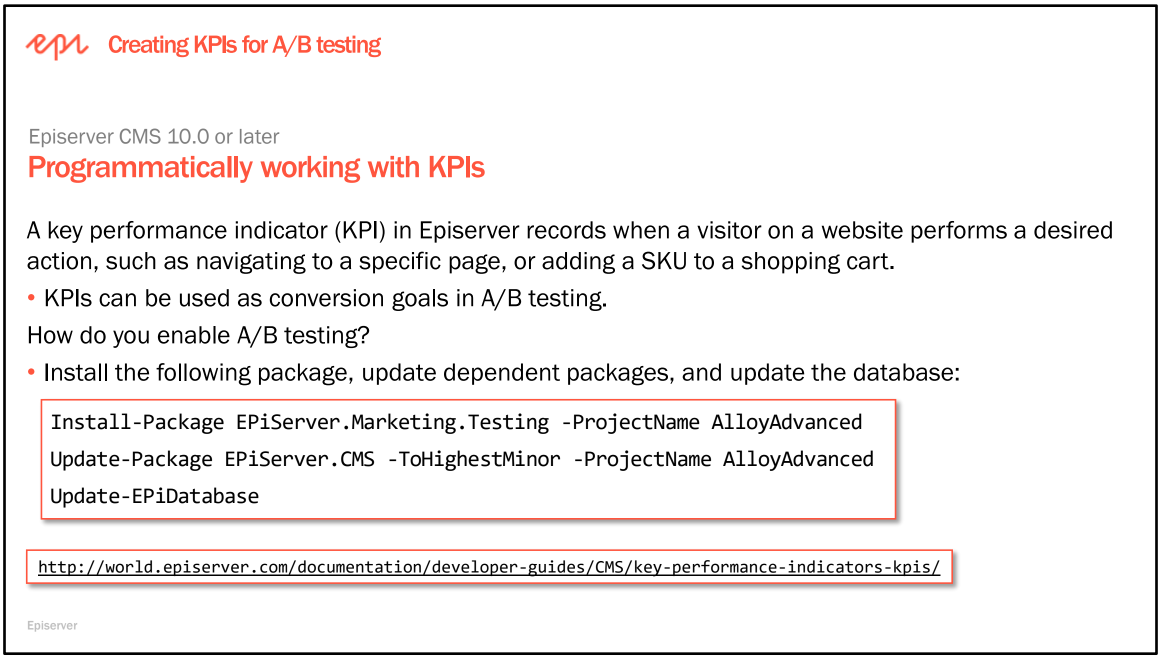

Tina the Editor wants to change the navigation menu and moves the Alloy Track page
Alloy
Start Alloy Plan Alloy Track Alloy Meet About us
Alloy Track remains in its original position and cannot be moved
Approval step
Alicia and Carlos are set as reviewers in the approval sequence
As soon as either one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



Start >
Alloy Track - Security change

Awaiting your approval | Options | Close

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



Creating KPIs for A/B testing

Episerver CMS 10.0 or later

Programmatically working with KPIs

A key performance indicator (KPI) in Episerver records when a visitor on a website performs a desired action, such as navigating to a specific page, or adding a SKU to a shopping cart.

- KPIs can be used as conversion goals in A/B testing.

How do you enable A/B testing?

- Install the following package, update dependent packages, and update the database:

```
Install-Package EPiServer.Marketing.Testing -ProjectName AlloyAdvanced
Update-Package EPiServer.CMS -ToHighestMinor -ProjectName AlloyAdvanced
Update-EPiDatabase
```

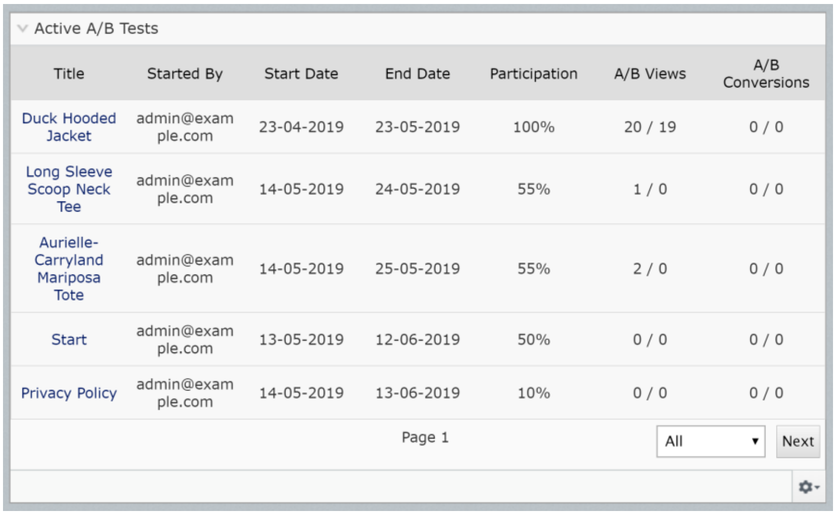
http://world.episerver.com/documentation/developer-guides/CMS/key-performance-indicators-kpis/

Episerver

Introducing the A/B Test List Gadget

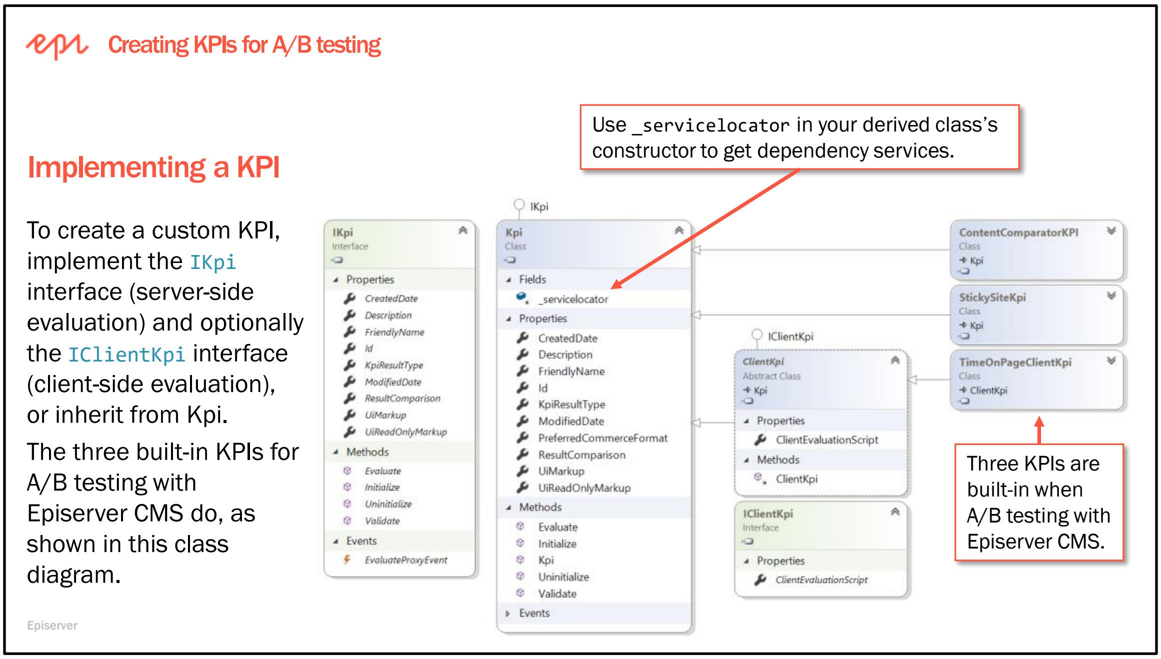
Zone decided to create a CMS dashboard gadget which gives editors a list of running A/B tests, owners, results, views, participation percentage and a direct link to the detailed test overview page. This list can also be filtered based on the test site directly from the component interface.

https://world.episerver.com/blogs/jacob-pretorius/dates/2019/5/introducing-the-ab-test-list-gadget/



Active A/B Tests

Title	Started By	Start Date	End Date	Participation	A/B Views	A/B Conversions
Duck Hooded Jacket	admin@example.com	23-04-2019	23-05-2019	100%	20 / 19	0 / 0
Long Sleeve Scoop Neck Tee	admin@example.com	14-05-2019	24-05-2019	55%	1 / 0	0 / 0
Aurielle-Carryland Mariposa Tote	admin@example.com	14-05-2019	25-05-2019	55%	2 / 0	0 / 0
Start	admin@example.com	13-05-2019	12-06-2019	50%	0 / 0	0 / 0
Privacy Policy	admin@example.com	14-05-2019	13-06-2019	10%	0 / 0	0 / 0



Creating KPIs for A/B testing

Implementing a KPI

To create a custom KPI, implement the `IKpi` interface (server-side evaluation) and optionally the `IClientKpi` interface (client-side evaluation), or inherit from Kpi.

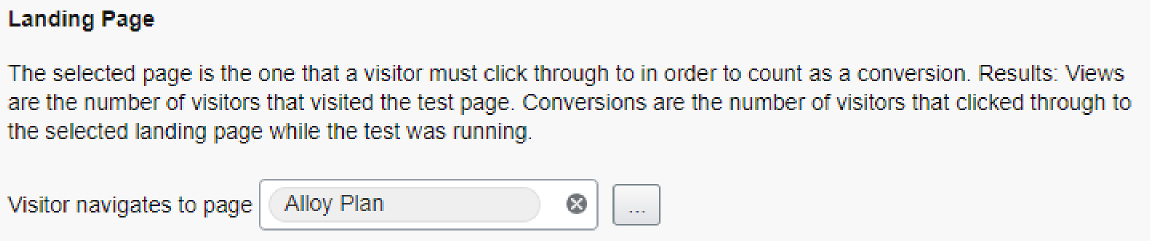
The three built-in KPIs for A/B testing with Episerver CMS do, as shown in this class diagram.

Use `_servicelocator` in your derived class's constructor to get dependency services.

Three KPIs are built-in when A/B testing with Episerver CMS.

Episerver

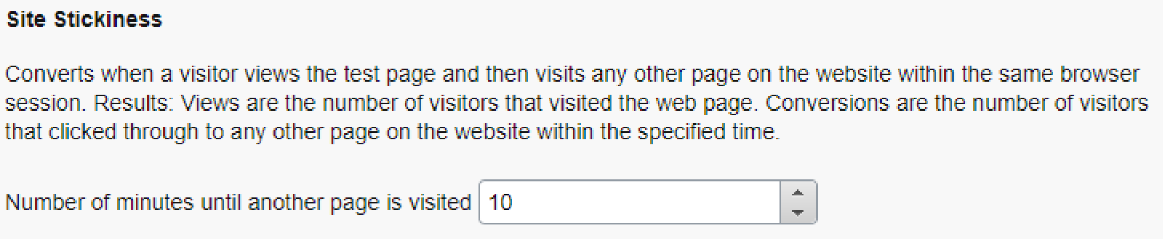
`IClientKpi` is an interface for defining a custom KPI that should be run on the client browser to convert an A/B test. It consists of only one method named `ClientEvaluationScript()` for retrieving the client JavaScript that needs to be presented in the browser to indicate when a conversion takes place.



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.

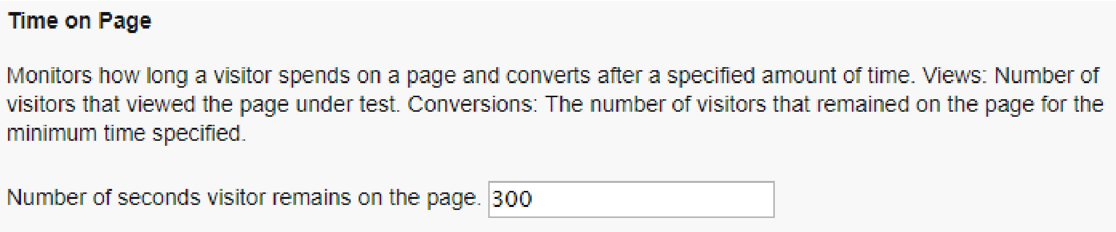
Visitor navigates to page: Alloy Plan



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.

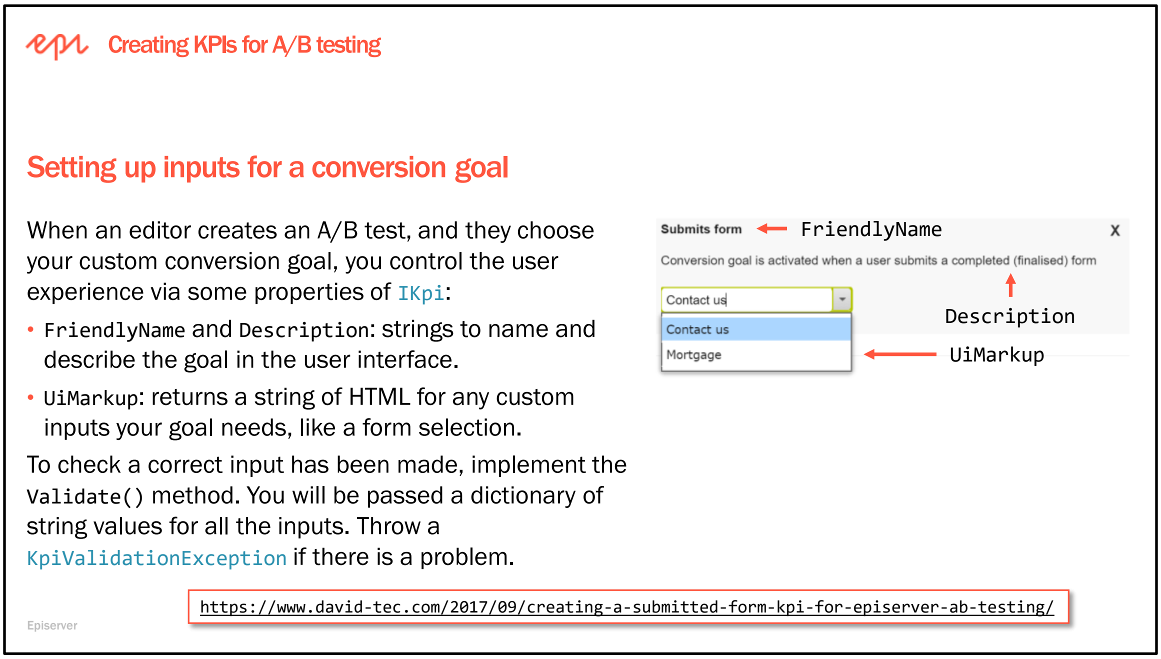
Number of minutes until another page is visited: 10



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



Creating KPIs for A/B testing

Setting up inputs for a conversion goal

When an editor creates an A/B test, and they choose your custom conversion goal, you control the user experience via some properties of `IKpi`:

- `FriendlyName` and `Description`: strings to name and describe the goal in the user interface.
- `UiMarkup`: returns a string of HTML for any custom inputs your goal needs, like a form selection.

To check a correct input has been made, implement the `Validate()` method. You will be passed a dictionary of string values for all the inputs. Throw a `KpiValidationException` if there is a problem.

Submits form
FriendlyName
X
Conversion goal is activated when a user submits a completed (finalised) form
Description
Contact us
Contact us
Mortgage
UiMarkup

https://www.david-tec.com/2017/09/creating-a-submitted-form-kpi-for-episerver-ab-testing/

Episerver

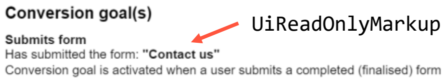
Creating KPIs for A/B testing

Running and evaluating a test with a custom conversion goal

Once a test is running, the implementation of `UiReadOnlyMarkup` is used to show the inputs of the custom conversion goal.

Implement the `EvaluateProxyEvent` event to add and remove a handler for the CMS content event that will trigger the conversion goal.

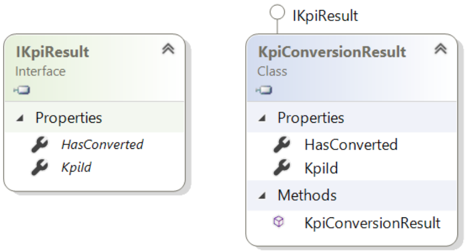
Implement the `Evaluate()` method to return an `IKpiResult` with its `HasConverted` property set to `true` if a conversion has been made.



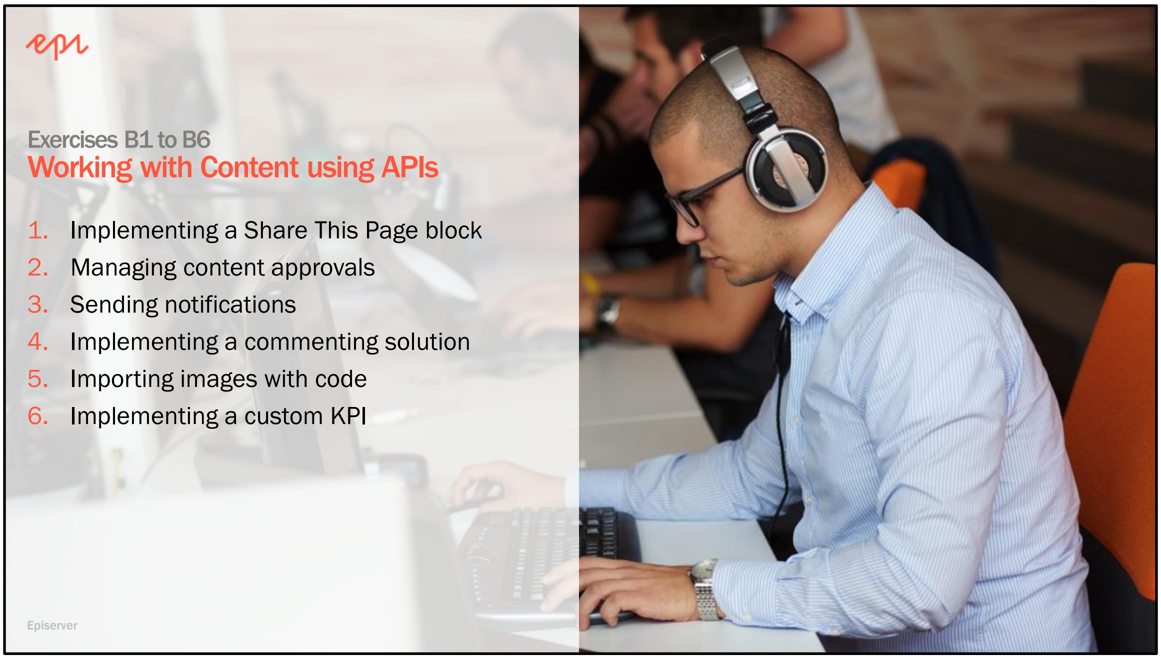
Conversion goal(s)

Submits form
Has submitted the form: **"Contact us"**
Conversion goal is activated when a user submits a completed (finalised) form

UiReadOnlyMarkup



Episerver

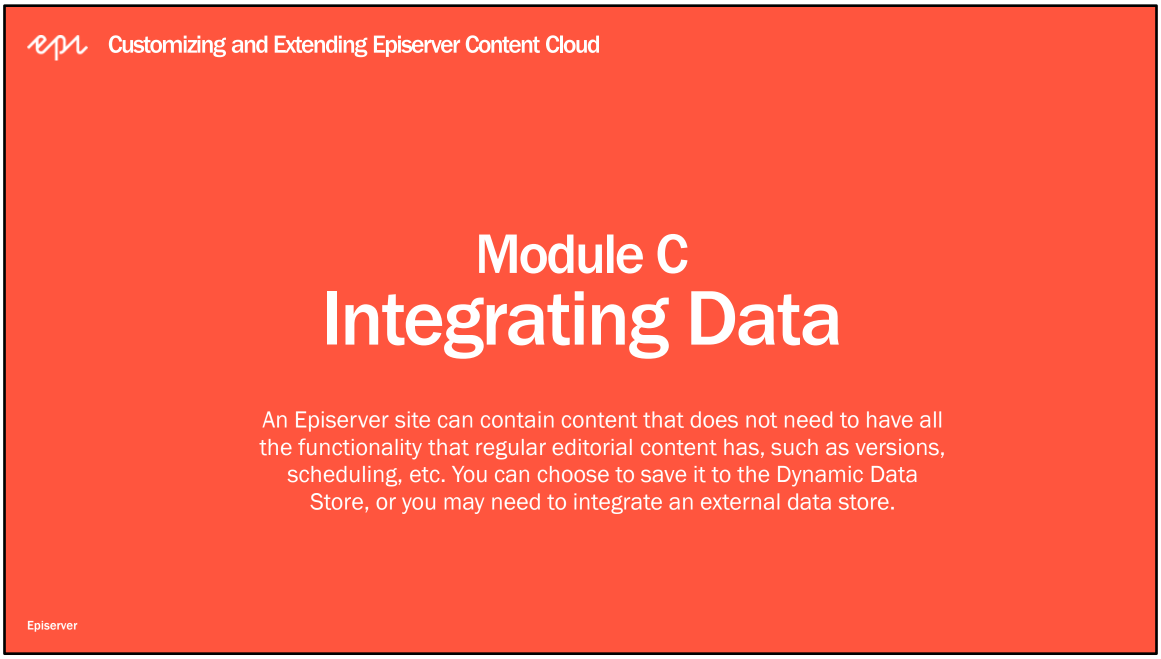


Exercises B1 to B6

Working with Content using APIs

1. Implementing a Share This Page block
2. Managing content approvals
3. Sending notifications
4. Implementing a commenting solution
5. Importing images with code
6. Implementing a custom KPI

Episerver



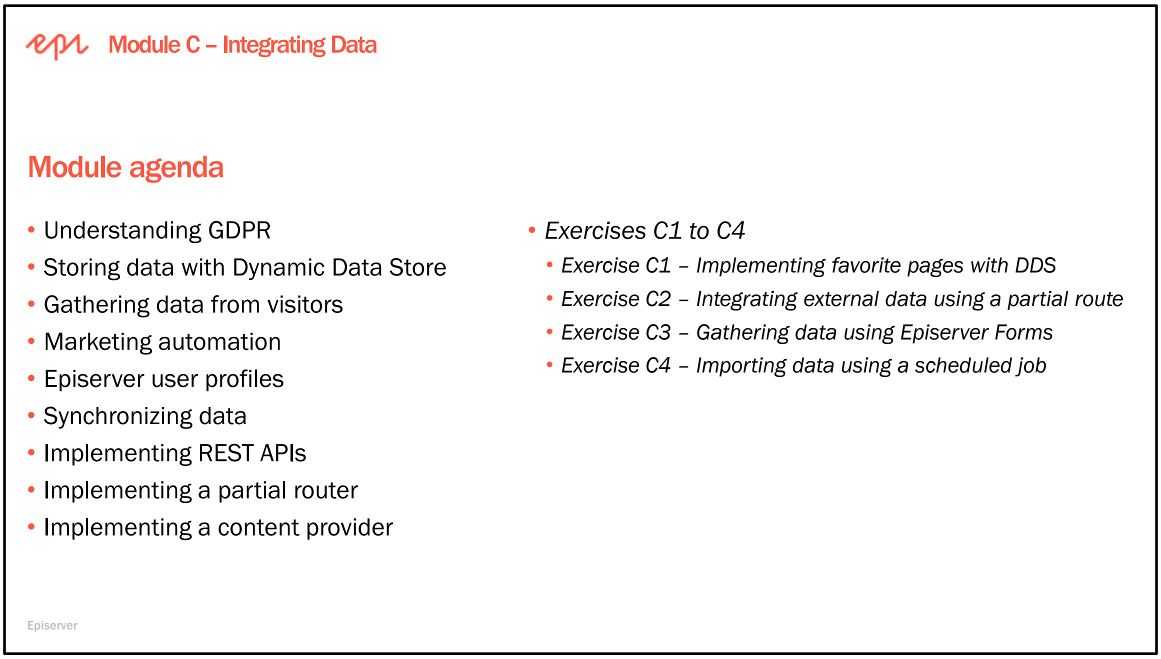
Customizing and Extending Episerver Content Cloud

Module C

Integrating Data

An Episerver site can contain content that does not need to have all the functionality that regular editorial content has, such as versions, scheduling, etc. You can choose to save it to the Dynamic Data Store, or you may need to integrate an external data store.

Episerver



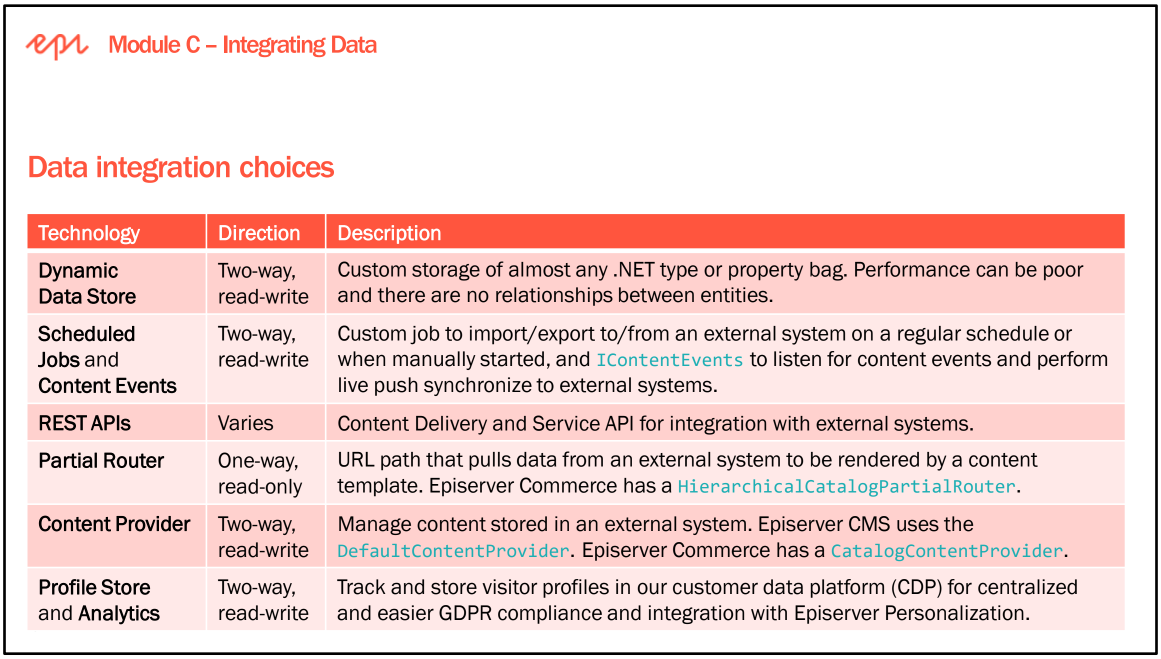
Module C – Integrating Data

Module agenda

- Understanding GDPR
- Storing data with Dynamic Data Store
- Gathering data from visitors
- Marketing automation
- Episerver user profiles
- Synchronizing data
- Implementing REST APIs
- Implementing a partial router
- Implementing a content provider

- *Exercises C1 to C4*
  - *Exercise C1 – Implementing favorite pages with DDS*
  - *Exercise C2 – Integrating external data using a partial route*
  - *Exercise C3 – Gathering data using Episerver Forms*
  - *Exercise C4 – Importing data using a scheduled job*

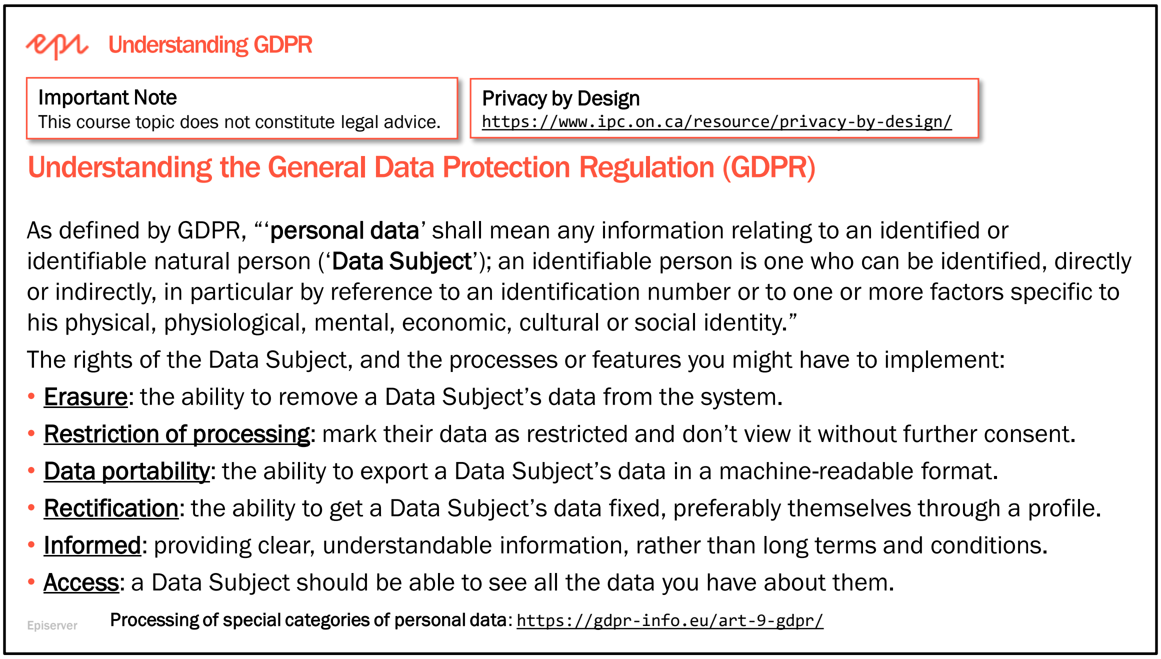
Episerver



Module C – Integrating Data

Data integration choices

Technology	Direction	Description
Dynamic Data Store	Two-way, read-write	Custom storage of almost any .NET type or property bag. Performance can be poor and there are no relationships between entities.
Scheduled Jobs and **Content Events**	Two-way, read-write	Custom job to import/export to/from an external system on a regular schedule or when manually started, and `IContentEvents` to listen for content events and perform live push synchronize to external systems.
REST APIs	Varies	Content Delivery and Service API for integration with external systems.
Partial Router	One-way, read-only	URL path that pulls data from an external system to be rendered by a content template. Episerver Commerce has a `HierarchicalCatalogPartialRouter`.
Content Provider	Two-way, read-write	Manage content stored in an external system. Episerver CMS uses the `DefaultContentProvider`. Episerver Commerce has a `CatalogContentProvider`.
Profile Store and **Analytics**	Two-way, read-write	Track and store visitor profiles in our customer data platform (CDP) for centralized and easier GDPR compliance and integration with Episerver Personalization.



Understanding GDPR

Important Note
This course topic does not constitute legal advice.

Privacy by Design
`https://www.ipc.on.ca/resource/privacy-by-design/`

Understanding the General Data Protection Regulation (GDPR)

As defined by GDPR, "'**personal data**' shall mean any information relating to an identified or identifiable natural person ('**Data Subject**'); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity."

The rights of the Data Subject, and the processes or features you might have to implement:

- **Erasure**: the ability to remove a Data Subject's data from the system.
- **Restriction of processing**: mark their data as restricted and don't view it without further consent.
- **Data portability**: the ability to export a Data Subject's data in a machine-readable format.
- **Rectification**: the ability to get a Data Subject's data fixed, preferably themselves through a profile.
- **Informed**: providing clear, understandable information, rather than long terms and conditions.
- **Access**: a Data Subject should be able to see all the data you have about them.

Episerver

Processing of special categories of personal data: `https://gdpr-info.eu/art-9-gdpr/`

The Episerver platform and GDPR
https://world.episerver.com/documentation/developer-guides/gdpr-guidelines/

Episerver CMS
https://world.episerver.com/documentation/developer-guides/gdpr-guidelines/the-episerver-platform-and-gdpr/episerver-cms/

Episerver Personalization
https://world.episerver.com/documentation/developer-guides/gdpr-guidelines/the-episerver-platform-and-gdpr/episerver-personalization/

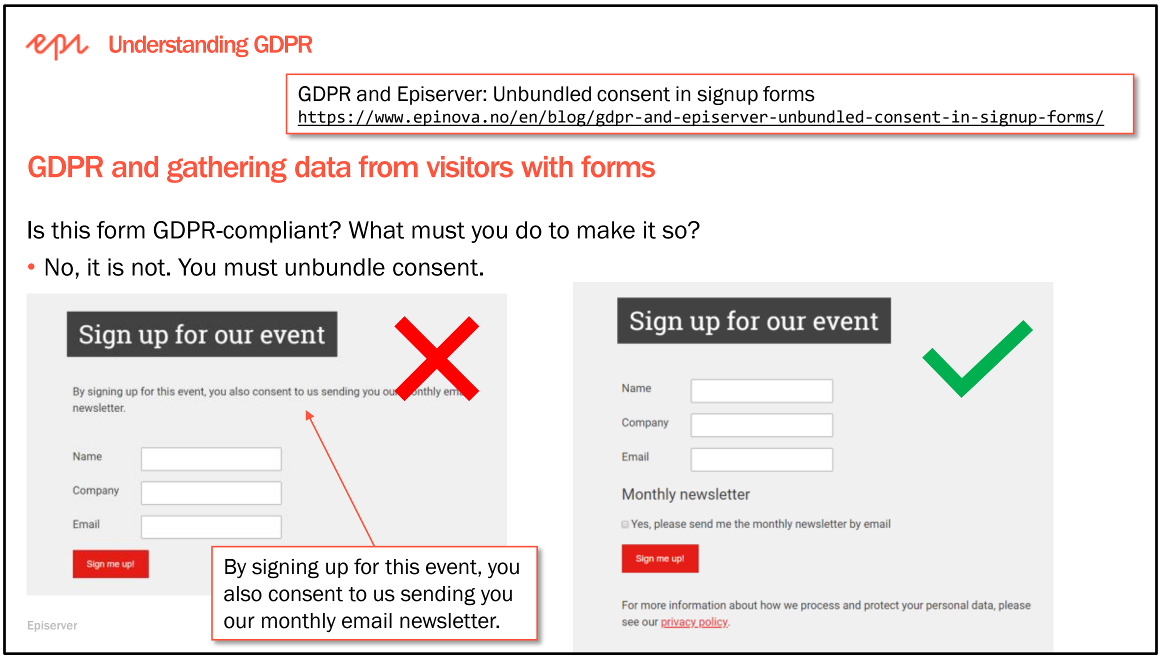
Disable visitor group personalization
`IPersonalizationEvaluator` is an interface that can be implemented to control whether personalization should occur or not. Episerver CMS includes an implementation that checks for presence of a Do Not Track header. If the header is present, no personalization is done for the request and no cookies are stored.
https://world.episerver.com/documentation/developer-guides/CMS/personalization/disable-visitor-group-personalization/

The Ultimate GDPR Guide for Marketers and Businesses
https://appinstitute.com/gdpr-guide/

How GDPR Will Change The Way You Develop
https://www.smashingmagazine.com/2018/02/gdpr-for-web-developers/

GDPR – A PRACTICAL GUIDE FOR DEVELOPERS
https://techblog.bozho.net/gdpr-practical-guide-developers/

GDPR: The difference between Personally Identifiable Information (PII) and Personal Data
https://www.linkedin.com/pulse/gdprthe-difference-between-personally-identifiable-jim-seaman



Understanding GDPR

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

GDPR and gathering data from visitors with forms

Is this form GDPR-compliant? What must you do to make it so?

- No, it is not. 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!

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

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

General Data Protection Regulation and Episerver

Learn how to leverage your organization's data to enable 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

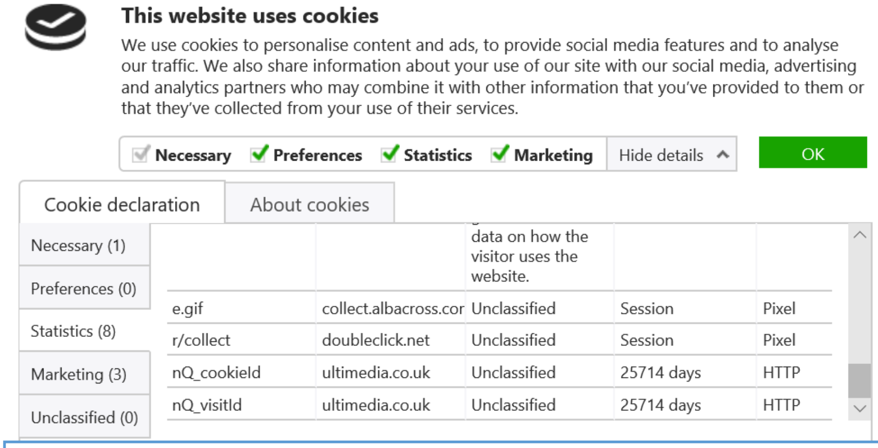
https://www.epinova.no/en/blog/gdpr-compliance-audit-of-the-episerver-qjet-demo-site/

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/

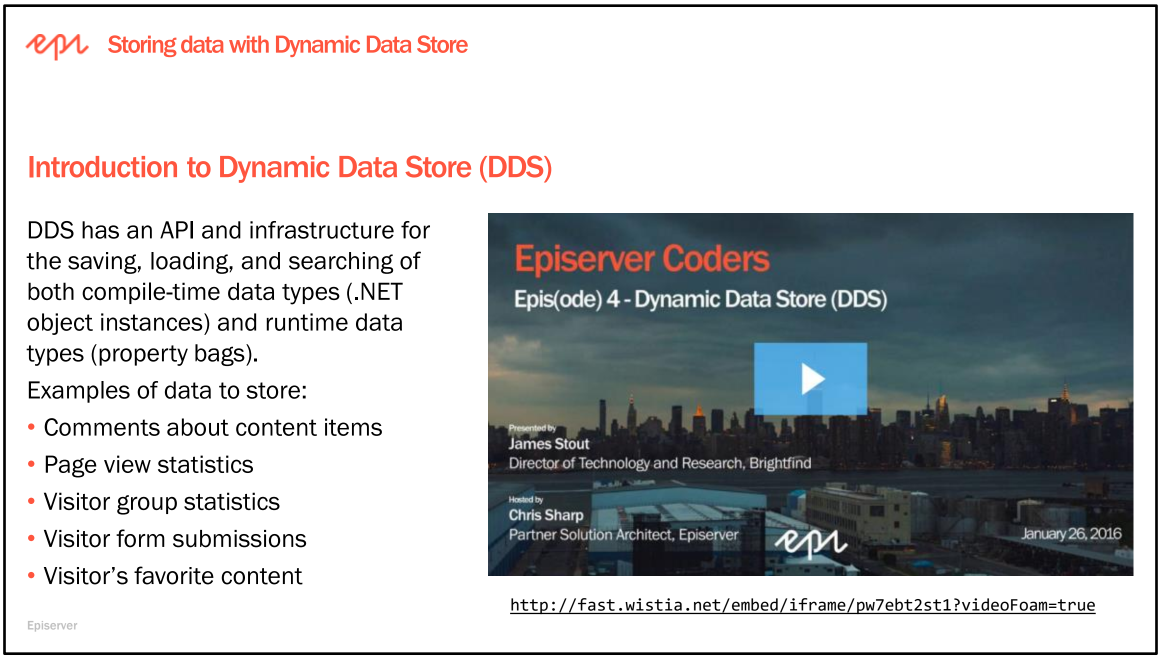


This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services.

		data on how the visitor uses the website.		
e.gif	collect.albacross.cor	Unclassified	Session	Pixel
r/collect	doubleclick.net	Unclassified	Session	Pixel
nQ_cookieId	ultimedia.co.uk	Unclassified	25714 days	HTTP
nQ_visitId	ultimedia.co.uk	Unclassified	25714 days	HTTP

Respect personalization policy to NOT collect data in FormElements
https://world.episerver.com/documentation/Release-Notes/ReleaseNote/?releaseNoteId=AFORM-1636



Storing data with Dynamic Data Store

Introduction to Dynamic Data Store (DDS)

DDS has an API and infrastructure for the saving, loading, and searching of both compile-time data types (.NET object instances) and runtime data types (property bags).

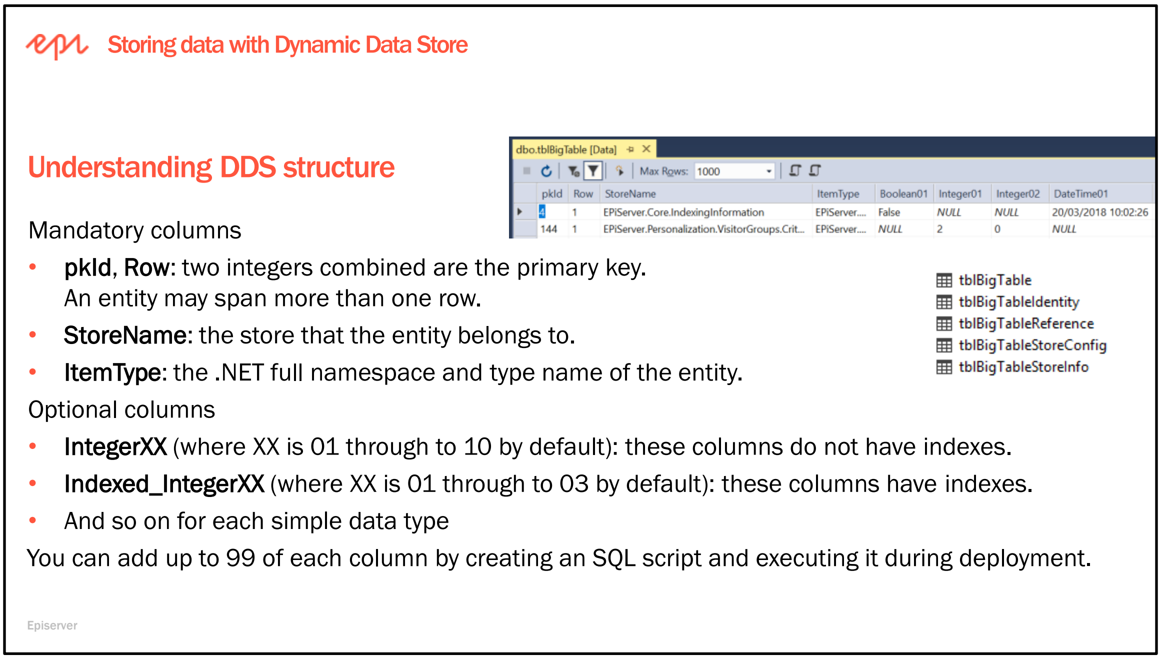
Examples of data to store:

- Comments about content items
- Page view statistics
- Visitor group statistics
- Visitor form submissions
- Visitor’s favorite content

Episerver Coders
Epis(ode) 4 - Dynamic Data Store (DDS)
Presented by
James Stout
Director of Technology and Research, Brightfind
Hosted by
Chris Sharp
Partner Solution Architect, Episerver
January 26, 2016

http://fast.wistia.net/embed/iframe/pw7ebt2st1?videoFoam=true

Episerver



Storing data with Dynamic Data Store

Understanding DDS structure

dbo.tblBigTable [Data]

Max Rows: 1000

	pkId	Row	StoreName	ItemType	Boolean01	Integer01	Integer02	DateTime01
▶	4	1	EPiServer.Core.IndexingInformation	EPiServer....	False	NULL	NULL	20/03/2018 10:02:26
	144	1	EPiServer.Personalization.VisitorGroups.Crit...	EPiServer....	NULL	2	0	NULL

▦ tblBigTable
▦ tblBigTableIdentity
▦ tblBigTableReference
▦ tblBigTableStoreConfig
▦ tblBigTableStoreInfo

Mandatory columns

- **pkId**, **Row**: two integers combined are the primary key.
  An entity may span more than one row.
- **StoreName**: the store that the entity belongs to.
- **ItemType**: the .NET full namespace and type name of the entity.

Optional columns

- **IntegerXX** (where XX is 01 through to 10 by default): these columns do not have indexes.
- **Indexed_IntegerXX** (where XX is 01 through to 03 by default): these columns have indexes.
- And so on for each simple data type

You can add up to 99 of each column by creating an SQL script and executing it during deployment.

Episerver

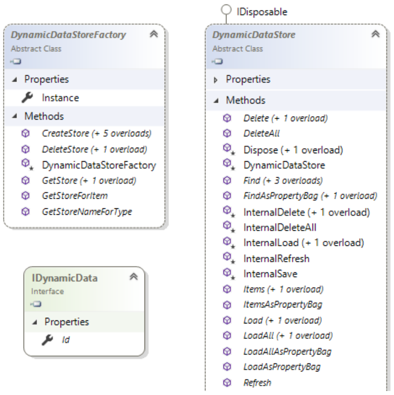
Inline mapping

Inline mapping is where a property of a class or PropertyBag can be mapped directly against one of the tblBigTable database columns. The following types can be mapped inline:

`System.Byte`	`System.Int16`	`System.Int32`	`System.Int64`	`System.Byte[]`
`System.Enum`	`System.Single`	`System.Double`	`System.DateTime`	`System.Char[]`
`System.String`	`System.Char`	`System.Boolean`	`System.Guid`	`EPiServer.Data.Identity`

All properties that cannot be mapped inline or as a collection are mapped as references. This means that their properties are mapped in-turn as a subtype and a link row is added in the reference table to link the parent data structure with the child data structure. This allows for complex trees of data structures (object graphs) to be saved in the Dynamic Data Store at the cost of low performance.

Storing data with Dynamic Data Store



Saving an entity to a DDS store

Define a type that you want to store, that optionally implements `IDynamicData`:

```
public class Favorite : IDynamicData
{
    public Identity Id { get; set; }
    public string UserName { get; set; }
```

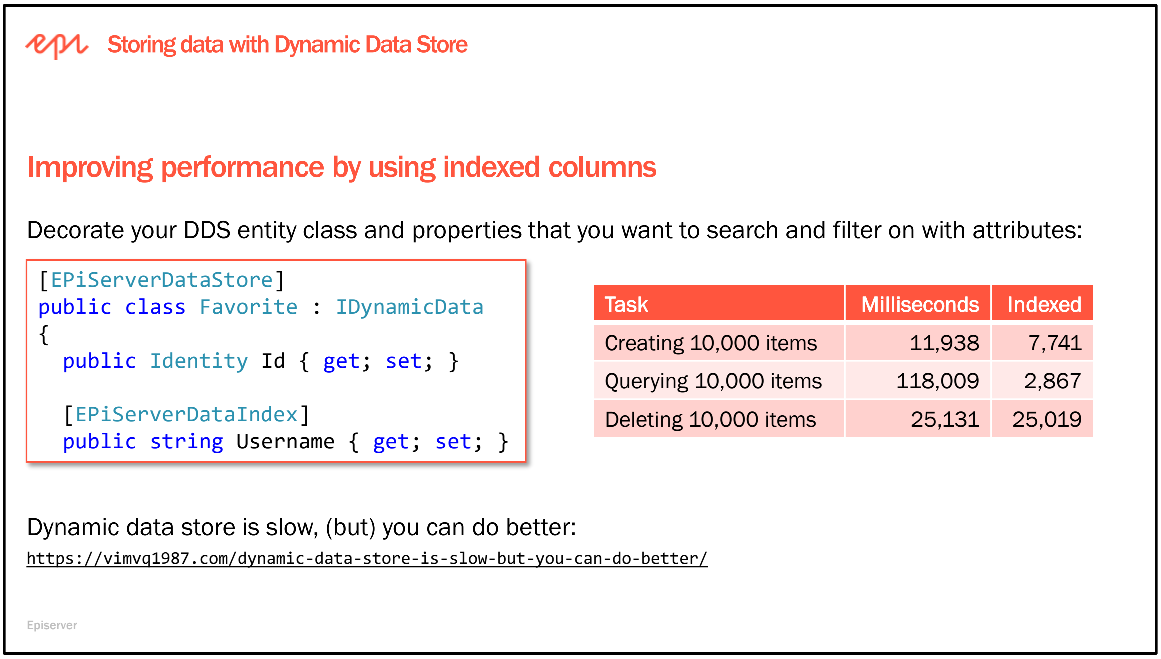
Create a named store:

```
DynamicDataStore store = DynamicDataStoreFactory.Instance
    .CreateStore("Favorites", typeof(Favorite));
```

Save an entity to the store:

```
Favorite fav = new ...;
store.Save(fav);
```

Episerver



Storing data with Dynamic Data Store

Improving performance by using indexed columns

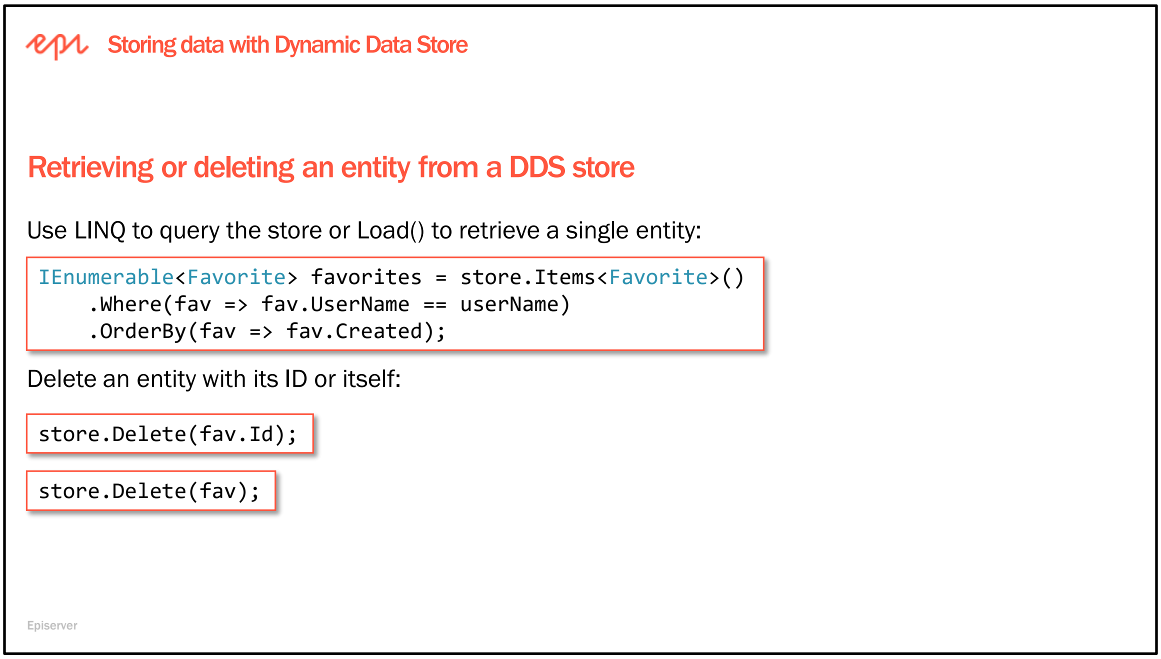
Decorate your DDS entity class and properties that you want to search and filter on with attributes:

```
[EPiServerDataStore]
public class Favorite : IDynamicData
{
  public Identity Id { get; set; }

  [EPiServerDataIndex]
  public string Username { get; set; }
```

Task	Milliseconds	Indexed
Creating 10,000 items	11,938	7,741
Querying 10,000 items	118,009	2,867
Deleting 10,000 items	25,131	25,019

Dynamic data store is slow, (but) you can do better:
https://vimvq1987.com/dynamic-data-store-is-slow-but-you-can-do-better/

Episerver



Storing data with Dynamic Data Store

Retrieving or deleting an entity from a DDS store

Use LINQ to query the store or Load() to retrieve a single entity:

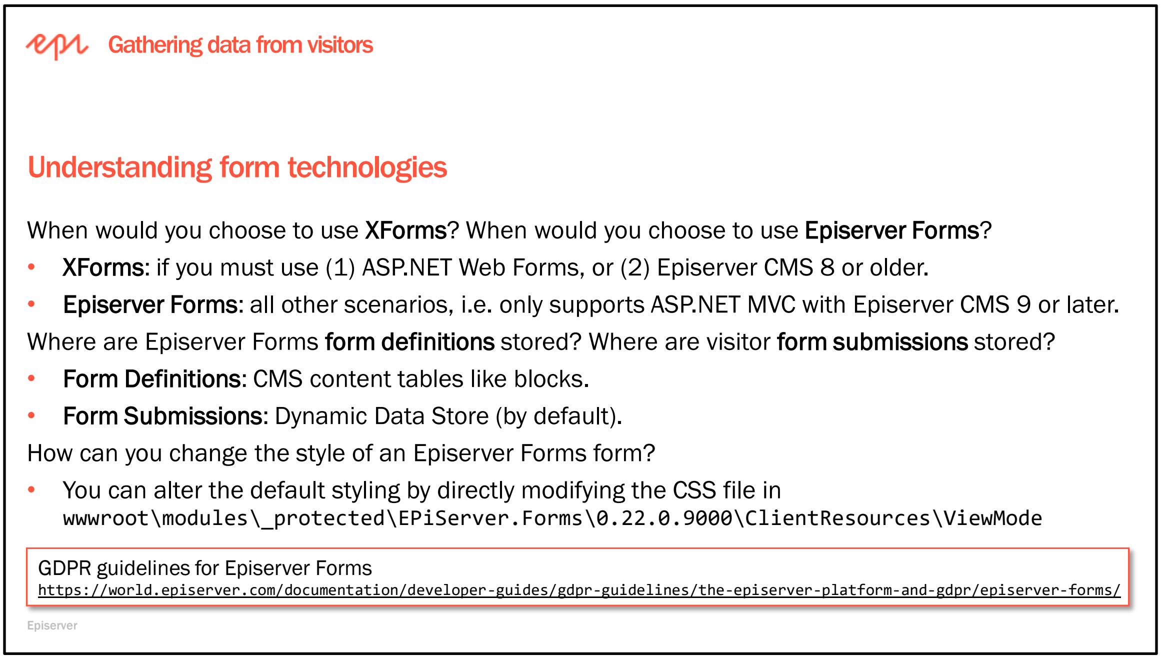
```
IEnumerable<Favorite> favorites = store.Items<Favorite>()
    .Where(fav => fav.UserName == userName)
    .OrderBy(fav => fav.Created);
```

Delete an entity with its ID or itself:

```
store.Delete(fav.Id);
```

```
store.Delete(fav);
```

Episerver



Gathering data from visitors

Understanding form technologies

When would you choose to use **XForms**? When would you choose to use **Episerver Forms**?

- **XForms**: if you must use (1) ASP.NET Web Forms, or (2) Episerver CMS 8 or older.
- **Episerver Forms**: all other scenarios, i.e. only supports ASP.NET MVC with Episerver CMS 9 or later.

Where are Episerver Forms **form definitions** stored? Where are visitor **form submissions** stored?

- **Form Definitions**: CMS content tables like blocks.
- **Form Submissions**: Dynamic Data Store (by default).

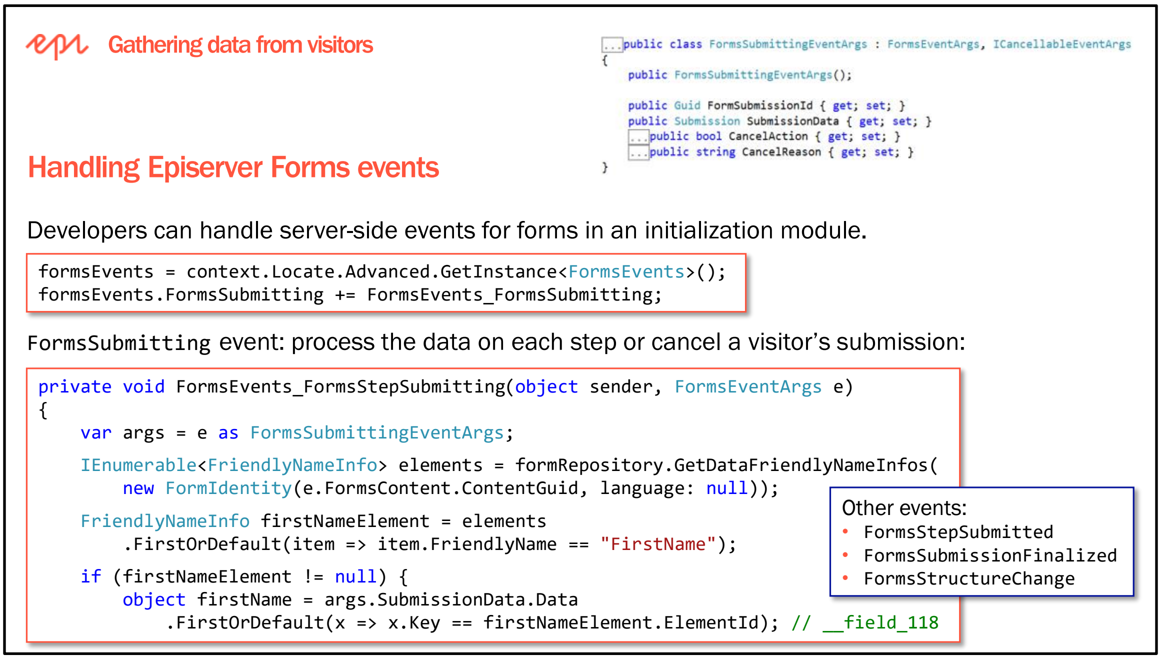
How can you change the style of an Episerver Forms form?

- You can alter the default styling by directly modifying the CSS file in `wwwroot\modules\_protected\EPiServer.Forms\0.22.0.9000\ClientResources\ViewMode`

GDPR guidelines for Episerver Forms
https://world.episerver.com/documentation/developer-guides/gdpr-guidelines/the-episerver-platform-and-gdpr/episerver-forms/

Episerver

Review Episerver Forms documentation
http://world.episerver.com/documentation/developer-guides/forms/
http://world.episerver.com/add-ons/episerver-forms/
http://world.episerver.com/blogs/Allan-Thran/Dates/2015/11/introducing-episerver-forms/
http://world.episerver.com/documentation/developer-guides/forms/css-styling-and-javascript/



Gathering data from visitors

Handling Episerver Forms events

```
...public class FormsSubmittingEventArgs : FormsEventArgs, ICancellableEventArgs
{
    public FormsSubmittingEventArgs();

    public Guid FormSubmissionId { get; set; }
    public Submission SubmissionData { get; set; }
    ...public bool CancelAction { get; set; }
    ...public string CancelReason { get; set; }
}
```

Developers can handle server-side events for forms in an initialization module.

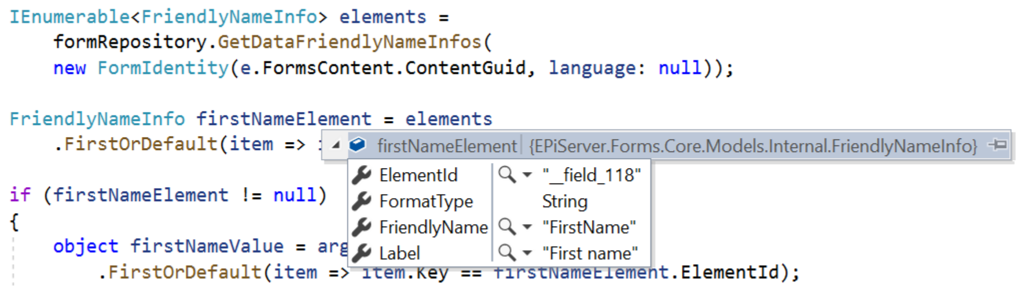
```
formsEvents = context.Locate.Advanced.GetInstance<FormsEvents>();
formsEvents.FormsSubmitting += FormsEvents_FormsSubmitting;
```

`FormsSubmitting` event: process the data on each step or cancel a visitor's submission:

```
private void FormsEvents_FormsStepSubmitting(object sender, FormsEventArgs e)
{
    var args = e as FormsSubmittingEventArgs;
    IEnumerable<FriendlyNameInfo> elements = formRepository.GetDataFriendlyNameInfos(
        new FormIdentity(e.FormsContent.ContentGuid, language: null));
    FriendlyNameInfo firstNameElement = elements
        .FirstOrDefault(item => item.FriendlyName == "FirstName");
    if (firstNameElement != null) {
        object firstName = args.SubmissionData.Data
            .FirstOrDefault(x => x.Key == firstNameElement.ElementId); // __field_118
```

Other events:

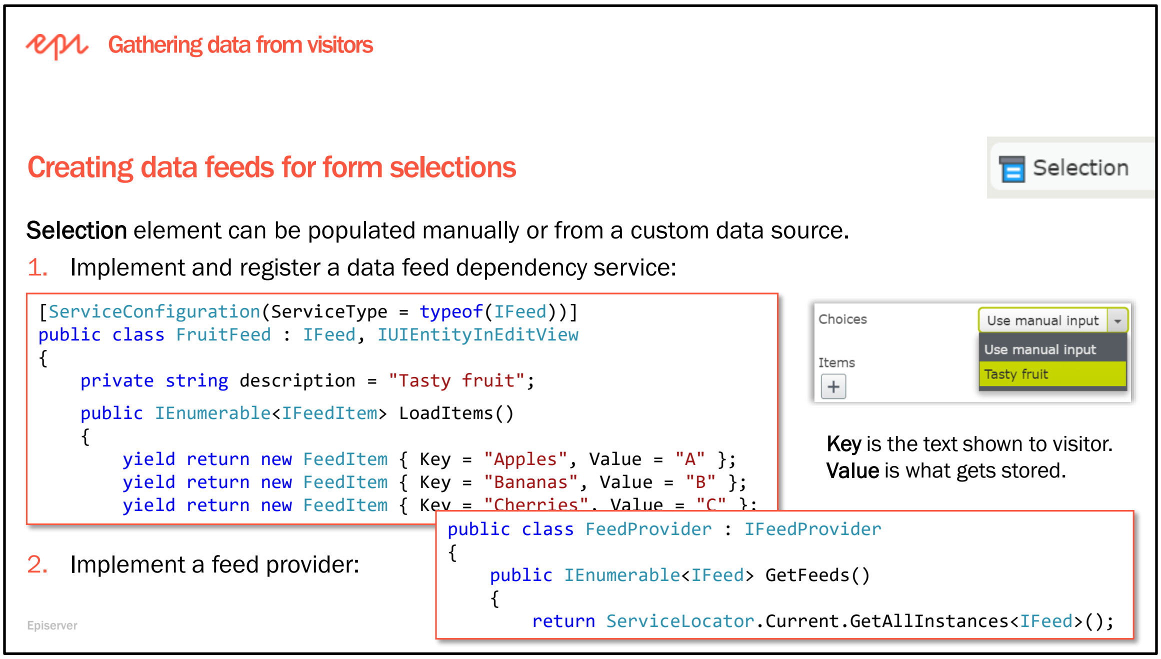
- `FormsStepSubmitted`
- `FormsSubmissionFinalized`
- `FormsStructureChange`



```
IEnumerable<FriendlyNameInfo> elements =
    formRepository.GetDataFriendlyNameInfos(
    new FormIdentity(e.FormsContent.ContentGuid, language: null));

FriendlyNameInfo firstNameElement = elements
    .FirstOrDefault(item => i

if (firstNameElement != null)
{
    object firstNameValue = arg
        .FirstOrDefault(item => item.Key == firstNameElement.ElementId);
```

firstNameElement {EPiServer.Forms.Core.Models.Internal.FriendlyNameInfo}

ElementId	"__field_118"
FormatType	String
FriendlyName	"FirstName"
Label	"First name"



Gathering data from visitors

Creating data feeds for form selections

Selection element can be populated manually or from a custom data source.

1. Implement and register a data feed dependency service:

```
[ServiceConfiguration(ServiceType = typeof(IFeed))]
public class FruitFeed : IFeed, IUIEntityInEditView
{
    private string description = "Tasty fruit";

    public IEnumerable<IFeedItem> LoadItems()
    {
        yield return new FeedItem { Key = "Apples", Value = "A" };
        yield return new FeedItem { Key = "Bananas", Value = "B" };
        yield return new FeedItem { Key = "Cherries", Value = "C" };
```

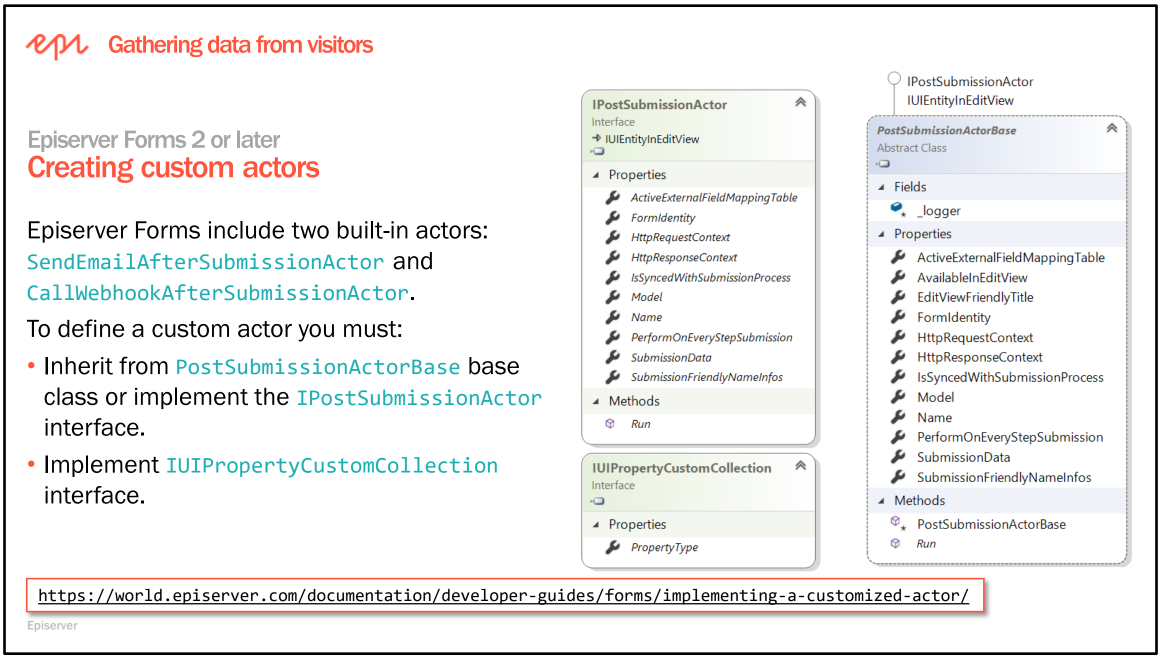
Key is the text shown to visitor.
Value is what gets stored.

2. Implement a feed provider:

```
public class FeedProvider : IFeedProvider
{
    public IEnumerable<IFeed> GetFeeds()
    {
        return ServiceLocator.Current.GetAllInstances<IFeed>();
```

Episerver

https://world.episerver.com/blogs/hieu-nguyen-trung/dates/2017/2/create-data-feeds-for-episerver-forms/



Gathering data from visitors

Episerver Forms 2 or later

Creating custom actors

Episerver Forms include two built-in actors: `SendEmailAfterSubmissionActor` and `CallWebhookAfterSubmissionActor`.

To define a custom actor you must:

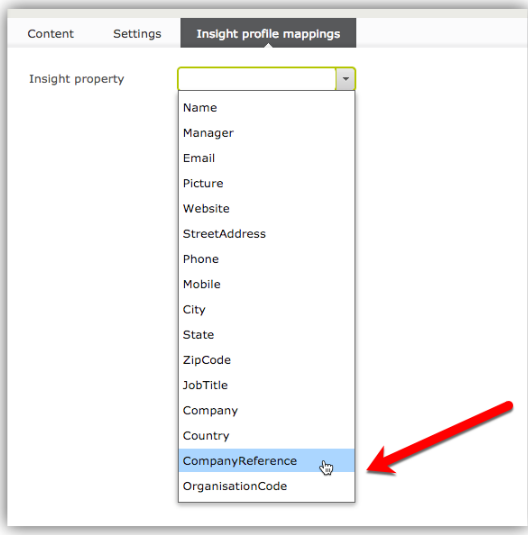
- Inherit from `PostSubmissionActorBase` base class or implement the `IPostSubmissionActor` interface.
- Implement `IUIPropertyCustomCollection` interface.

https://world.episerver.com/documentation/developer-guides/forms/implementing-a-customized-actor/

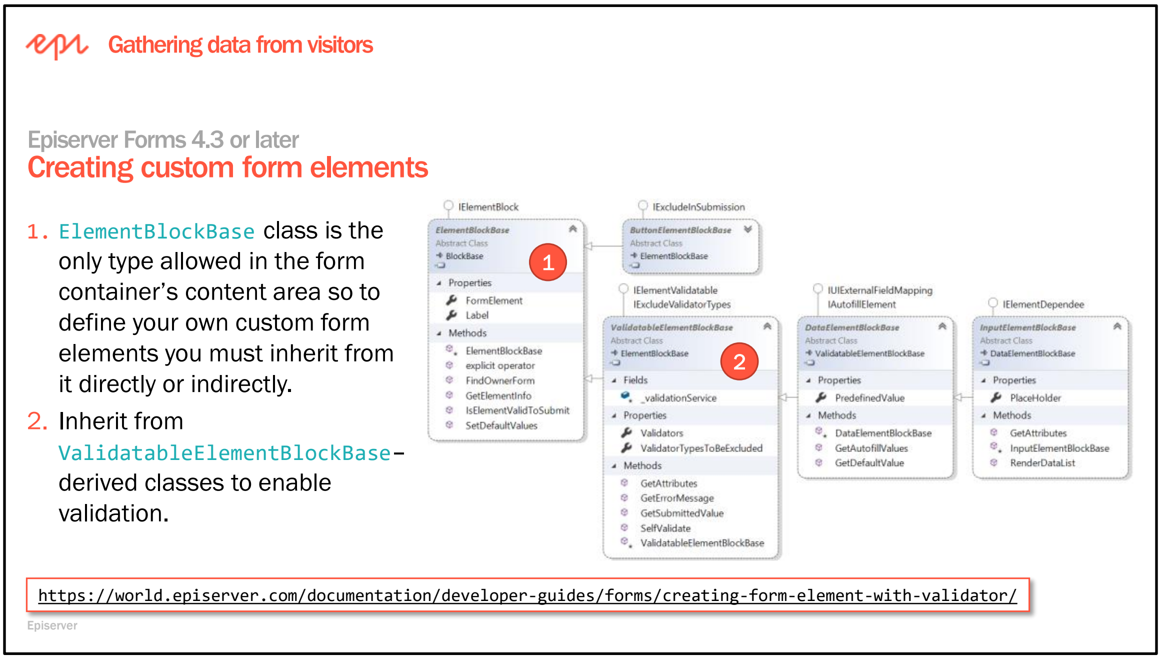
Episerver

Populate Episerver Insight profiles from Episerver Form fields

Episerver Profile store is an tool for capturing profile information and behaviours that can be visualised in Episerver Insight. Episerver Profile store can be connected to any system using standard RESTful APIs to update and add profile information for users. However there isn't currently an out the box way for users to collect user data using Episerver Forms and push this data into Episerver Profile store which can be seen in the Episerver Insight UI. David Knipe decided to create an add-on that would allow editors to map Episerver Form fields to Episerver Insight/Profile store fields. When using it editors set up their form as normal but also get an additional tab called "Insight profile mappings". This tab can be used to specify a property to save the form data to in the Episerver Insight profile.



https://www.david-tec.com/2018/04/populate-episerver-insight-profiles-from-episerver-formfields/



Gathering data from visitors

Episerver Forms 4.3 or later

Creating custom form elements

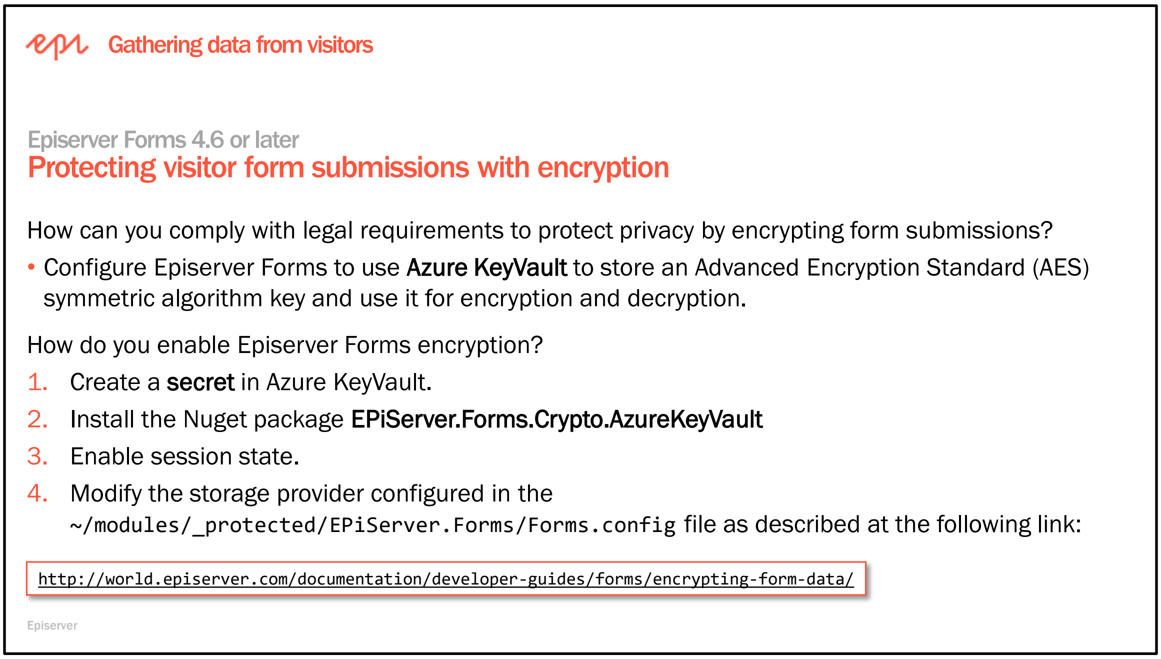
1. `ElementBlockBase` class is the only type allowed in the form container's content area so to define your own custom form elements you must inherit from it directly or indirectly.
2. Inherit from `ValidatableElementBlockBase`-derived classes to enable validation.

https://world.episerver.com/documentation/developer-guides/forms/creating-form-element-with-validator/

Episerver

Extending Episerver Forms: Postcode Lookup Tool
https://world.episerver.com/blogs/david-harlow/dates/2017/12/extending-episerver-forms-postcode-lookup-tool/

Custom FieldSet element block for EPiServer.Forms
https://world.episerver.com/blogs/le-giang/dates/2018/2/custom-fieldset-emelent-block-for-episerver-form/



Gathering data from visitors

Episerver Forms 4.6 or later

Protecting visitor form submissions with encryption

How can you comply with legal requirements to protect privacy by encrypting form submissions?

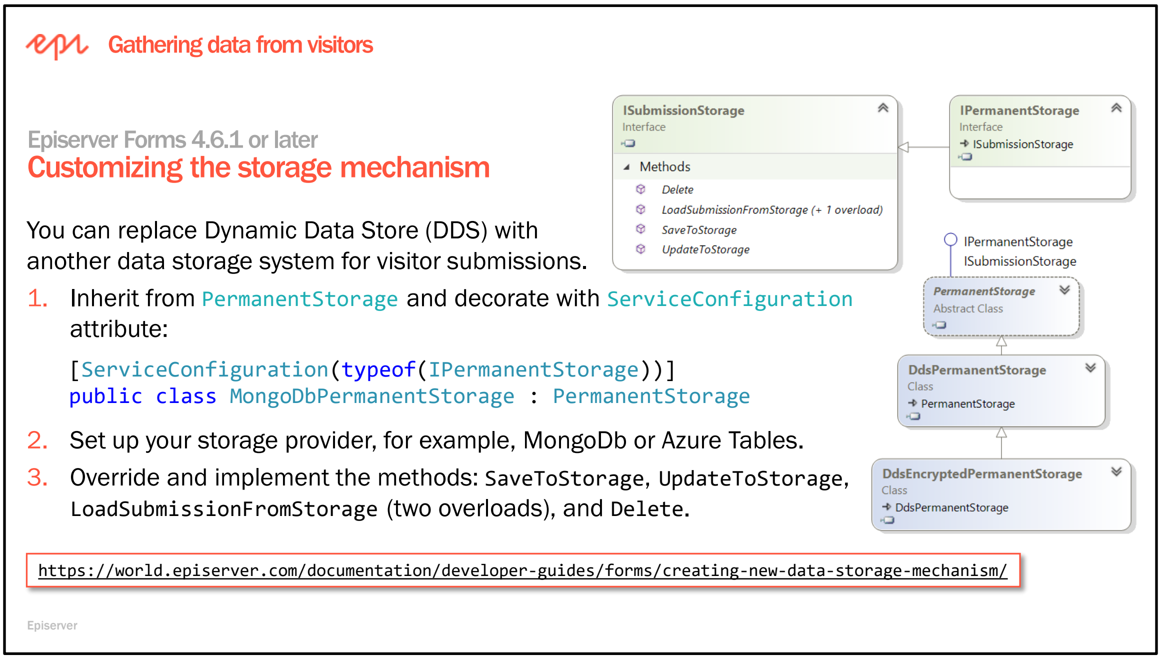
- Configure Episerver Forms to use **Azure KeyVault** to store an Advanced Encryption Standard (AES) symmetric algorithm key and use it for encryption and decryption.

How do you 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



Gathering data from visitors

Episerver Forms 4.6.1 or later

Customizing the storage mechanism

You can replace Dynamic Data Store (DDS) with another data storage system for visitor submissions.

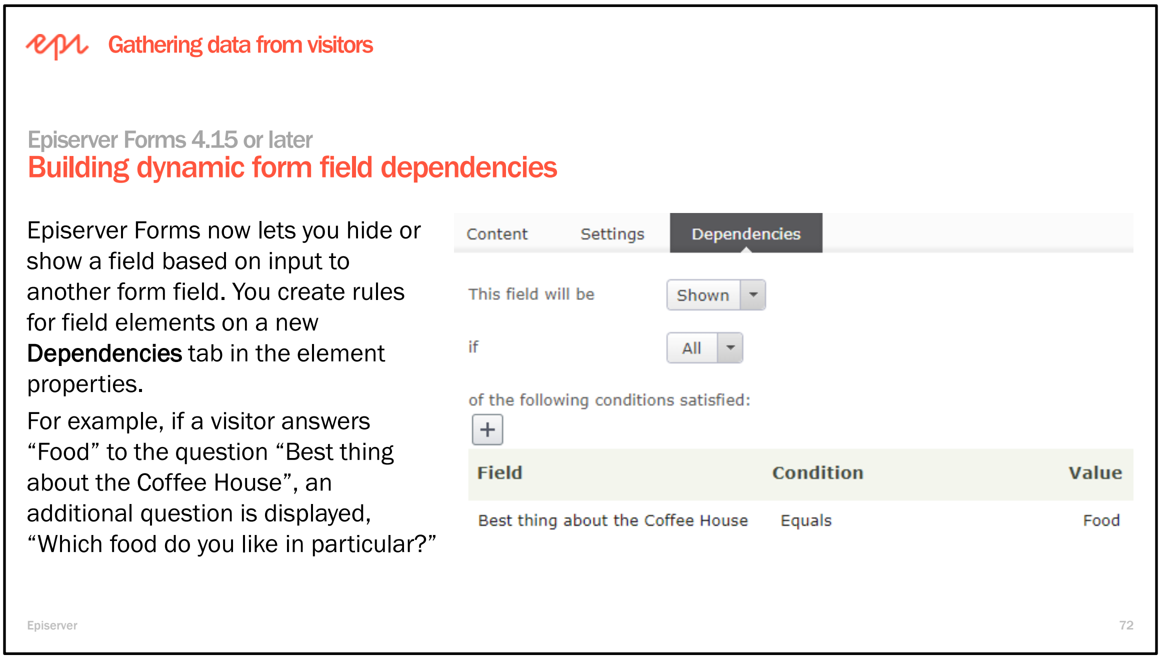
1. Inherit from `PermanentStorage` and decorate with `ServiceConfiguration` attribute:

```
[ServiceConfiguration(typeof(IPermanentStorage))]
public class MongoDbPermanentStorage : PermanentStorage
```

2. Set up your storage provider, for example, MongoDb or Azure Tables.
3. Override and implement the methods: `SaveToStorage`, `UpdateToStorage`, `LoadSubmissionFromStorage` (two overloads), and `Delete`.

https://world.episerver.com/documentation/developer-guides/forms/creating-new-data-storage-mechanism/

Episerver



Gathering data from visitors

Episerver Forms 4.15 or later

Building dynamic form field dependencies

Episerver Forms now lets you hide or show a field based on input to another form field. You create rules for field elements on a new **Dependencies** tab in the element properties.

For example, if a visitor answers “Food” to the question “Best thing about the Coffee House”, an additional question is displayed, “Which food do you like in particular?”

This field will be Shown

if All

of the following conditions satisfied:

Field	Condition	Value
Best thing about the Coffee House	Equals	Food

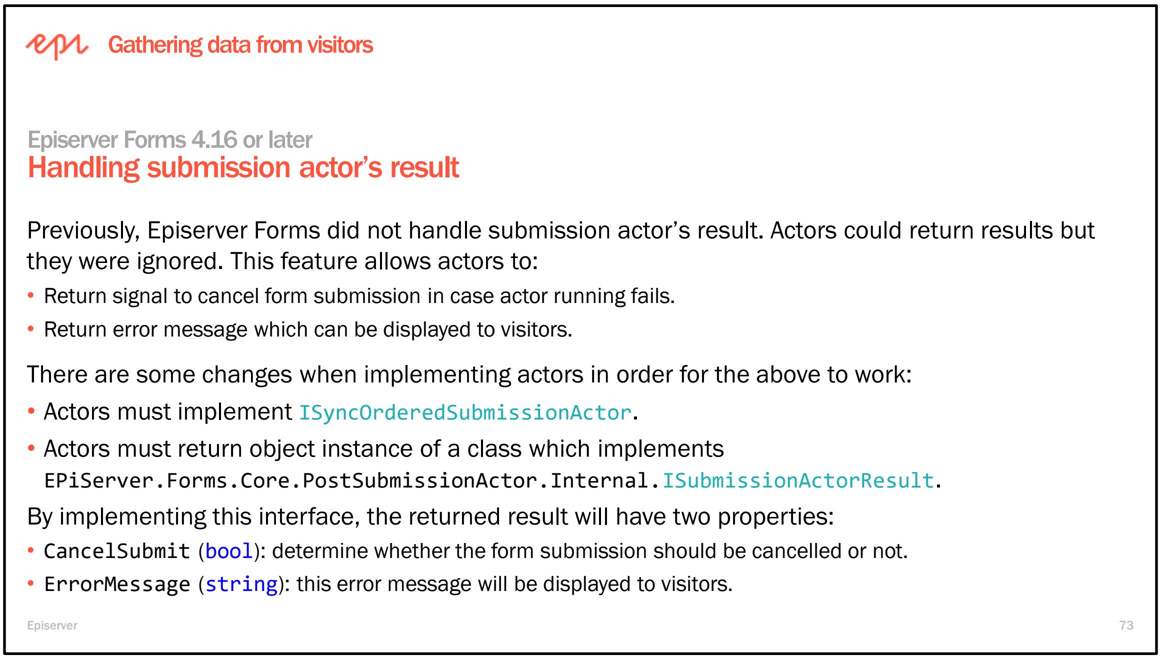
Allow editor to build dynamic form field dependencies

You can create dependency rules for the following field element types:

- Choice element
- ImageChoice element
- Number element
- Range element
- Selection element
- TextArea element
- TextBox element
- Url element
- FileUpload element
- Multi or single choice element

Custom elements (like the ones in Forms.Sample) may not work well with field dependency by default. If you create custom elements, you are responsible for making them compatible with field dependency.

https://world.episerver.com/documentation/Release-Notes/ReleaseNote/?releaseNoteId=AFORM-1499



Gathering data from visitors

Episerver Forms 4.16 or later

Handling submission actor’s result

Previously, Episerver Forms did not handle submission actor’s result. Actors could return results but they were ignored. This feature allows actors to:

- Return signal to cancel form submission in case actor running fails.
- Return error message which can be displayed to visitors.

There are some changes when implementing actors in order for the above to work:

- Actors must implement `ISyncOrderedSubmissionActor`.
- Actors must return object instance of a class which implements `EPiServer.Forms.Core.PostSubmissionActor.Internal.ISubmissionActorResult`.

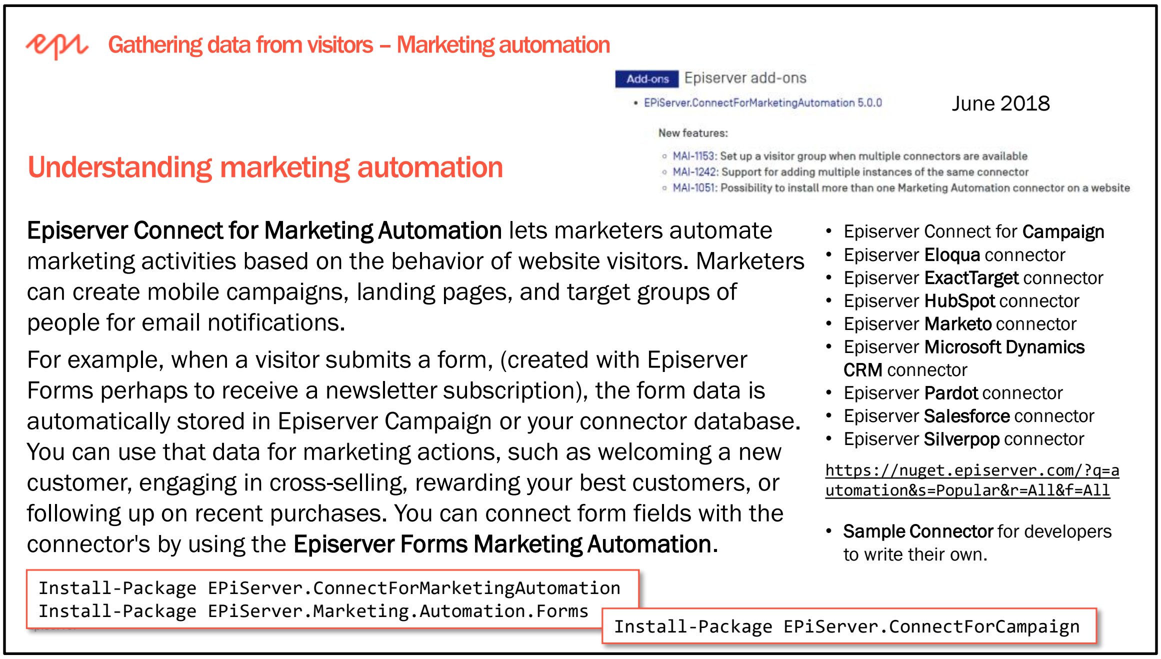
By implementing this interface, the returned result will have two properties:

- `CancelSubmit` (`bool`): determine whether the form submission should be cancelled or not.
- `ErrorMessage` (`string`): this error message will be displayed to visitors.

Episerver 73

Synchronous processing of form submissions

Actors implementing this interface will run synchronously in ascending order, regardless of `IsSyncedWithSubmissionProcess` value (we force the actor to run synchronously because we cannot control the result of async actors).



Gathering data from visitors – Marketing automation

Add-ons Episerver add-ons

- EPiServer.ConnectForMarketingAutomation 5.0.0

June 2018

New features:

- MAI-1153: Set up a visitor group when multiple connectors are available
- MAI-1242: Support for adding multiple instances of the same connector
- MAI-1051: Possibility to install more than one Marketing Automation connector on a website

Understanding marketing automation

Episerver Connect for Marketing Automation lets marketers automate marketing activities based on the behavior of website visitors. Marketers can create mobile campaigns, landing pages, and target groups of people for email notifications.

For example, when a visitor submits a form, (created with Episerver Forms perhaps to receive a newsletter subscription), the form data is automatically stored in Episerver Campaign or your connector database. You can use that data for marketing actions, such as welcoming a new customer, engaging in cross-selling, rewarding your best customers, or following up on recent purchases. You can connect form fields with the connector's by using the **Episerver Forms Marketing Automation**.

```
Install-Package EPiServer.ConnectForMarketingAutomation
Install-Package EPiServer.Marketing.Automation.Forms
```

- Episerver Connect for **Campaign**
- Episerver **Eloqua** connector
- Episerver **ExactTarget** connector
- Episerver **HubSpot** connector
- Episerver **Marketo** connector
- Episerver **Microsoft Dynamics CRM** connector
- Episerver **Pardot** connector
- Episerver **Salesforce** connector
- Episerver **Silverpop** connector

https://nuget.episerver.com/?q=automation&s=Popular&r=All&f=All

- **Sample Connector** for developers to write their own.

```
Install-Package EPiServer.ConnectForCampaign
```

Marketing Automation

System administrators should be aware of the **Fetch data from MAI Connector** scheduled job. It improves the performance of Marketing Automation connectors by fetching and caching databases and lists (wherever applicable) upon site initialization.

http://webhelp.episerver.com/latest/addons/marketing-automation/episerver-connect-for-ma.htm

Sample connector – IMarketingConnector

The Sample Connector demonstrates how you can build custom connectors for use with the Marketing Automation framework.

https://world.episerver.com/add-ons/sample-connector-imarketingconnector/

Episerver Marketing Connectors

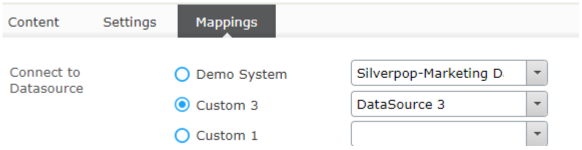
EPiServer Connect for Marketing Automation 5.0.0 package lets you configure multiple instances of a connector with different credentials that will act independently within the CMS. The initial implementation of this feature does not have a user interface so you have to configure the second instance of the same connector with code.

https://world.episerver.com/blogs/jason-masterson/dates/2018/7/episerver-marketing-connectors---multiple-instances/

Multiple external systems

From version 4.18.0, Episerver Forms can support multiple external systems. Editors can choose one of the registered systems as connected data source in the user interface.

https://world.episerver.com/documentation/developer-guides/forms/multiple-external-systems/



Connect to Datasource

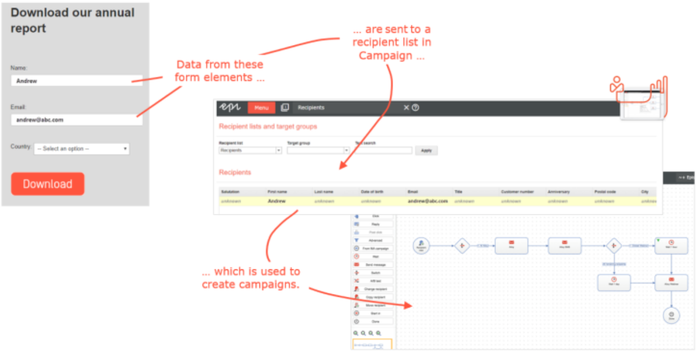
○ Demo System — Silverpop-Marketing D
● Custom 3 — DataSource 3
○ Custom 1

Gathering data from visitors – Marketing automation

Understanding Episerver Connect for Campaign

Episerver Connect for Campaign is an add-on that connects Episerver CMS, Episerver Forms, and Episerver Campaign.

- You can collect visitor data and pass that on to Campaign.
- In Campaign, the data is added to a recipient list, which can be used to create campaigns across different channels such as web, email, and mobile text messages.



Download our annual report

Name: Andrew

Email: andrew@abc.com

Country: -- Select an option --

Download

Data from these form elements ...

... are sent to a recipient list in Campaign ...

... which is used to create campaigns.

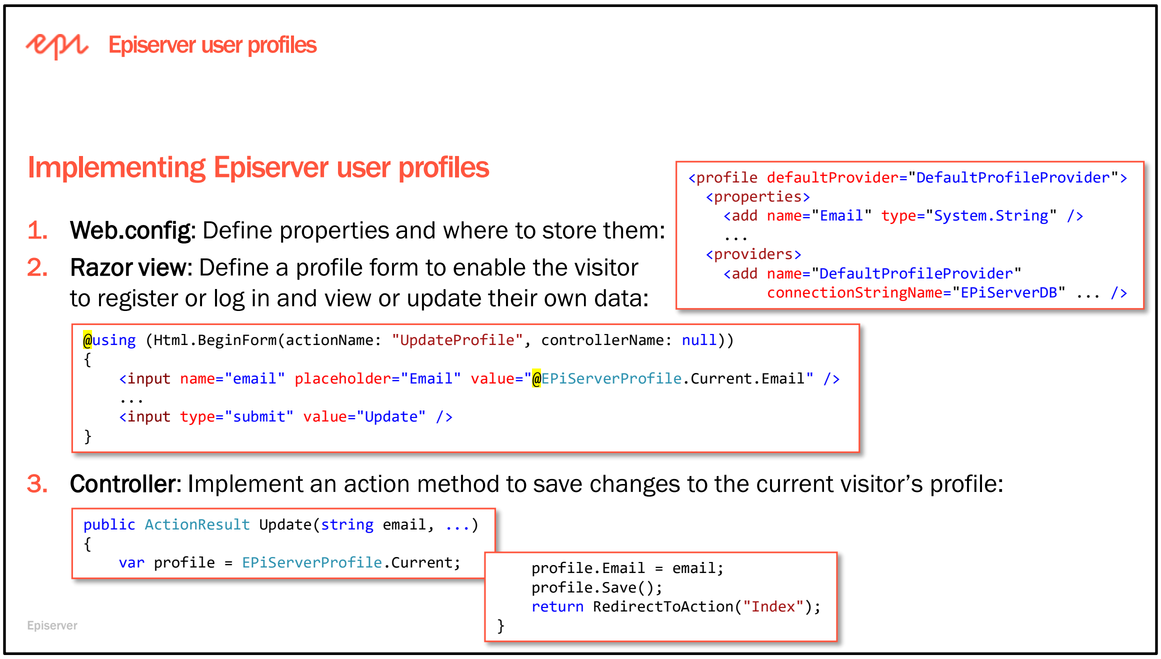
http://webhelp.episerver.com/latest/addons/marketing-automation/connect-for-campaign.htm

Episerver

To start using Connect for Campaign, the following steps must be performed:

1. A developer must install the add-on, as well as Connect for Marketing Automation, Episerver Forms, and the Episerver Forms Marketing Automation connector.
2. The system administrator must authenticate the Connect for Campaign connector with Episerver Campaign.
3. Your website must be set up with Episerver forms.
4. You must map the form to a recipient list in Episerver Campaign.
5. You must map the form elements to specific fields in the recipient list.

http://webhelp.episerver.com/latest/addons/marketing-automation/connect-for-campaign.htm



Episerver user profiles

Implementing Episerver user profiles

1. **Web.config**: Define properties and where to store them:

```
<profile defaultProvider="DefaultProfileProvider">
  <properties>
    <add name="Email" type="System.String" />
    ...
  <providers>
    <add name="DefaultProfileProvider"
         connectionStringName="EPiServerDB" ... />
```

2. **Razor view**: Define a profile form to enable the visitor to register or log in and view or update their own data:

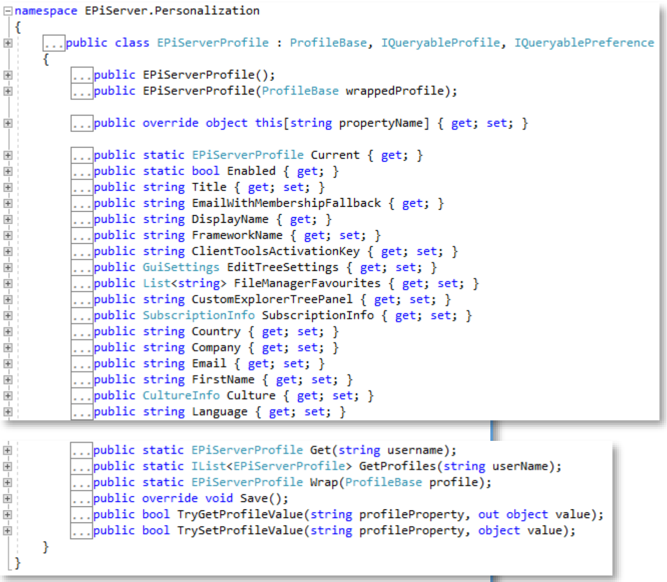
```
@using (Html.BeginForm(actionName: "UpdateProfile", controllerName: null))
{
    <input name="email" placeholder="Email" value="@EPiServerProfile.Current.Email" />
    ...
    <input type="submit" value="Update" />
}
```

3. **Controller**: Implement an action method to save changes to the current visitor's profile:

```
public ActionResult Update(string email, ...)
{
    var profile = EPiServerProfile.Current;
    profile.Email = email;
    profile.Save();
    return RedirectToAction("Index");
}
```

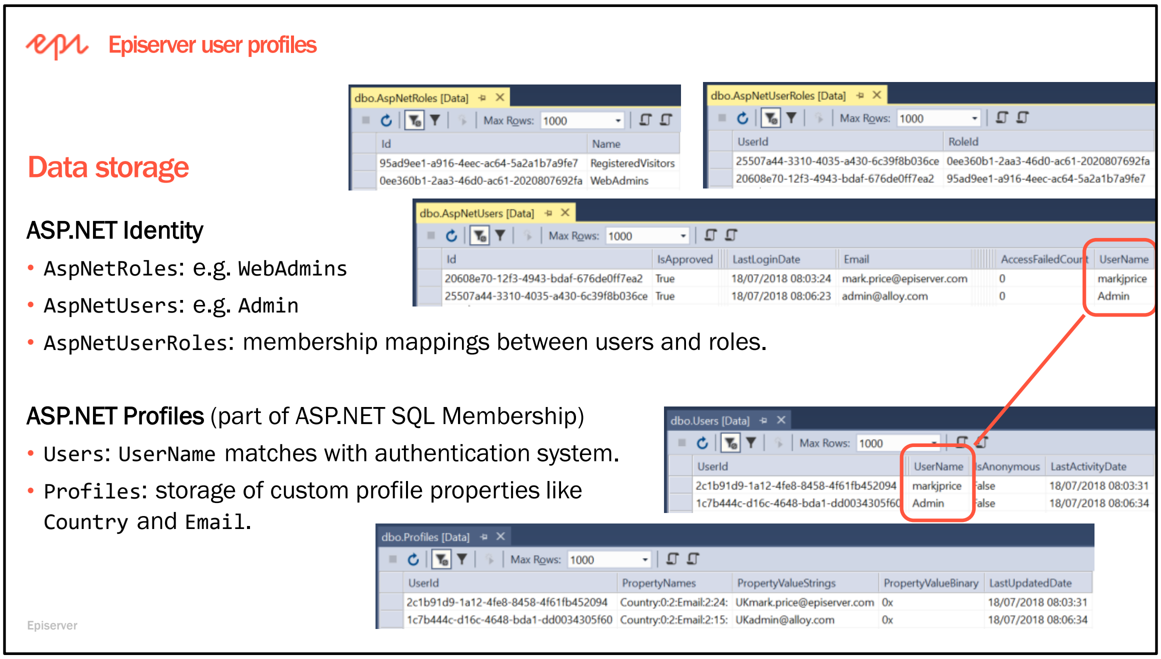
Episerver

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>
```



Episerver user profiles

Data storage

ASP.NET Identity

- `AspNetRoles`: e.g. `WebAdmins`
- `AspNetUsers`: e.g. `Admin`
- `AspNetUserRoles`: membership mappings between users and roles.

ASP.NET Profiles (part of ASP.NET SQL Membership)

- `Users`: `UserName` matches with authentication system.
- `Profiles`: storage of custom profile properties like `Country` and `Email`.

dbo.AspNetRoles [Data]

Id	Name
95ad9ee1-a916-4eec-ac64-5a2a1b7a9fe7	RegisteredVisitors
0ee360b1-2aa3-46d0-ac61-2020807692fa	WebAdmins

dbo.AspNetUserRoles [Data]

UserId	RoleId
25507a44-3310-4035-a430-6c39f8b036ce	0ee360b1-2aa3-46d0-ac61-2020807692fa
20608e70-12f3-4943-bdaf-676de0ff7ea2	95ad9ee1-a916-4eec-ac64-5a2a1b7a9fe7

dbo.AspNetUsers [Data]

Id	IsApproved	LastLoginDate	Email	AccessFailedCount	UserName
20608e70-12f3-4943-bdaf-676de0ff7ea2	True	18/07/2018 08:03:24	mark.price@episerver.com	0	markjprice
25507a44-3310-4035-a430-6c39f8b036ce	True	18/07/2018 08:06:23	admin@alloy.com	0	Admin

dbo.Users [Data]

UserId	UserName	IsAnonymous	LastActivityDate
2c1b91d9-1a12-4fe8-8458-4f61fb452094	markjprice	False	18/07/2018 08:03:31
1c7b444c-d16c-4648-bda1-dd0034305f60	Admin	False	18/07/2018 08:06:34

dbo.Profiles [Data]

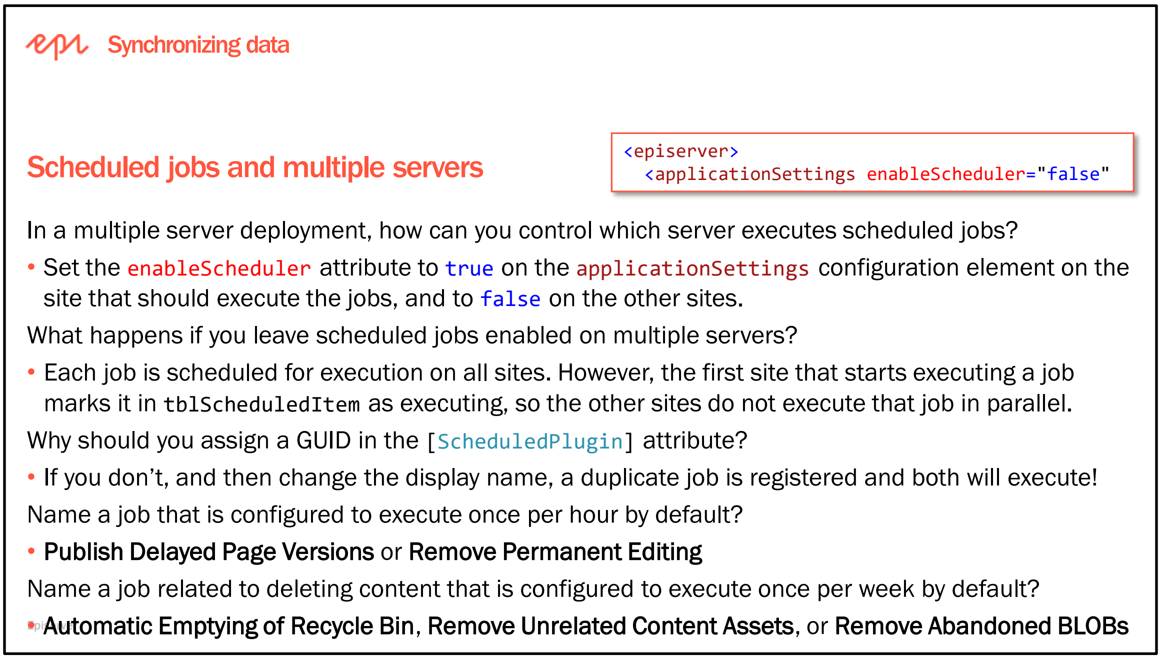
UserId	PropertyNames	PropertyValueStrings	PropertyValueBinary	LastUpdatedDate
2c1b91d9-1a12-4fe8-8458-4f61fb452094	Country:0:2:Email:2:24:	UKmark.price@episerver.com	0x	18/07/2018 08:03:31
1c7b444c-d16c-4648-bda1-dd0034305f60	Country:0:2:Email:2:15:	UKadmin@alloy.com	0x	18/07/2018 08:06:34

Episerver

Set the correct email address

If you use the `[PageViewTracking]` attribute or the `ITrackingService` to track page views and you do not explicitly set the `User` property, then you must make sure that the correct username and email are set in the Episerver profile system, not in the authentication system. The email address stored in the **AspNetUsers** table is ignored by Profile Store, and it uses the `UserName` and `Email` in the **Profiles** table instead.

The tracking data interceptor named `UserDataTrackingDataInterceptor` is registered with a `SortOrder` of `210`, and will check the `User` property. If it is `null`, then it sets it to use the `UserName` and `Email` from the visitor's Episerver profile. It will also add three profile properties to the `Info` dictionary: `Title`, `Company`, and `Country`.



Synchronizing data

Scheduled jobs and multiple servers

```
<episerver>
  <applicationSettings enableScheduler="false"
```

In a multiple server deployment, how can you control which server executes scheduled jobs?

- Set the `enableScheduler` attribute to `true` on the `applicationSettings` configuration element on the site that should execute the jobs, and to `false` on the other sites.

What happens if you leave scheduled jobs enabled on multiple servers?

- Each job is scheduled for execution on all sites. However, the first site that starts executing a job marks it in `tblScheduledItem` as executing, so the other sites do not execute that job in parallel.

Why should you assign a GUID in the `[ScheduledPlugin]` attribute?

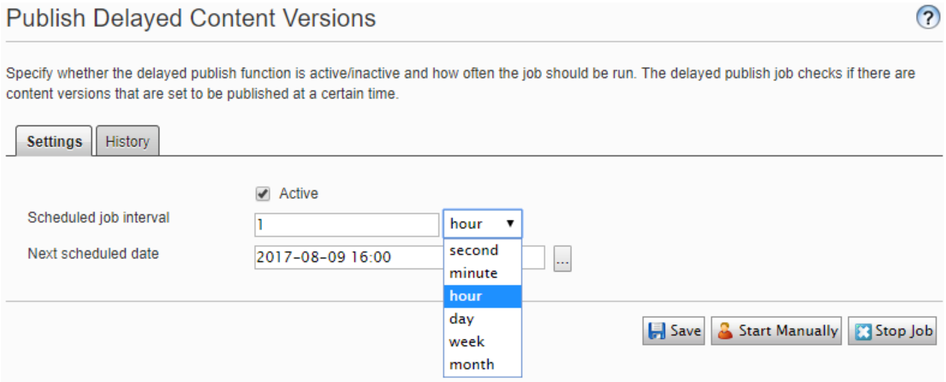
- If you don't, and then change the display name, a duplicate job is registered and both will execute!

Name a job that is configured to execute once per hour by default?

- **Publish Delayed Page Versions** or **Remove Permanent Editing**

Name a job related to deleting content that is configured to execute once per week by default?

- **Automatic Emptying of Recycle Bin**, **Remove Unrelated Content Assets**, or **Remove Abandoned BLOBs**



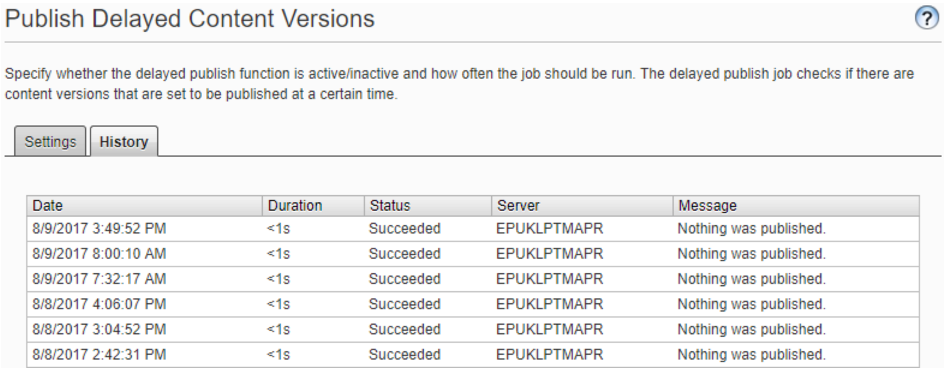
Publish Delayed Content Versions

Specify whether the delayed publish function is active/inactive and how often the job should be run. The delayed publish job checks if there are content versions that are set to be published at a certain time.

☑ Active

Scheduled job interval: 1 hour

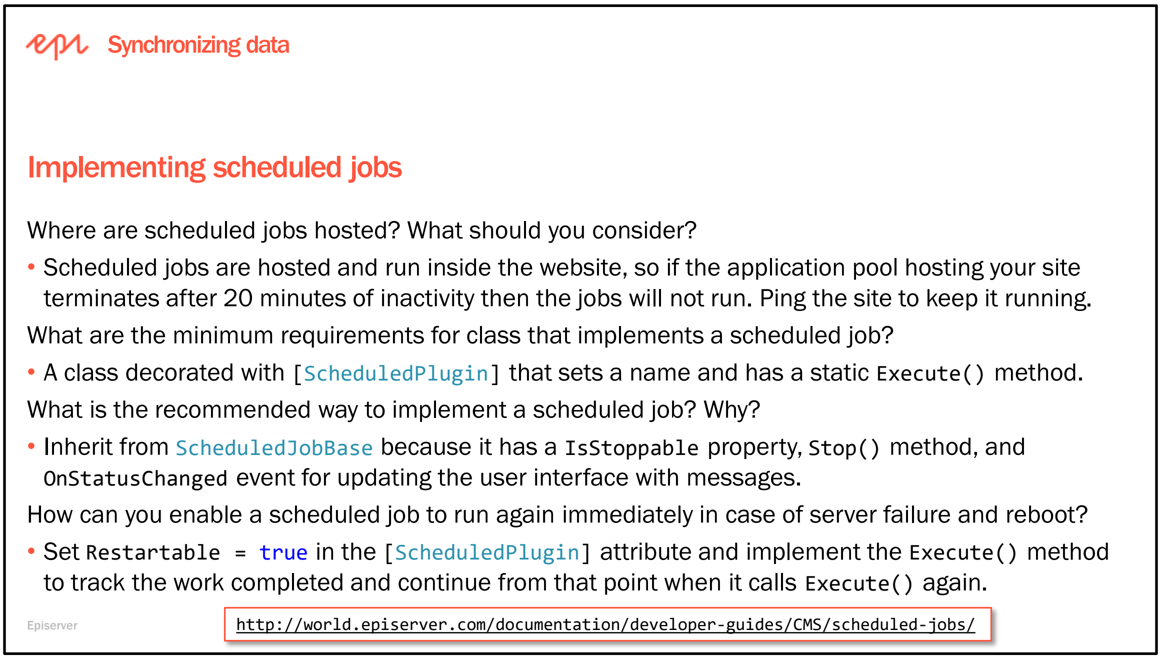
Next scheduled date: 2017-08-09 16:00



Publish Delayed Content Versions

Specify whether the delayed publish function is active/inactive and how often the job should be run. The delayed publish job checks if there are content versions that are set to be published at a certain time.

Date	Duration	Status	Server	Message
8/9/2017 3:49:52 PM	<1s	Succeeded	EPUKLPTMAPR	Nothing was published.
8/9/2017 8:00:10 AM	<1s	Succeeded	EPUKLPTMAPR	Nothing was published.
8/9/2017 7:32:17 AM	<1s	Succeeded	EPUKLPTMAPR	Nothing was published.
8/8/2017 4:06:07 PM	<1s	Succeeded	EPUKLPTMAPR	Nothing was published.
8/8/2017 3:04:52 PM	<1s	Succeeded	EPUKLPTMAPR	Nothing was published.
8/8/2017 2:42:31 PM	<1s	Succeeded	EPUKLPTMAPR	Nothing was published.



Synchronizing data

Implementing scheduled jobs

Where are scheduled jobs hosted? What should you consider?

- Scheduled jobs are hosted and run inside the website, so if the application pool hosting your site terminates after 20 minutes of inactivity then the jobs will not run. Ping the site to keep it running.

What are the minimum requirements for class that implements a scheduled job?

- A class decorated with `[ScheduledPlugin]` that sets a name and has a static `Execute()` method.

What is the recommended way to implement a scheduled job? Why?

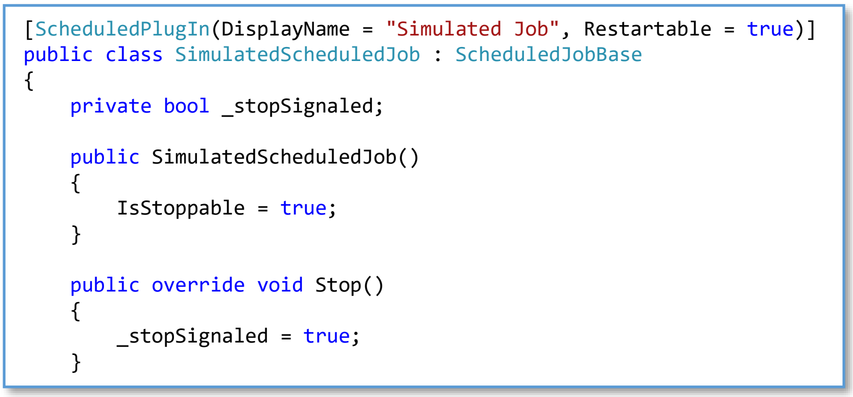
- Inherit from `ScheduledJobBase` because it has a `IsStoppable` property, `Stop()` method, and `OnStatusChanged` event for updating the user interface with messages.

How can you enable a scheduled job to run again immediately in case of server failure and reboot?

- Set `Restartable = true` in the `[ScheduledPlugin]` attribute and implement the `Execute()` method to track the work completed and continue from that point when it calls `Execute()` again.

Episerver

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

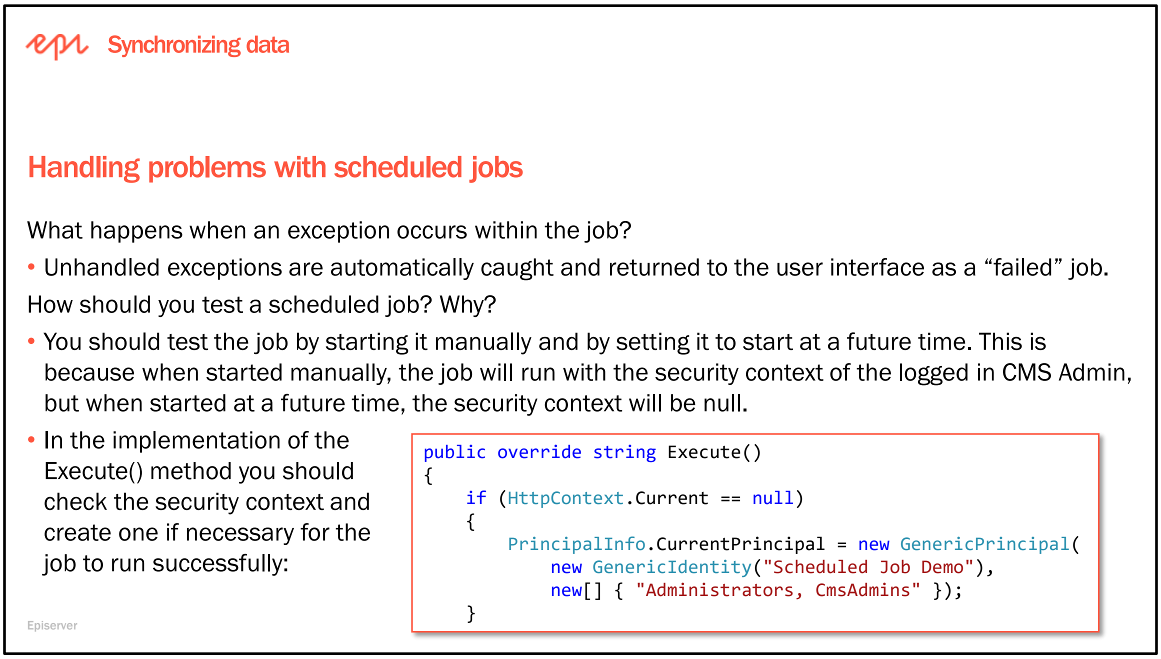


```
[ScheduledPlugIn(DisplayName = "Simulated Job", Restartable = true)]
public class SimulatedScheduledJob : ScheduledJobBase
{
    private bool _stopSignaled;

    public SimulatedScheduledJob()
    {
        IsStoppable = true;
    }

    public override void Stop()
    {
        _stopSignaled = true;
    }
```

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.

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.



Synchronizing data

Handling problems with scheduled jobs

What happens when an exception occurs within the job?

- Unhandled exceptions are automatically caught and returned to the user interface as a “failed” job.

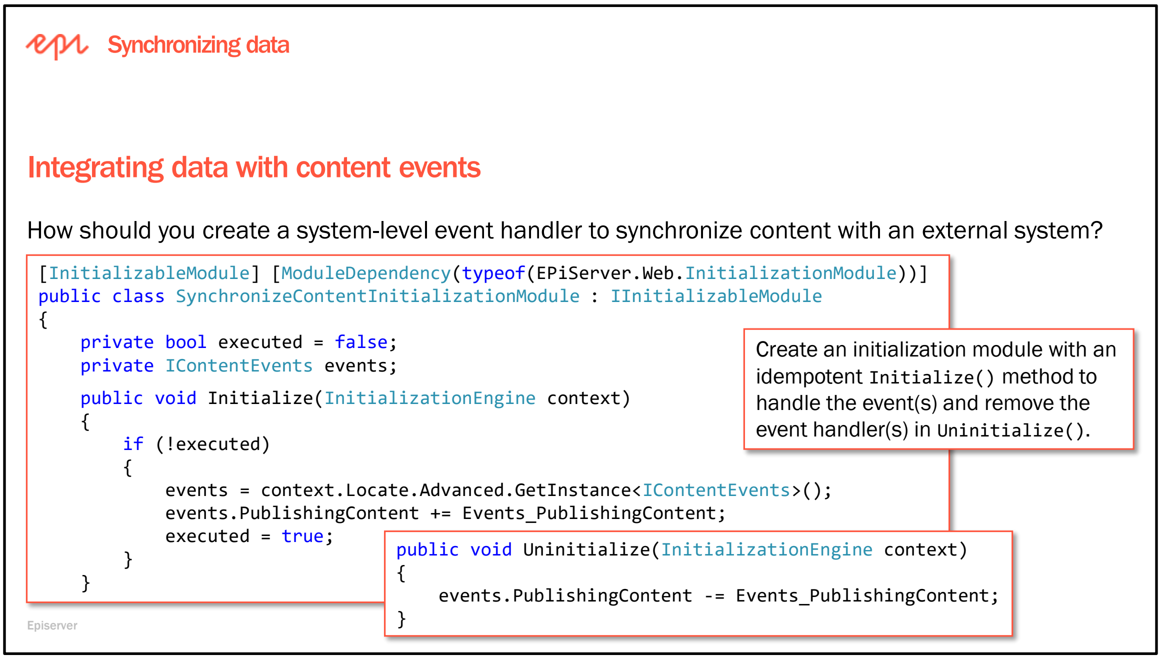
How should you test a scheduled job? Why?

- You should test the job by starting it manually and by setting it to start at a future time. This is because when started manually, the job will run with the security context of the logged in CMS Admin, but when started at a future time, the security context will be null.
- In the implementation of the Execute() method you should check the security context and create one if necessary for the job to run successfully:

```
public override string Execute()
{
    if (HttpContext.Current == null)
    {
        PrincipalInfo.CurrentPrincipal = new GenericPrincipal(
            new GenericIdentity("Scheduled Job Demo"),
            new[] { "Administrators, CmsAdmins" });
    }
```

Episerver

```
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!";
}
```



Synchronizing data

Integrating data with content events

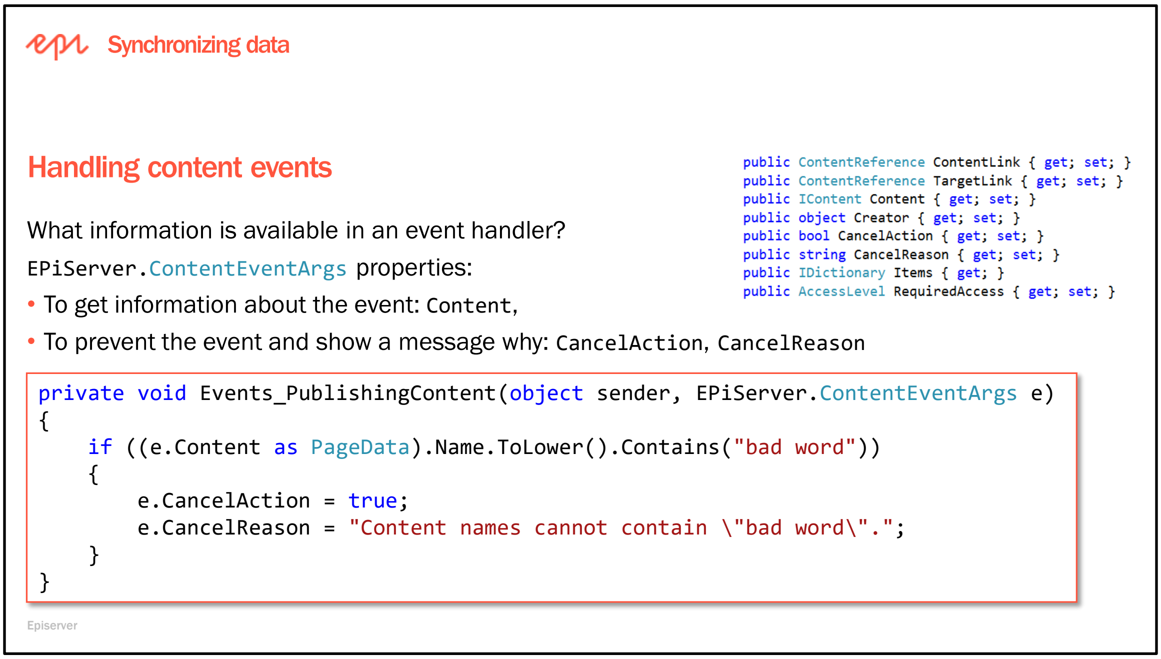
How should you create a system-level event handler to synchronize content with an external system?

```
[InitializableModule] [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
public class SynchronizeContentInitializationModule : IInitializableModule
{
    private bool executed = false;
    private IContentEvents events;

    public void Initialize(InitializationEngine context)
    {
        if (!executed)
        {
            events = context.Locate.Advanced.GetInstance<IContentEvents>();
            events.PublishingContent += Events_PublishingContent;
            executed = true;
        }
    }
```

Create an initialization module with an idempotent `Initialize()` method to handle the event(s) and remove the event handler(s) in `Uninitialize()`.

```
public void Uninitialize(InitializationEngine context)
{
    events.PublishingContent -= Events_PublishingContent;
}
```

Episerver



Synchronizing data

Handling content events

```
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; }
```

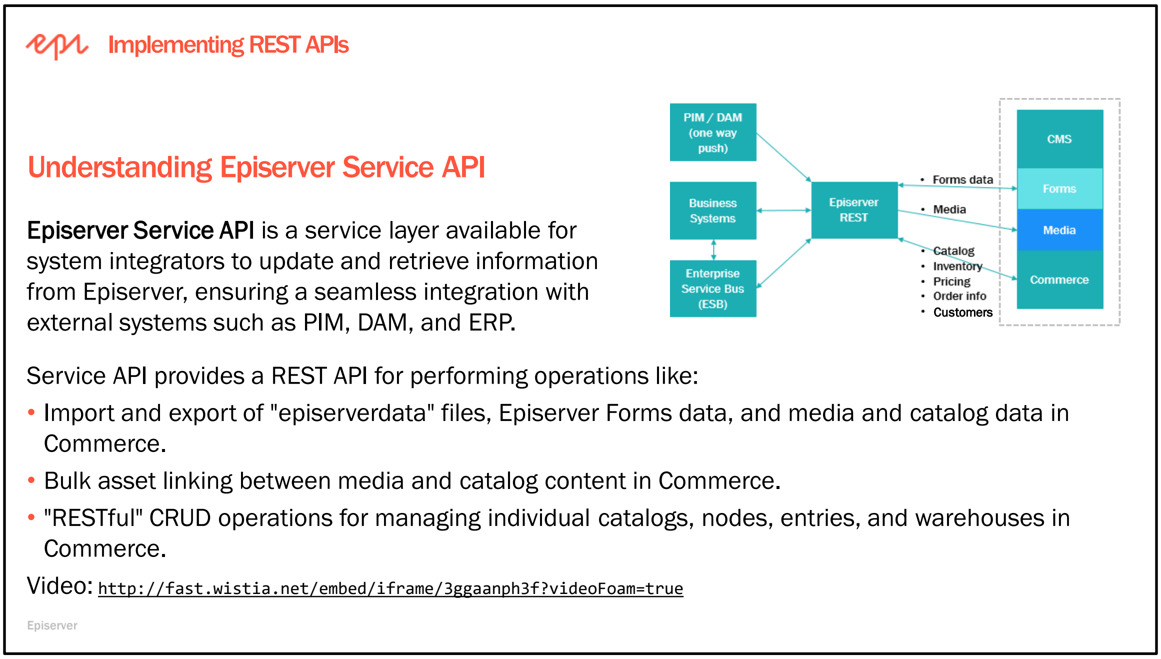
What information is available in an event handler?

`EPiServer.ContentEventArgs` properties:

- To get information about the event: `Content`,
- To prevent the event and show a message why: `CancelAction`, `CancelReason`

```
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



Implementing REST APIs

Understanding Episerver Service API

Episerver Service API is a service layer available for system integrators to update and retrieve information from Episerver, ensuring a seamless integration with external systems such as PIM, DAM, and ERP.

PIM / DAM (one way push)
Business Systems
Enterprise Service Bus (ESB)
Episerver REST
Forms data
Media
Catalog
Inventory
Pricing
Order info
Customers
CMS
Forms
Media
Commerce

Service API provides a REST API for performing operations like:

- Import and export of "episerverdata" files, Episerver Forms data, and media and catalog data in Commerce.
- Bulk asset linking between media and catalog content in Commerce.
- "RESTful" CRUD operations for managing individual catalogs, nodes, entries, and warehouses in Commerce.

Video: http://fast.wistia.net/embed/iframe/3ggaanph3f?videoFoam=true

Episerver

CMS content import/export service URLs

CMS site bulk import with file
`episerverapi/commerce/import/cms/site/{siteName}/{hostname}/{culture=}`

CMS site bulk import with file upload identifier
`episerverapi/commerce/import/cms/site/{siteName}/{hostname}/{uploadId:guid}/{culture=}`

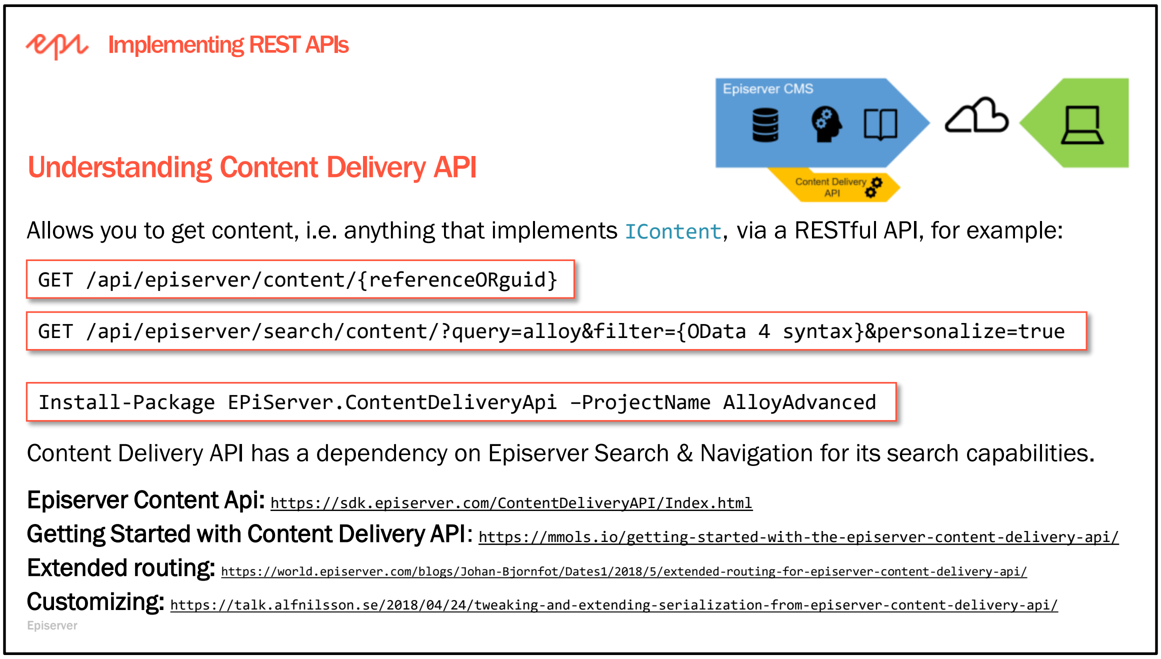
CMS assets bulk import with file
`episerverapi/commerce/import/cms/assetglobalroot`

CMS assets bulk import with file upload identifier
`episerverapi/commerce/import/cms/assetglobalroot/{uploadId:guid}`

CMS bulk export
`episerverapi/commerce/export/site/{siteName}`

Learn more

https://world.episerver.com/documentation/developer-guides/Episerver-Service-API/working-with-bulk-operations-using-tasks/cms-content-import-service/



Episerver CMS
Content Delivery API

Implementing REST APIs

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
```

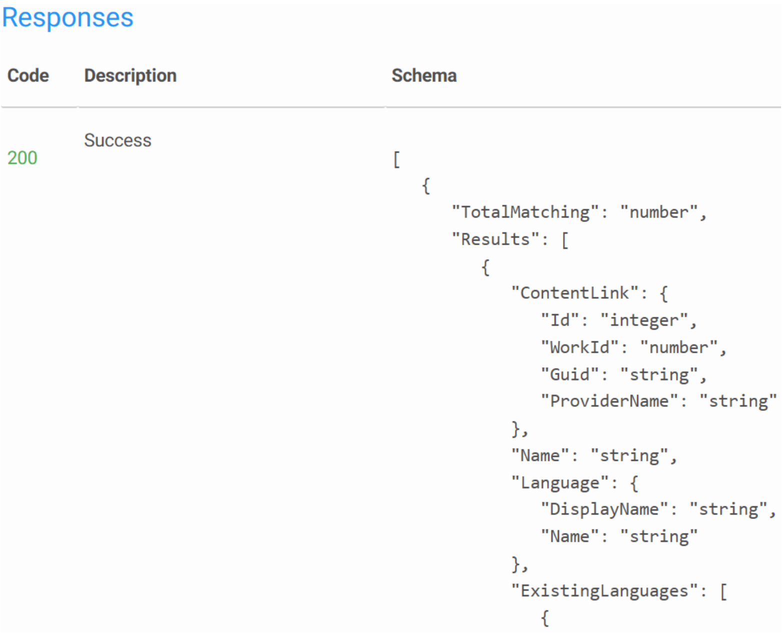
```
Install-Package EPiServer.ContentDeliveryApi -ProjectName AlloyAdvanced
```

Content Delivery API has a dependency on Episerver Search & Navigation for its search capabilities.

Episerver Content Api: https://sdk.episerver.com/ContentDeliveryAPI/Index.html
Getting Started with Content Delivery API: https://mmols.io/getting-started-with-the-episerver-content-delivery-api/
Extended routing: https://world.episerver.com/blogs/Johan-Bjornfot/Dates1/2018/5/extended-routing-for-episerver-content-delivery-api/
Customizing: https://talk.alfnilsson.se/2018/04/24/tweaking-and-extending-serialization-from-episerver-content-delivery-api/

Episerver

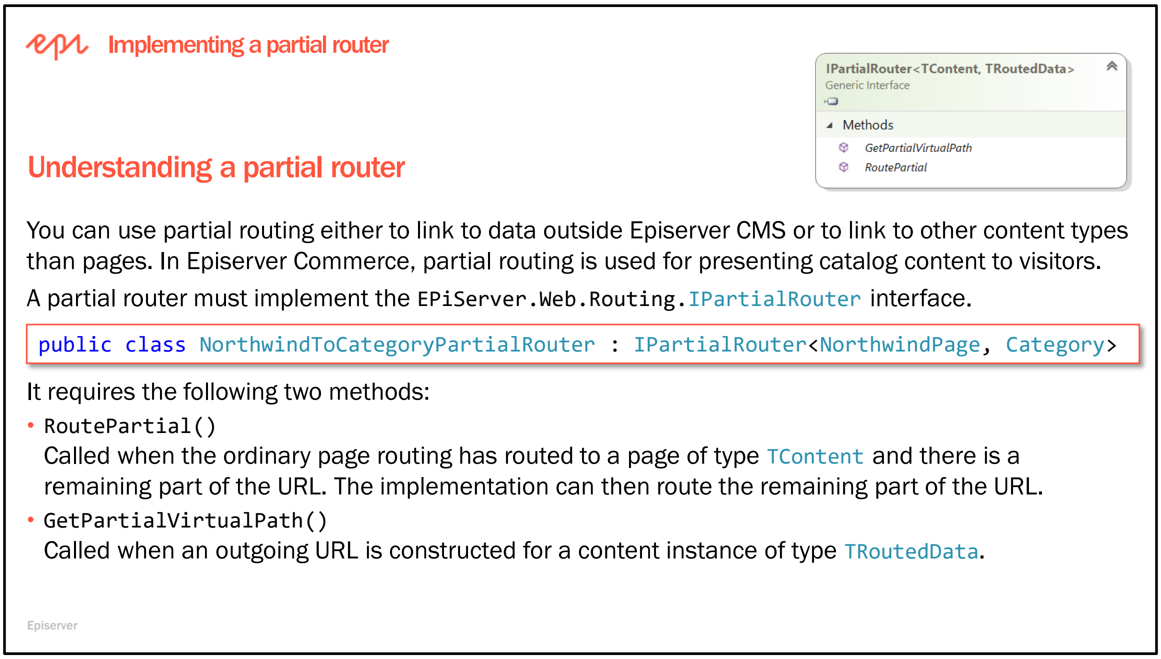
Content Delivery API
https://world.episerver.com/documentation/developer-guides/CMS/Content/content-delivery-api/
https://www.david-tec.com/2018/06/episerver-as-headless-episerver-ascend-2018-presentation/



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": [
          {
```


IPartialRouter<TContent, TRoutedData>
Generic Interface
Methods
GetPartialVirtualPath
RoutePartial

Implementing a partial router

Understanding a partial router

You can use partial routing either to link to data outside Episerver CMS or to link to other content types than pages. In Episerver Commerce, partial routing is used for presenting catalog content to visitors.

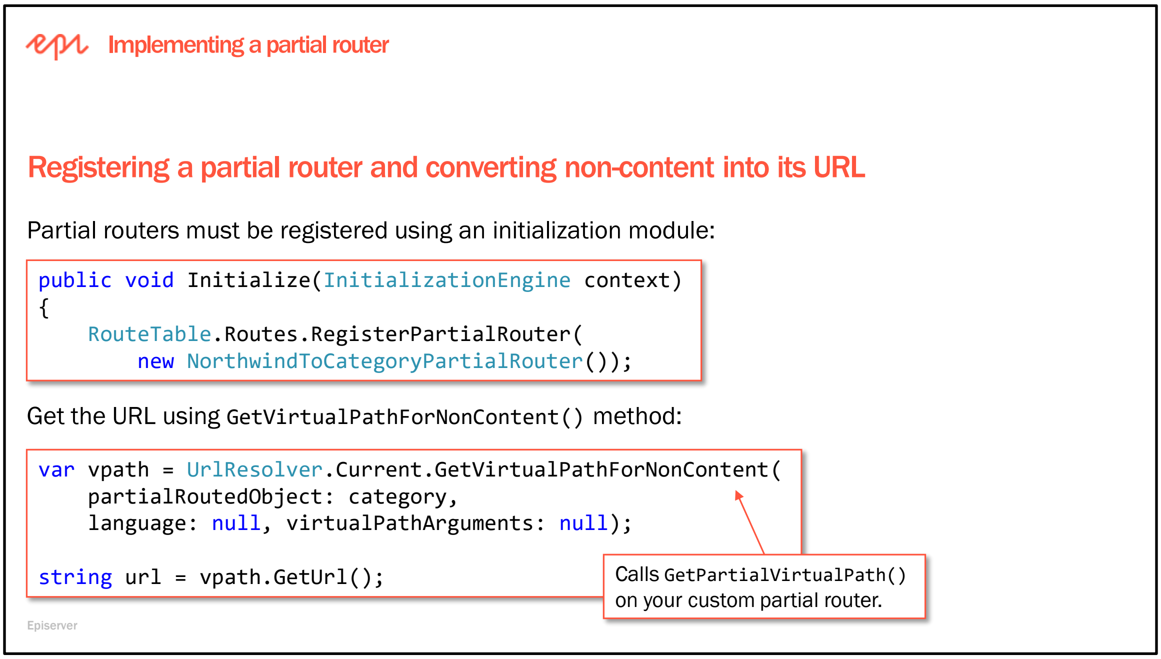
A partial router must implement the `EPiServer.Web.Routing.IPartialRouter` interface.

```
public class NorthwindToCategoryPartialRouter : IPartialRouter<NorthwindPage, Category>
```

It requires the following two methods:

- `RoutePartial()`
  Called when the ordinary page routing has routed to a page of type `TContent` and there is a remaining part of the URL. The implementation can then route the remaining part of the URL.
- `GetPartialVirtualPath()`
  Called when an outgoing URL is constructed for a content instance of type `TRoutedData`.

Episerver



Implementing a partial router

Registering a partial router and converting non-content into its URL

Partial routers must be registered using an initialization module:

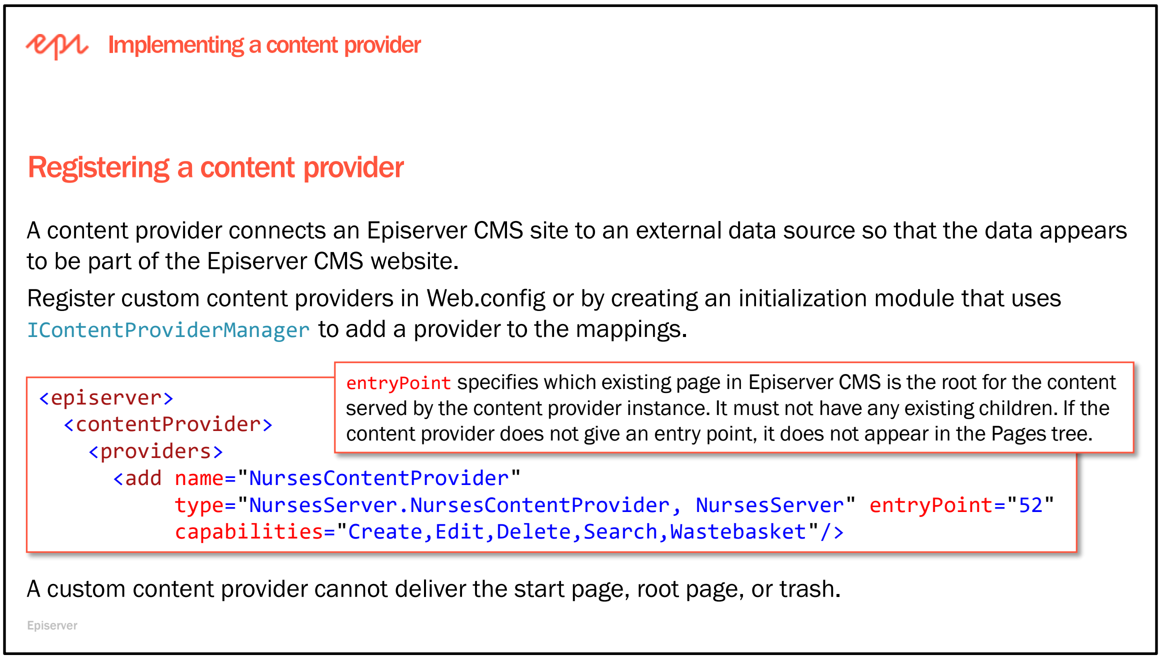
```
public void Initialize(InitializationEngine context)
{
    RouteTable.Routes.RegisterPartialRouter(
        new NorthwindToCategoryPartialRouter());
```

Get the URL using `GetVirtualPathForNonContent()` method:

```
var vpath = UrlResolver.Current.GetVirtualPathForNonContent(
    partialRoutedObject: category,
    language: null, virtualPathArguments: null);

string url = vpath.GetUrl();
```

Calls `GetPartialVirtualPath()` on your custom partial router.

Episerver



Implementing a content provider

Registering a content provider

A content provider connects an Episerver CMS site to an external data source so that the data appears to be part of the Episerver CMS website.

Register custom content providers in Web.config or by creating an initialization module that uses `IContentProviderManager` to add a provider to the mappings.

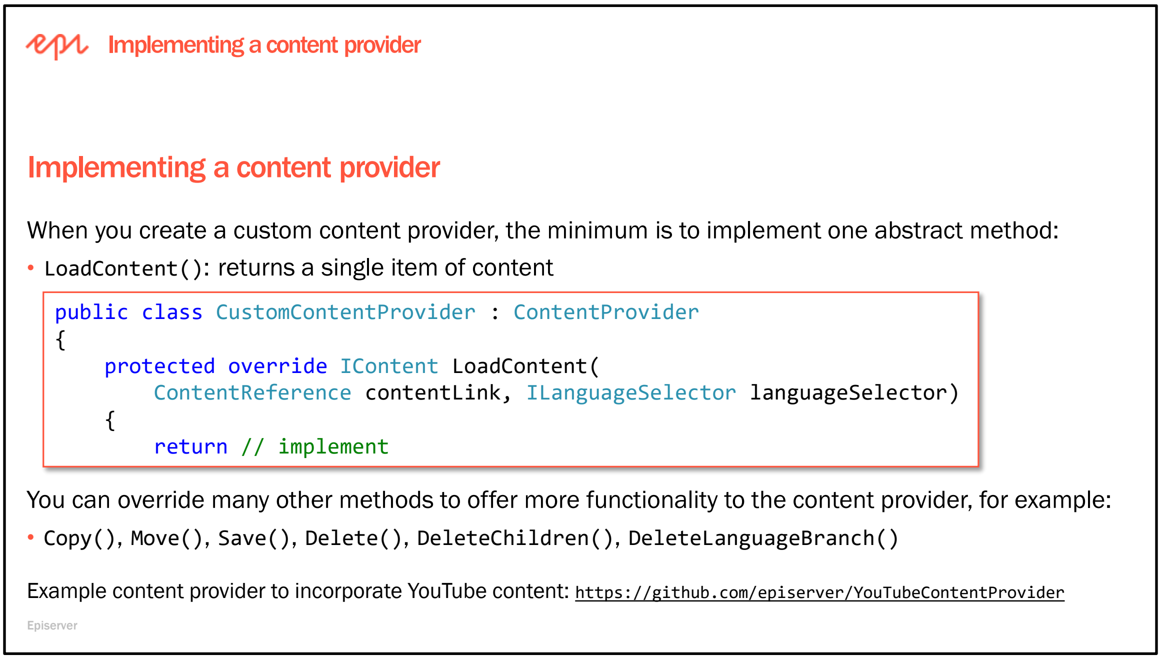
```
<episerver>
  <contentProvider>
    <providers>
      <add name="NursesContentProvider"
           type="NursesServer.NursesContentProvider, NursesServer" entryPoint="52"
           capabilities="Create,Edit,Delete,Search,Wastebasket"/>
```

`entryPoint` specifies which existing page in Episerver CMS is the root for the content served by the content provider instance. It must not have any existing children. If the content provider does not give an entry point, it does not appear in the Pages tree.

A custom content provider cannot deliver the start page, root page, or trash.

Episerver

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



Implementing a content provider

Implementing a content provider

When you create a custom content provider, the minimum is to implement one abstract method:

- `LoadContent()`: returns a single item of content

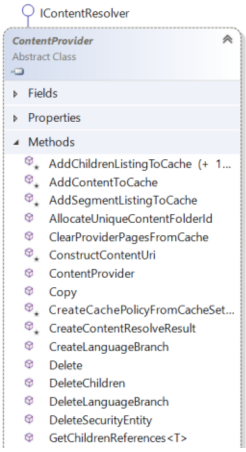
```
public class CustomContentProvider : ContentProvider
{
    protected override IContent LoadContent(
        ContentReference contentLink, ILanguageSelector languageSelector)
    {
        return // implement
```

You can override many other methods to offer more functionality to the content provider, for example:

- `Copy()`, `Move()`, `Save()`, `Delete()`, `DeleteChildren()`, `DeleteLanguageBranch()`

Example content provider to incorporate YouTube content: https://github.com/episerver/YouTubeContentProvider

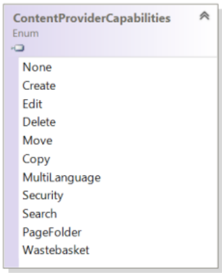
Episerver



IContentResolver

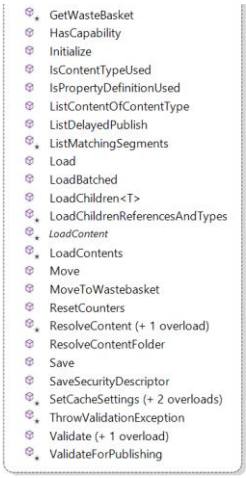
ContentProvider
Abstract Class

- Fields
- Properties
- Methods
  - AddChildrenListingToCache (+ 1...
  - AddContentToCache
  - AddSegmentListingToCache
  - AllocateUniqueContentFolderId
  - ClearProviderPagesFromCache
  - ConstructContentUri
  - ContentProvider
  - Copy
  - CreateCachePolicyFromCacheSet...
  - CreateContentResolveResult
  - CreateLanguageBranch
  - Delete
  - DeleteChildren
  - DeleteLanguageBranch
  - DeleteSecurityEntity
  - GetChildrenReferences<T>



ContentProviderCapabilities
Enum

- None
- Create
- Edit
- Delete
- Move
- Copy
- MultiLanguage
- Security
- Search
- PageFolder
- Wastebasket



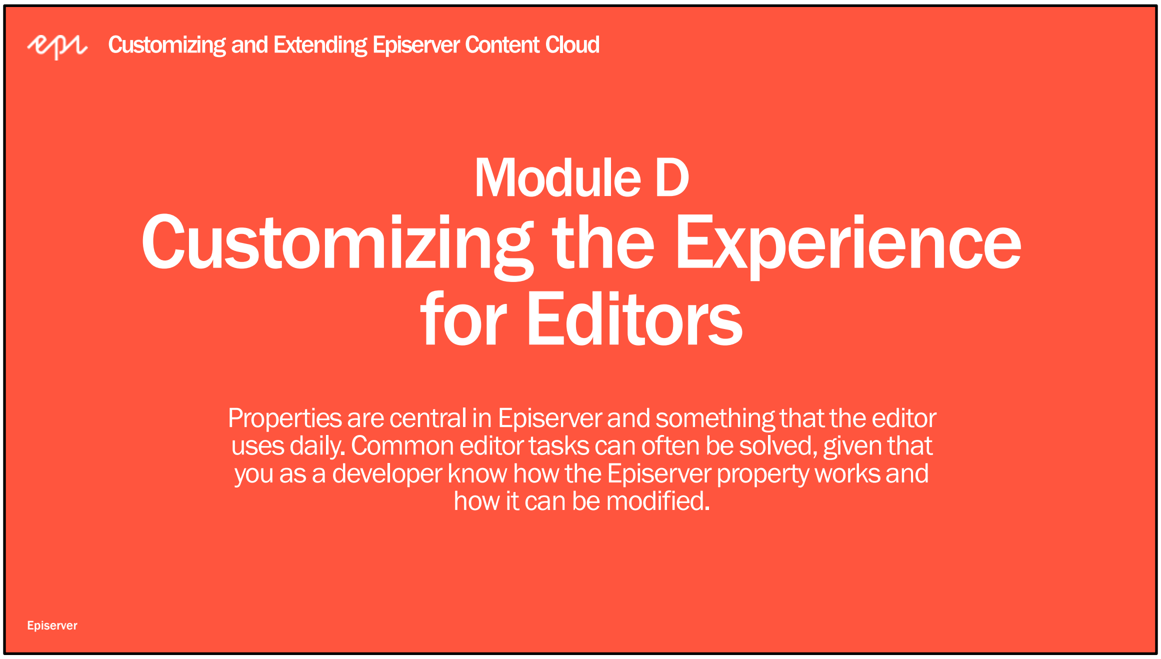
- GetWasteBasket
- HasCapability
- Initialize
- IsContentTypeUsed
- IsPropertyDefinitionUsed
- ListContentOfContentType
- ListDelayedPublish
- ListMatchingSegments
- Load
- LoadBatched
- LoadChildren<T>
- LoadChildrenReferencesAndTypes
- *LoadContent*
- LoadContents
- Move
- MoveToWastebasket
- ResetCounters
- ResolveContent (+ 1 overload)
- ResolveContentFolder
- Save
- SaveSecurityDescriptor
- SetCacheSettings (+ 2 overloads)
- ThrowValidationException
- Validate (+ 1 overload)
- ValidateForPublishing

Exercises C1 to C4

Integrating Data

1. Implementing favorite pages using DDS
2. Integrating external data using a partial router
3. Gathering data using Episerver Forms
4. Importing data using a scheduled job

Episerver

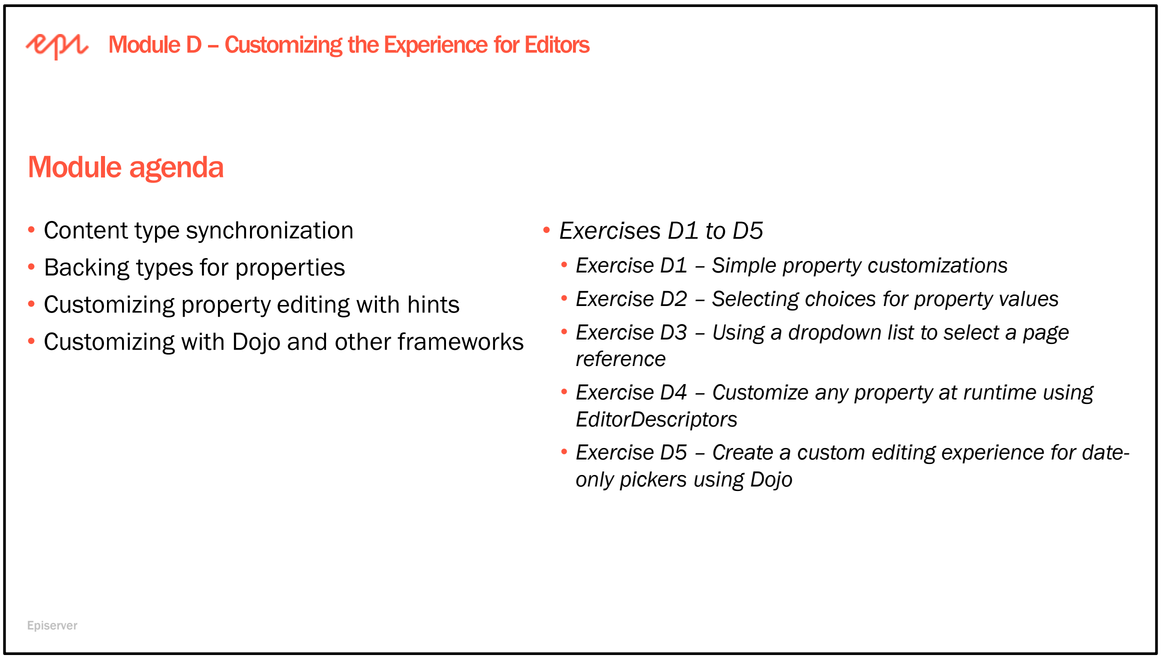


Customizing and Extending Episerver Content Cloud

Module D
Customizing the Experience for Editors

Properties are central in Episerver and something that the editor uses daily. Common editor tasks can often be solved, given that you as a developer know how the Episerver property works and how it can be modified.

Episerver



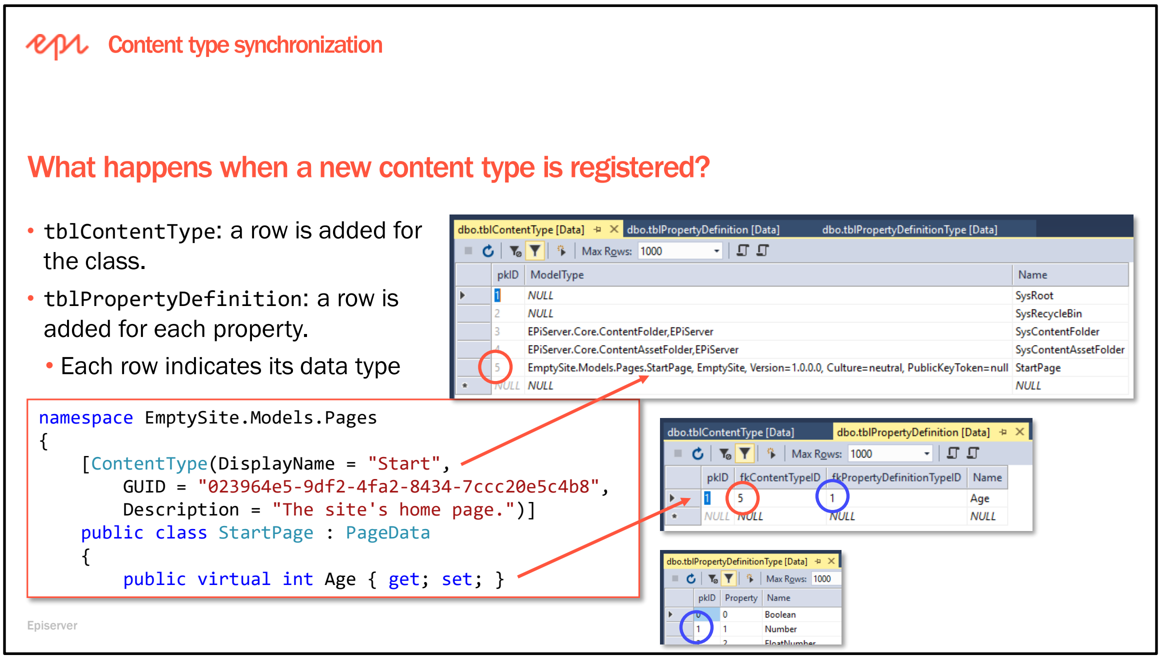
Module D – Customizing the Experience for Editors

Module agenda

- Content type synchronization
- Backing types for properties
- Customizing property editing with hints
- Customizing with Dojo and other frameworks

- *Exercises D1 to D5*
  - *Exercise D1 – Simple property customizations*
  - *Exercise D2 – Selecting choices for property values*
  - *Exercise D3 – Using a dropdown list to select a page reference*
  - *Exercise D4 – Customize any property at runtime using EditorDescriptors*
  - *Exercise D5 – Create a custom editing experience for date-only pickers using Dojo*

Episerver



Content type synchronization

What happens when a new content type is registered?

- `tblContentType`: a row is added for the class.
- `tblPropertyDefinition`: a row is added for each property.
  - Each row indicates its data type

```
namespace EmptySite.Models.Pages
{
    [ContentType(DisplayName = "Start",
        GUID = "023964e5-9df2-4fa2-8434-7ccc20e5c4b8",
        Description = "The site's home page.")]
    public class StartPage : PageData
    {
        public virtual int Age { get; set; }
```

dbo.tblContentType [Data]
dbo.tblPropertyDefinition [Data]
dbo.tblPropertyDefinitionType [Data]
Max Rows: 1000

pkID	ModelType	Name
1	NULL	SysRoot
2	NULL	SysRecycleBin
3	EPiServer.Core.ContentFolder,EPiServer	SysContentFolder
4	EPiServer.Core.ContentAssetFolder,EPiServer	SysContentAssetFolder
5	EmptySite.Models.Pages.StartPage, EmptySite, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null	StartPage
NULL	NULL	NULL

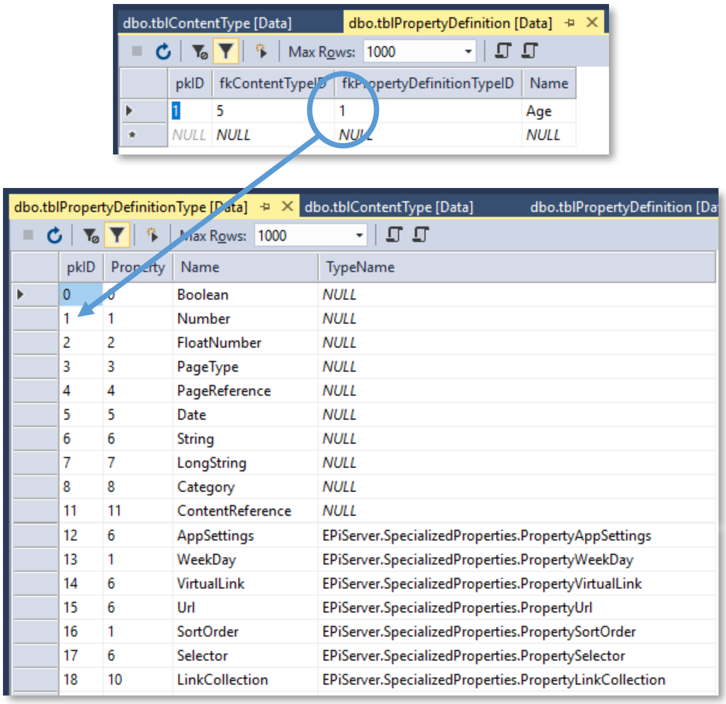
pkID	fkContentTypeID	fkPropertyDefinitionTypeID	Name
1	5	1	Age
NULL	NULL	NULL	NULL

pkID	Property	Name
0	0	Boolean
1	1	Number
2	2	FloatNumber

Episerver

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

Each row in the `tblPropertyDefinition` table relates to the data type of the property:



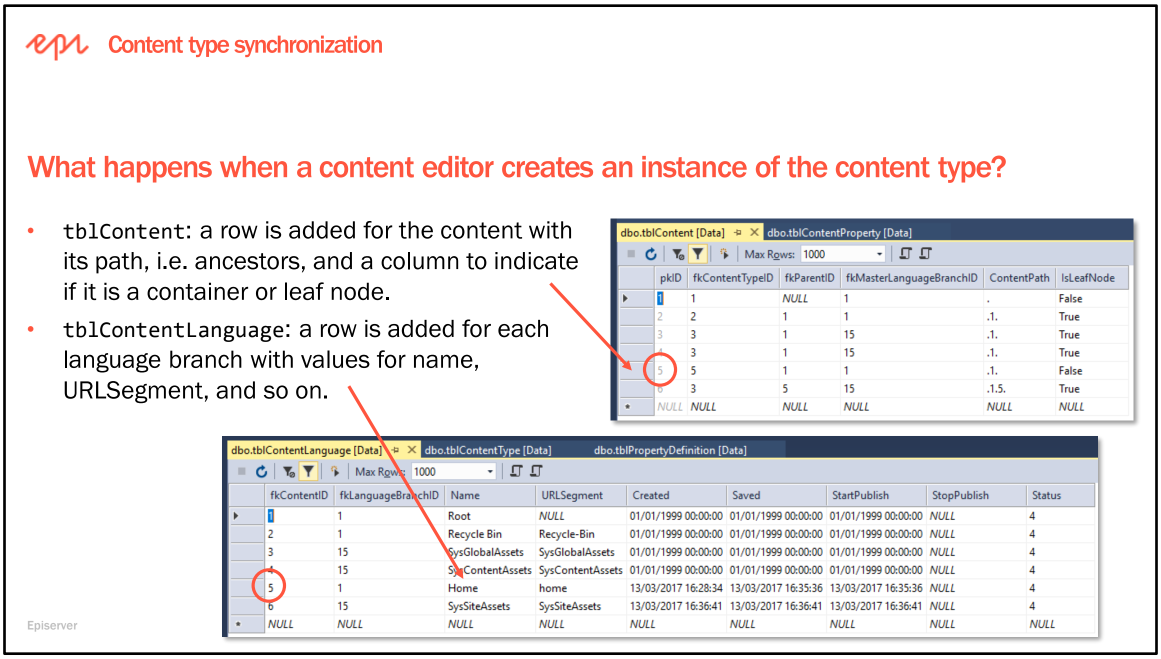
dbo.tblContentType [Data]
dbo.tblPropertyDefinition [Data]
Max Rows: 1000

pkID	fkContentTypeID	fkPropertyDefinitionTypeID	Name
1	5	1	Age
NULL	NULL	NULL	NULL

dbo.tblPropertyDefinitionType [Data]
dbo.tblContentType [Data]
dbo.tblPropertyDefinition [Da
Max Rows: 1000

pkID	Property	Name	TypeName
0	0	Boolean	NULL
1	1	Number	NULL
2	2	FloatNumber	NULL
3	3	PageType	NULL
4	4	PageReference	NULL
5	5	Date	NULL
6	6	String	NULL
7	7	LongString	NULL
8	8	Category	NULL
11	11	ContentReference	NULL
12	6	AppSettings	EPiServer.SpecializedProperties.PropertyAppSettings
13	1	WeekDay	EPiServer.SpecializedProperties.PropertyWeekDay
14	6	VirtualLink	EPiServer.SpecializedProperties.PropertyVirtualLink
15	6	Url	EPiServer.SpecializedProperties.PropertyUrl
16	1	SortOrder	EPiServer.SpecializedProperties.PropertySortOrder
17	6	Selector	EPiServer.SpecializedProperties.PropertySelector
18	10	LinkCollection	EPiServer.SpecializedProperties.PropertyLinkCollection

Property types like **AppSettings** and **Url** are stored as **String** types.



Content type synchronization

What happens when a content editor creates an instance of the content type?

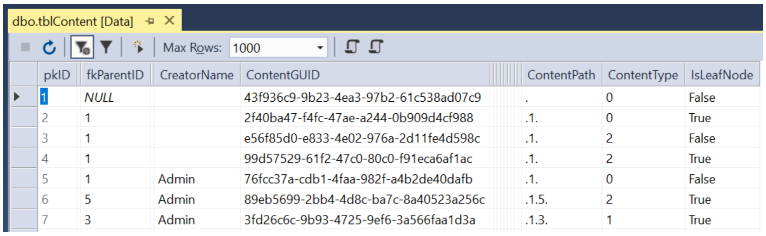
- `tblContent`: a row is added for the content with its path, i.e. ancestors, and a column to indicate if it is a container or leaf node.
- `tblContentLanguage`: a row is added for each language branch with values for name, URLSegment, and so on.

pkID	fkContentTypeID	fkParentID	fkMasterLanguageBranchID	ContentPath	IsLeafNode
1	1	NULL	1	.	False
2	2	1	1	.1.	True
3	3	1	15	.1.	True
4	3	1	15	.1.	True
5	5	1	1	.1.	False
6	3	5	15	.1.5.	True
NULL	NULL	NULL	NULL	NULL	NULL

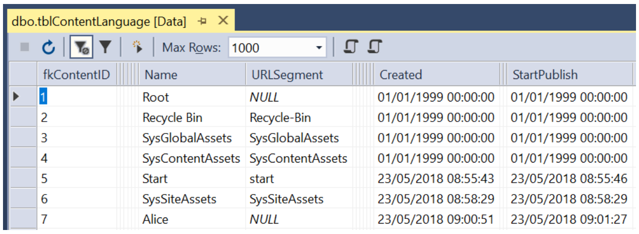
dbo.tblContentLanguage [Data]
dbo.tblContentType [Data]
dbo.tblPropertyDefinition [Data]
Max Rows: 1000

fkContentID	fkLanguageBranchID	Name	URLSegment	Created	Saved	StartPublish	StopPublish	Status
1	1	Root	NULL	01/01/1999 00:00:00	01/01/1999 00:00:00	01/01/1999 00:00:00	NULL	4
2	1	Recycle Bin	Recycle-Bin	01/01/1999 00:00:00	01/01/1999 00:00:00	01/01/1999 00:00:00	NULL	4
3	15	SysGlobalAssets	SysGlobalAssets	01/01/1999 00:00:00	01/01/1999 00:00:00	01/01/1999 00:00:00	NULL	4
4	15	SysContentAssets	SysContentAssets	01/01/1999 00:00:00	01/01/1999 00:00:00	01/01/1999 00:00:00	NULL	4
5	1	Home	home	13/03/2017 16:28:34	13/03/2017 16:35:36	13/03/2017 16:35:36	NULL	4
6	15	SysSiteAssets	SysSiteAssets	13/03/2017 16:36:41	13/03/2017 16:36:41	13/03/2017 16:36:41	NULL	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

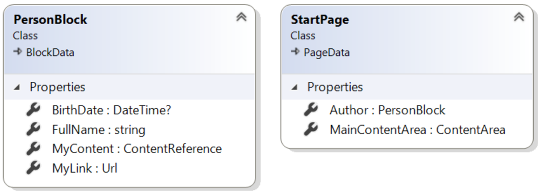
Episerver



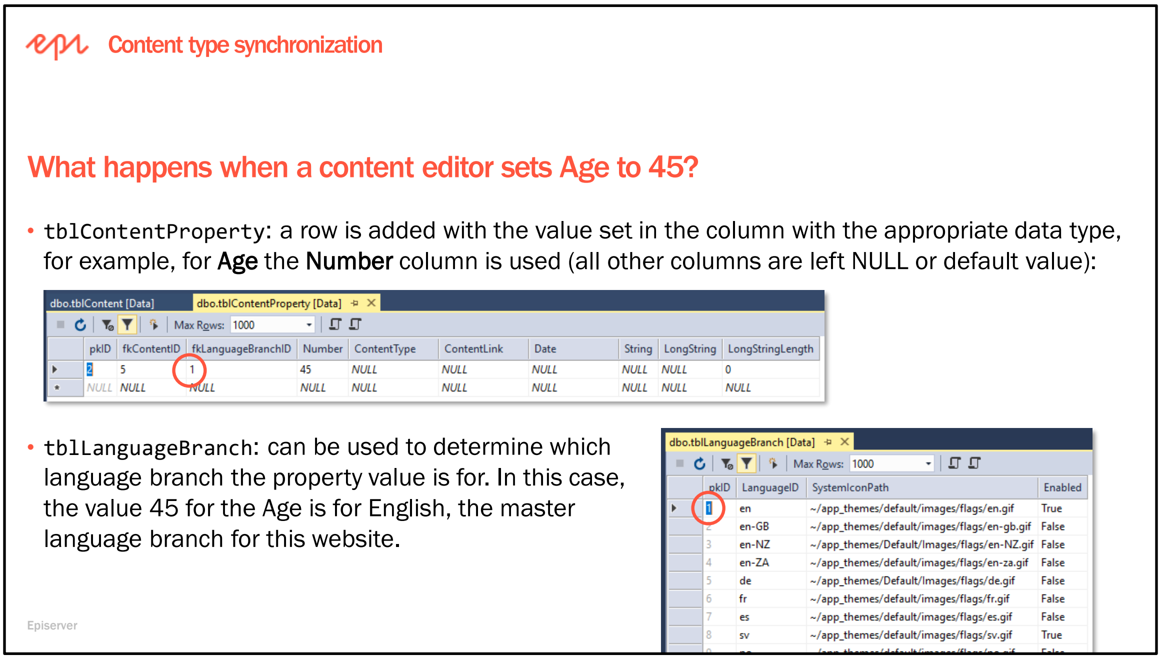
pkID	fkParentID	CreatorName	ContentGUID	ContentPath	ContentType	IsLeafNode
1	NULL		43f936c9-9b23-4ea3-97b2-61c538ad07c9	.	0	False
2	1		2f40ba47-f4fc-47ae-a244-0b909d4cf988	.1.	0	True
3	1		e56f85d0-e833-4e02-976a-2d11fe4d598c	.1.	2	False
4	1		99d57529-61f2-47c0-80c0-f91eca6af1ac	.1.	2	True
5	1	Admin	76fcc37a-cdb1-4faa-982f-a4b2de40dafb	.1.	0	False
6	5	Admin	89eb5699-2bb4-4d8c-ba7c-8a40523a256c	.1.5.	2	True
7	3	Admin	3fd26c6c-9b93-4725-9ef6-3a566faa1d3a	.1.3.	1	True



fkContentID	Name	URLSegment	Created	StartPublish
1	Root	NULL	01/01/1999 00:00:00	01/01/1999 00:00:00
2	Recycle Bin	Recycle-Bin	01/01/1999 00:00:00	01/01/1999 00:00:00
3	SysGlobalAssets	SysGlobalAssets	01/01/1999 00:00:00	01/01/1999 00:00:00
4	SysContentAssets	SysContentAssets	01/01/1999 00:00:00	01/01/1999 00:00:00
5	Start	start	23/05/2018 08:55:43	23/05/2018 08:55:46
6	SysSiteAssets	SysSiteAssets	23/05/2018 08:58:29	23/05/2018 08:58:29
7	Alice	NULL	23/05/2018 09:00:51	23/05/2018 09:01:27



How is a property block stored in the CMS database?
How is a shared asset block stored in the CMS database?



Content type synchronization

What happens when a content editor sets Age to 45?

- `tblContentProperty`: a row is added with the value set in the column with the appropriate data type, for example, for **Age** the **Number** column is used (all other columns are left NULL or default value):

dbo.tblContent [Data] | dbo.tblContentProperty [Data]

Max Rows: 1000

pkID	fkContentID	fkLanguageBranchID	Number	ContentType	ContentLink	Date	String	LongString	LongStringLength
2	5	1	45	NULL	NULL	NULL	NULL	NULL	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- `tblLanguageBranch`: can be used to determine which language branch the property value is for. In this case, the value 45 for the Age is for English, the master language branch for this website.

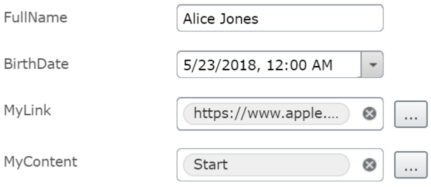
dbo.tblLanguageBranch [Data]

Max Rows: 1000

pkID	LanguageID	SystemIconPath	Enabled
1	en	~/app_themes/default/images/flags/en.gif	True
2	en-GB	~/app_themes/default/images/flags/en-gb.gif	False
3	en-NZ	~/app_themes/Default/Images/flags/en-NZ.gif	False
4	en-ZA	~/app_themes/default/images/flags/en-za.gif	False
5	de	~/app_themes/Default/Images/flags/de.gif	False
6	fr	~/app_themes/default/images/flags/fr.gif	False
7	es	~/app_themes/default/images/flags/es.gif	False
8	sv	~/app_themes/default/images/flags/sv.gif	True

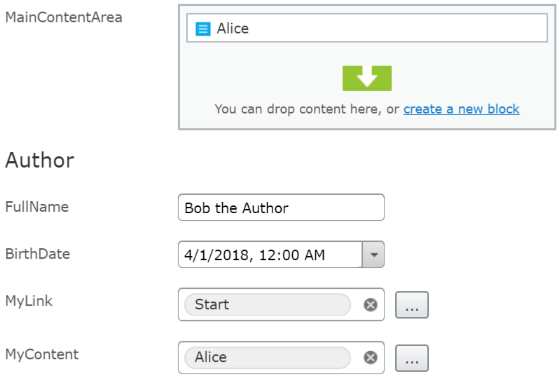
Episerver

An instance of PersonBlock named Alice (Alice Jones, 22 May 2018, Apple home page, content reference to Start).



FullName	Alice Jones
BirthDate	5/23/2018, 12:00 AM
MyLink	https://www.apple....
MyContent	Start

An instance of StartPage named Start with Bob as Author.



MainContentArea: Alice

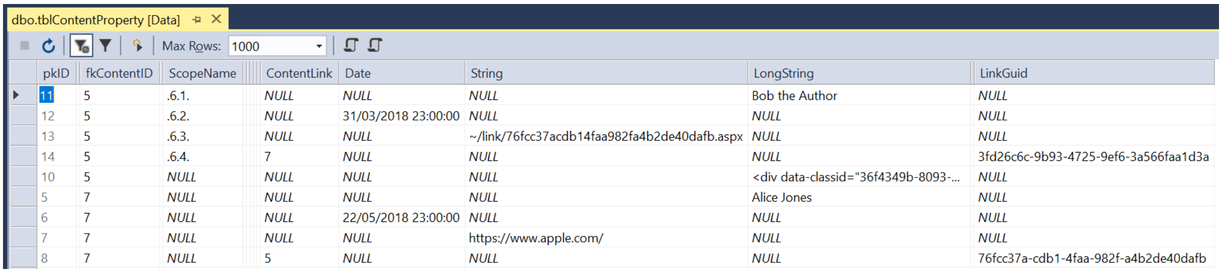
You can drop content here, or create a new block

Author

FullName	Bob the Author
BirthDate	4/1/2018, 12:00 AM
MyLink	Start
MyContent	Alice

Rows in tblContentProperties

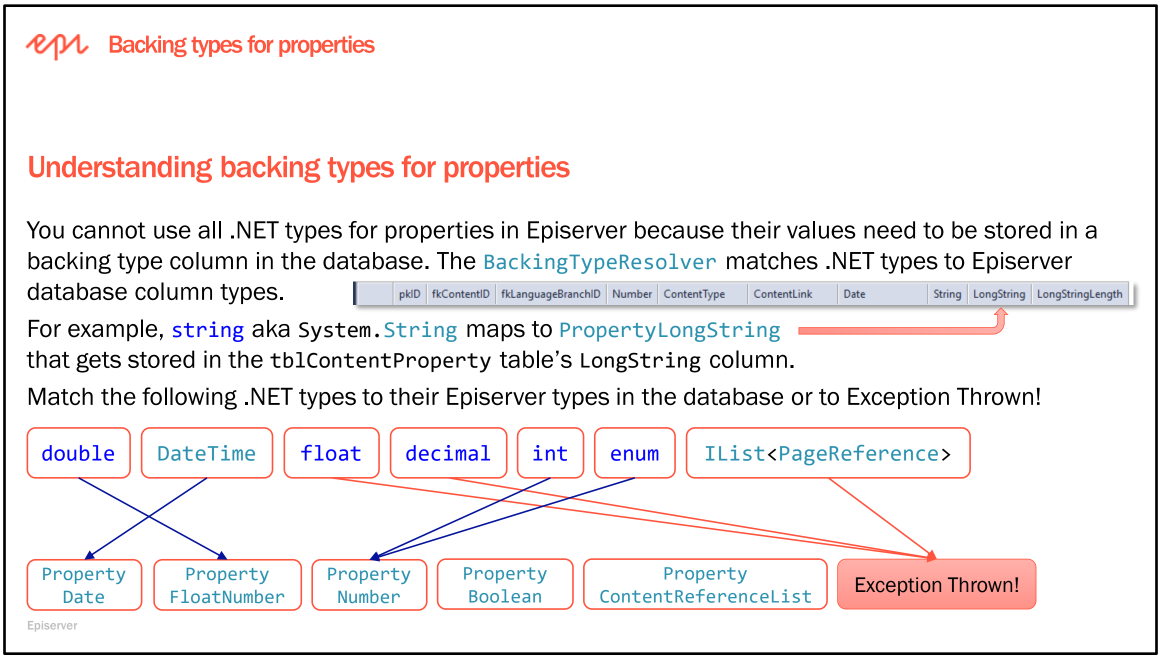
Note the Author property's four properties have been stored as separate rows as if the properties belonged to StartPage itself:



dbo.tblContentProperty [Data]

Max Rows: 1000

pkID	fkContentID	ScopeName	ContentLink	Date	String	LongString	LinkGuid
11	5	.6.1.	NULL	NULL	NULL	Bob the Author	NULL
12	5	.6.2.	NULL	31/03/2018 23:00:00	NULL	NULL	NULL
13	5	.6.3.	NULL	NULL	~/link/76fcc37acdb14faa982fa4b2de40dafb.aspx	NULL	NULL
14	5	.6.4.	7	NULL	NULL	NULL	3fd26c6c-9b93-4725-9ef6-3a566faa1d3a
10	5	NULL	NULL	NULL	NULL	<div data-classid="36f4349b-8093-...	NULL
5	7	NULL	NULL	NULL	NULL	Alice Jones	NULL
6	7	NULL	NULL	22/05/2018 23:00:00	NULL	NULL	NULL
7	7	NULL	NULL	NULL	https://www.apple.com/	NULL	NULL
8	7	NULL	5	NULL	NULL	NULL	76fcc37a-cdb1-4faa-982f-a4b2de40dafb



Backing types for properties

Understanding backing types for properties

You cannot use all .NET types for properties in Episerver because their values need to be stored in a backing type column in the database. The `BackingTypeResolver` matches .NET types to Episerver database column types.

For example, `string` aka `System.String` maps to `PropertyLongString` that gets stored in the `tblContentProperty` table's `LongString` column.

Match the following .NET types to their Episerver types in the database or to Exception Thrown!

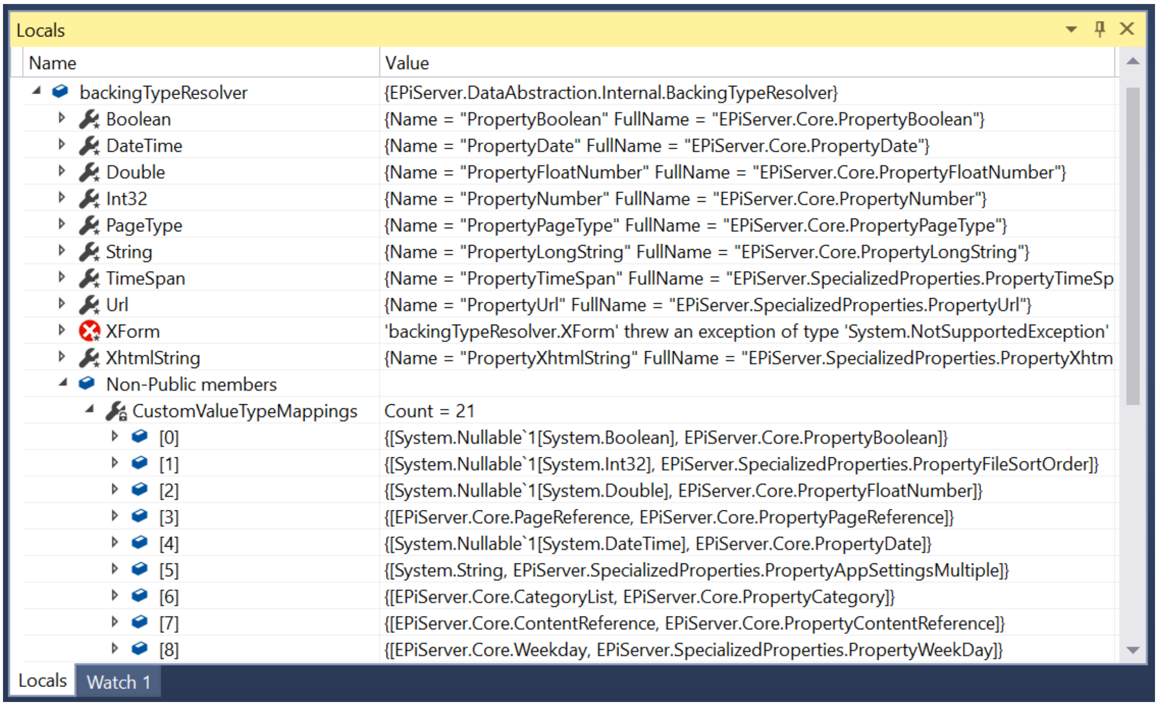
`double` `DateTime` `float` `decimal` `int` `enum` `IList<PageReference>`

`Property Date` `Property FloatNumber` `Property Number` `Property Boolean` `Property ContentReferenceList` Exception Thrown!

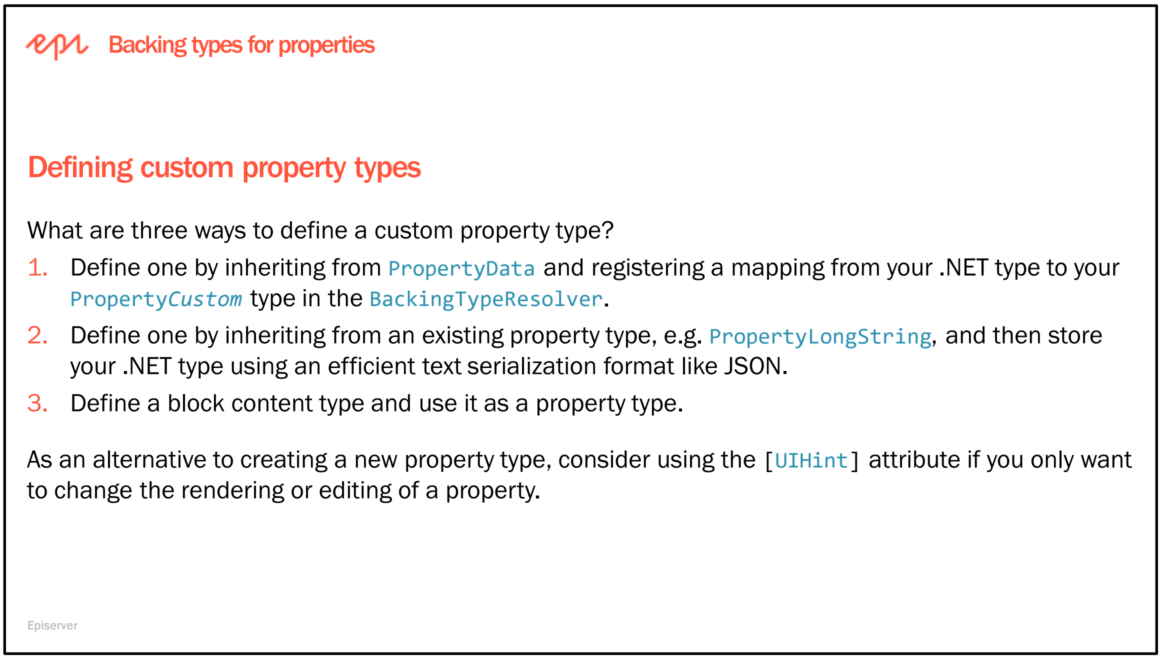
Episerver

If you want to use a type without a registered backing type, and that type can be converted into a simpler type, for example enums can be converted into integers and strings, then you can apply the [BackingType] attribute to specify how to store and type in the CMS database:

```
[BackingType(typeof(PropertyNumber))]
[UIHint("SortOrder")]
[DefaultValue(FilterSortOrder.PublishedDescending)]
public virtual FilterSortOrder SortOrder { get; set; }
```



```
Locals
Name                              Value
◢ backingTypeResolver             {EPiServer.DataAbstraction.Internal.BackingTypeResolver}
  ▷ Boolean                       {Name = "PropertyBoolean" FullName = "EPiServer.Core.PropertyBoolean"}
  ▷ DateTime                      {Name = "PropertyDate" FullName = "EPiServer.Core.PropertyDate"}
  ▷ Double                        {Name = "PropertyFloatNumber" FullName = "EPiServer.Core.PropertyFloatNumber"}
  ▷ Int32                         {Name = "PropertyNumber" FullName = "EPiServer.Core.PropertyNumber"}
  ▷ PageType                      {Name = "PropertyPageType" FullName = "EPiServer.Core.PropertyPageType"}
  ▷ String                        {Name = "PropertyLongString" FullName = "EPiServer.Core.PropertyLongString"}
  ▷ TimeSpan                      {Name = "PropertyTimeSpan" FullName = "EPiServer.SpecializedProperties.PropertyTimeSp
  ▷ Url                           {Name = "PropertyUrl" FullName = "EPiServer.SpecializedProperties.PropertyUrl"}
  ▷ XForm                         'backingTypeResolver.XForm' threw an exception of type 'System.NotSupportedException'
  ▷ XhtmlString                   {Name = "PropertyXhtmlString" FullName = "EPiServer.SpecializedProperties.PropertyXhtm
  ◢ Non-Public members
    ◢ CustomValueTypeMappings     Count = 21
      ▷ [0]                       {[System.Nullable`1[System.Boolean], EPiServer.Core.PropertyBoolean]}
      ▷ [1]                       {[System.Nullable`1[System.Int32], EPiServer.SpecializedProperties.PropertyFileSortOrder]}
      ▷ [2]                       {[System.Nullable`1[System.Double], EPiServer.Core.PropertyFloatNumber]}
      ▷ [3]                       {[EPiServer.Core.PageReference, EPiServer.Core.PropertyPageReference]}
      ▷ [4]                       {[System.Nullable`1[System.DateTime], EPiServer.Core.PropertyDate]}
      ▷ [5]                       {[System.String, EPiServer.SpecializedProperties.PropertyAppSettingsMultiple]}
      ▷ [6]                       {[EPiServer.Core.CategoryList, EPiServer.Core.PropertyCategory]}
      ▷ [7]                       {[EPiServer.Core.ContentReference, EPiServer.Core.PropertyContentReference]}
      ▷ [8]                       {[EPiServer.Core.Weekday, EPiServer.SpecializedProperties.PropertyWeekDay]}
Locals  Watch 1
```



Backing types for properties

Defining custom property types

What are three ways to define a custom property type?

1. Define one by inheriting from `PropertyData` and registering a mapping from your .NET type to your `PropertyCustom` type in the `BackingTypeResolver`.
2. Define one by inheriting from an existing property type, e.g. `PropertyLongString`, and then store your .NET type using an efficient text serialization format like JSON.
3. Define a block content type and use it as a property type.

As an alternative to creating a new property type, consider using the `[UIHint]` attribute if you only want to change the rendering or editing of a property.

Episerver

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

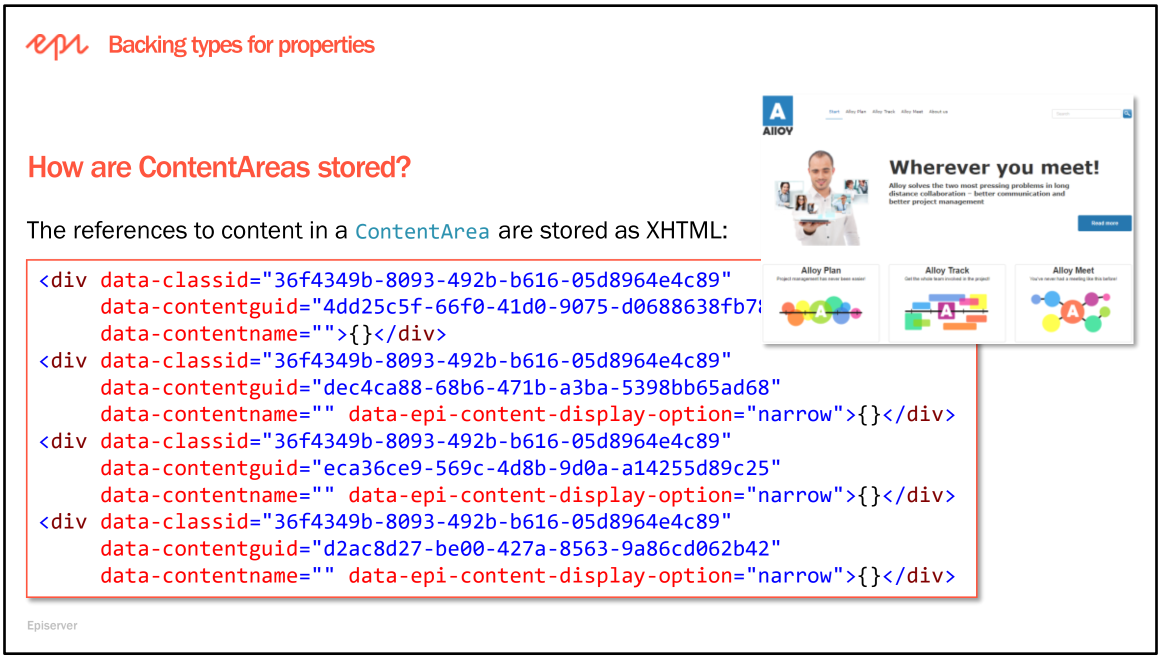
More information:

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/



Backing types for properties

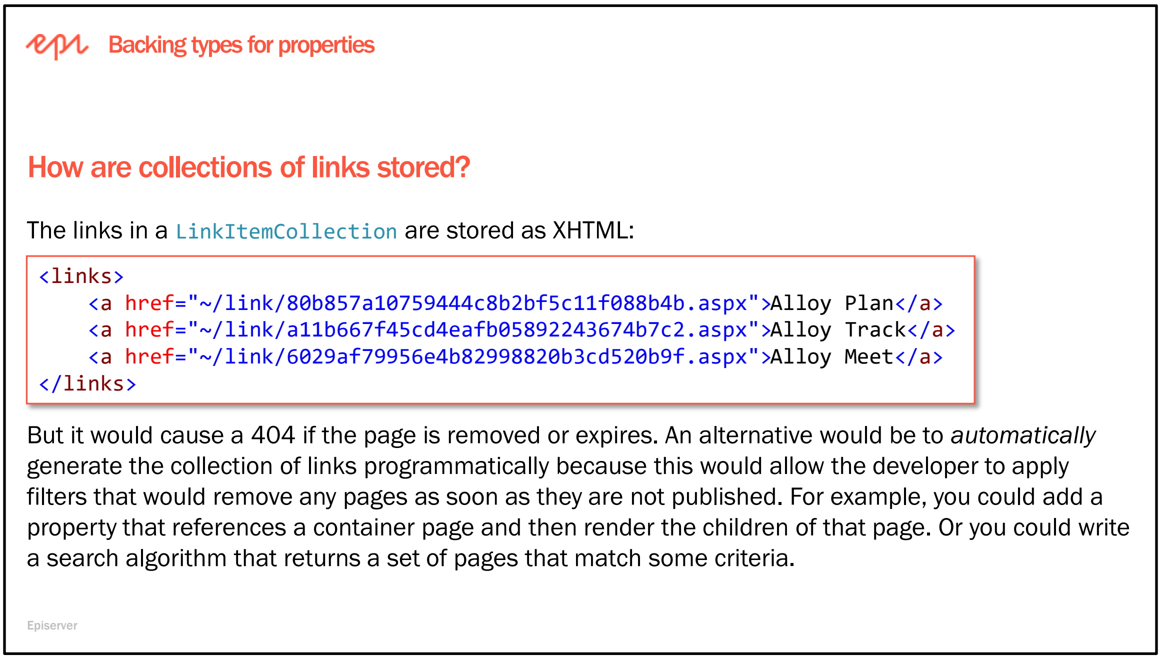
How are ContentAreas stored?

The references to content in a `ContentArea` are stored as XHTML:

```
<div data-classid="36f4349b-8093-492b-b616-05d8964e4c89"
     data-contentguid="4dd25c5f-66f0-41d0-9075-d0688638fb7
     data-contentname="">{}</div>
<div data-classid="36f4349b-8093-492b-b616-05d8964e4c89"
     data-contentguid="dec4ca88-68b6-471b-a3ba-5398bb65ad68"
     data-contentname="" data-epi-content-display-option="narrow">{}</div>
<div data-classid="36f4349b-8093-492b-b616-05d8964e4c89"
     data-contentguid="eca36ce9-569c-4d8b-9d0a-a14255d89c25"
     data-contentname="" data-epi-content-display-option="narrow">{}</div>
<div data-classid="36f4349b-8093-492b-b616-05d8964e4c89"
     data-contentguid="d2ac8d27-be00-427a-8563-9a86cd062b42"
     data-contentname="" data-epi-content-display-option="narrow">{}</div>
```

Episerver

Alloy
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!
Alloy Track
Get the whole team involved in the project!
Alloy Meet
You've never had a meeting like this before!



Backing types for properties

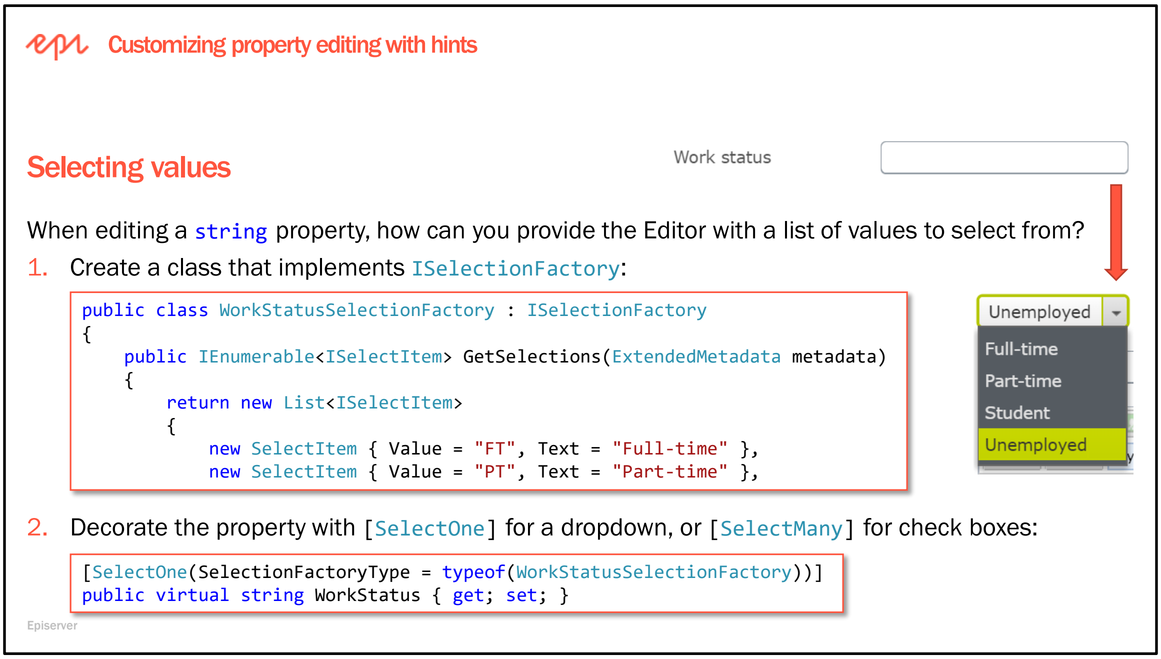
How are collections of links stored?

The links in a `LinkItemCollection` are stored as XHTML:

```
<links>
    <a href="~/link/80b857a10759444c8b2bf5c11f088b4b.aspx">Alloy Plan</a>
    <a href="~/link/a11b667f45cd4eafb05892243674b7c2.aspx">Alloy Track</a>
    <a href="~/link/6029af79956e4b82998820b3cd520b9f.aspx">Alloy Meet</a>
</links>
```

But it would cause a 404 if the page is removed or expires. An alternative would be to *automatically* generate the collection of links programmatically because this would allow the developer to apply filters that would remove any pages as soon as they are not published. For example, you could add a property that references a container page and then render the children of that page. Or you could write a search algorithm that returns a set of pages that match some criteria.

Episerver



Customizing property editing with hints

Selecting values

When editing a `string` property, how can you provide the Editor with a list of values to select from?

1. Create a class that implements `ISelectionFactory`:

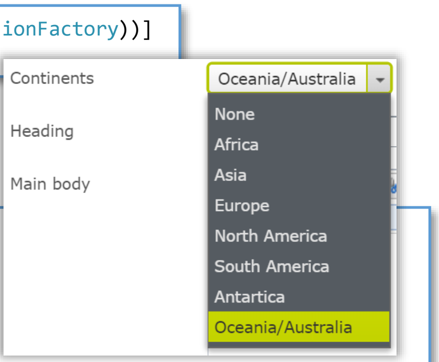
```
public class WorkStatusSelectionFactory : ISelectionFactory
{
    public IEnumerable<ISelectItem> GetSelections(ExtendedMetadata metadata)
    {
        return new List<ISelectItem>
        {
            new SelectItem { Value = "FT", Text = "Full-time" },
            new SelectItem { Value = "PT", Text = "Part-time" },
```

2. Decorate the property with `[SelectOne]` for a dropdown, or `[SelectMany]` for check boxes:

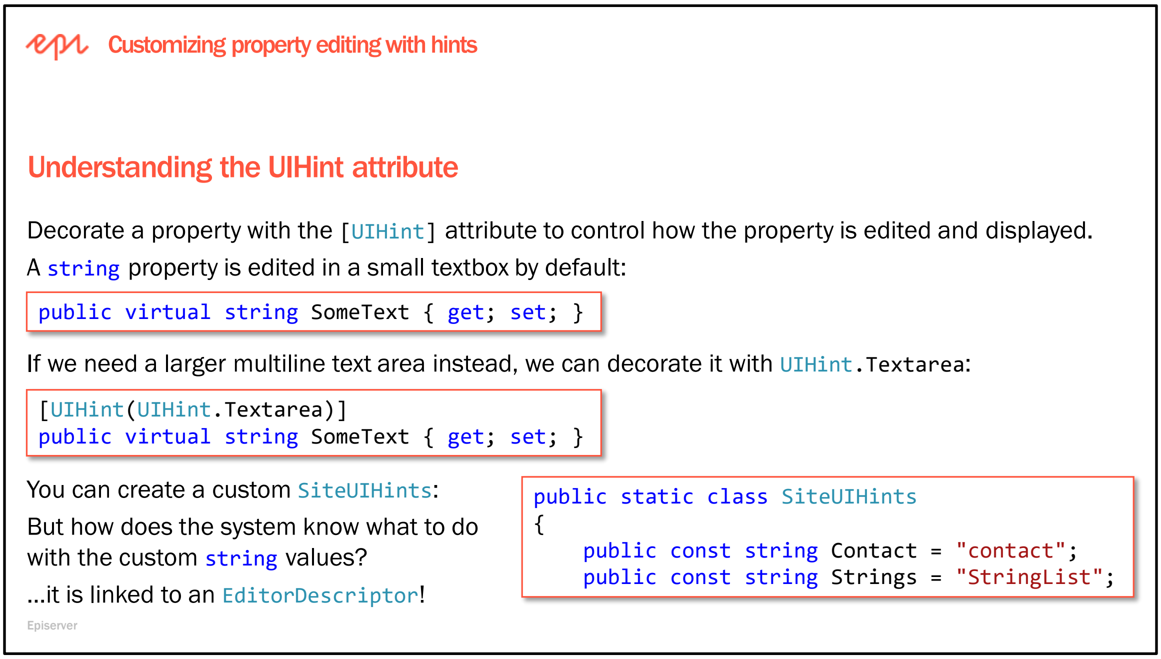
```
[SelectOne(SelectionFactoryType = typeof(WorkStatusSelectionFactory))]
public virtual string WorkStatus { get; set; }
```

Episerver

```
[SelectOne(SelectionFactoryType = typeof(ContinentsSelectionFactory))]
public virtual Continents Continents { get; set; }
```



```
using EPiServer.Shell.ObjectEditing;
using System.Collections.Generic;

namespace AlloyTraining.Business.SelectionFactories
{
    public enum Continents
    {
        None, Africa, Asia, Europe, NorthAmerica, SouthAmerica, Antartica, Oceania
    }

    public class ContinentsSelectionFactory : ISelectionFactory
    {
        public IEnumerable<ISelectItem> GetSelections(ExtendedMetadata metadata)
        {
            return new List<SelectItem>
            {
                new SelectItem { Value = Continents.None, Text = "None" },
                new SelectItem { Value = Continents.Africa, Text = "Africa" },
                new SelectItem { Value = Continents.Asia, Text = "Asia" },
                new SelectItem { Value = Continents.Europe, Text = "Europe" },
                new SelectItem { Value = Continents.NorthAmerica, Text = "North America" },
                new SelectItem { Value = Continents.SouthAmerica, Text = "South America" },
                new SelectItem { Value = Continents.Antartica, Text = "Antartica" },
                new SelectItem { Value = Continents.Oceania, Text = "Oceania/Australia" }
            };
        }
    }
}
```



Customizing property editing with hints

Understanding the UIHint attribute

Decorate a property with the `[UIHint]` attribute to control how the property is edited and displayed.

A `string` property is edited in a small textbox by default:

```
public virtual string SomeText { get; set; }
```

If we need a larger multiline text area instead, we can decorate it with `UIHint.Textarea`:

```
[UIHint(UIHint.Textarea)]
public virtual string SomeText { get; set; }
```

You can create a custom `SiteUIHints`:

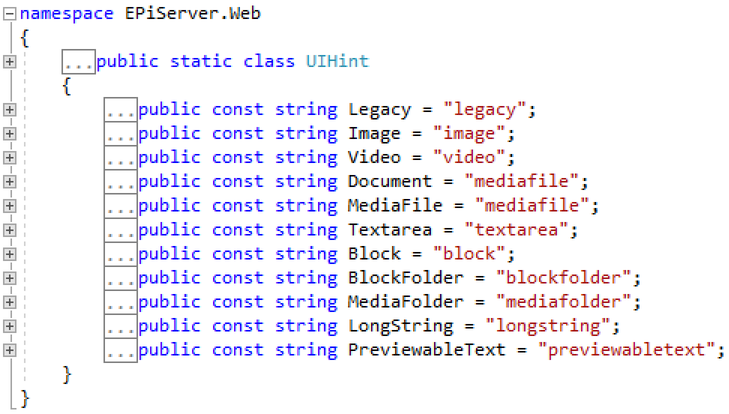
```
public static class SiteUIHints
{
    public const string Contact = "contact";
    public const string Strings = "StringList";
```

But how does the system know what to do with the custom `string` values?

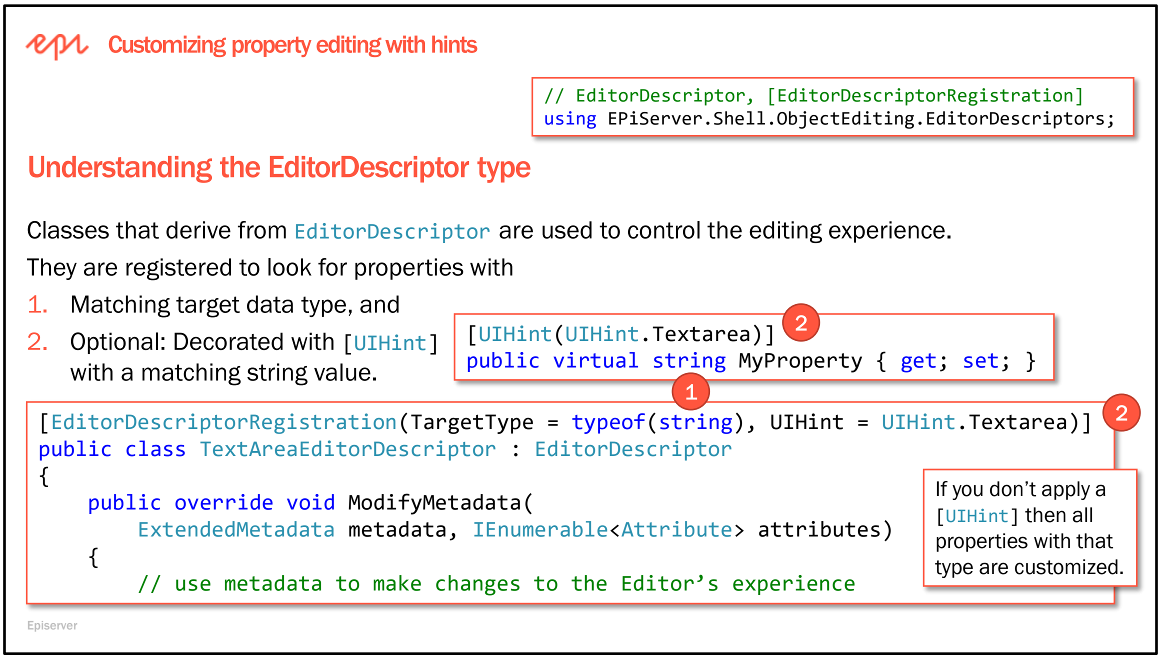
...it is linked to an `EditorDescriptor`!

Episerver

`UIHint.BlockFolder` and `UIHint.MediaFolder` are deprecated in CMS 11. Use `UIHint.AssetsFolder` instead.



```
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";
    }
}
```



Customizing property editing with hints

```
// EditorDescriptor, [EditorDescriptorRegistration]
using EPiServer.Shell.ObjectEditing.EditorDescriptors;
```

Understanding the EditorDescriptor type

Classes that derive from `EditorDescriptor` are used to control the editing experience.

They are registered to look for properties with

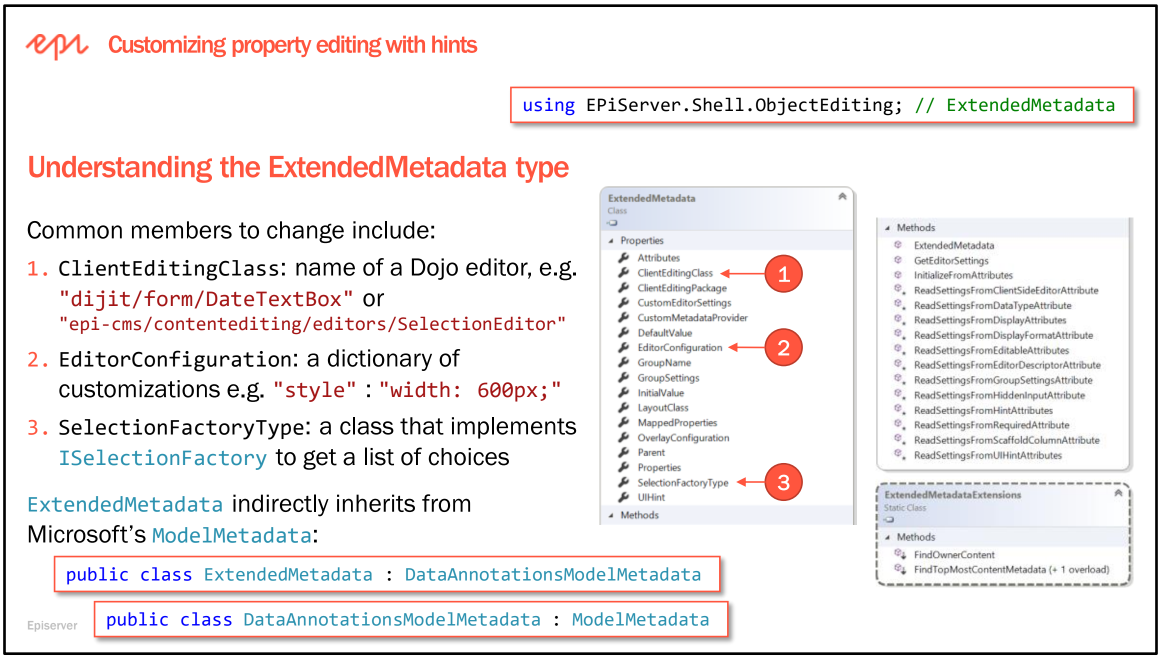
1. Matching target data type, and
2. Optional: Decorated with `[UIHint]` with a matching string value.

```
[UIHint(UIHint.Textarea)]
public virtual string MyProperty { get; set; }
```

```
[EditorDescriptorRegistration(TargetType = typeof(string), UIHint = UIHint.Textarea)]
public class TextAreaEditorDescriptor : EditorDescriptor
{
    public override void ModifyMetadata(
        ExtendedMetadata metadata, IEnumerable<Attribute> attributes)
    {
        // use metadata to make changes to the Editor's experience
```

If you don't apply a `[UIHint]` then all properties with that type are customized.

Episerver



Customizing property editing with hints

`using EPiServer.Shell.ObjectEditing; // ExtendedMetadata`

Understanding the ExtendedMetadata type

Common members to change include:

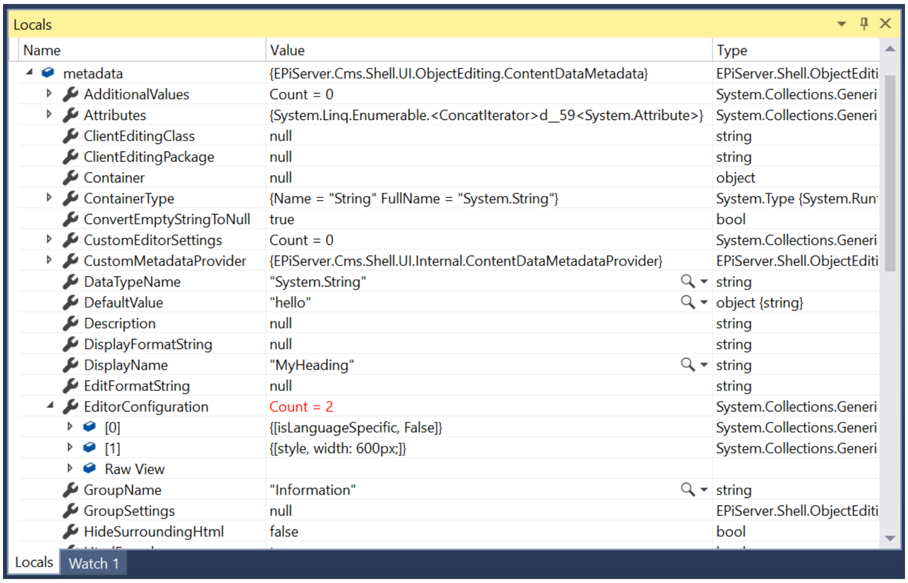
1. `ClientEditingClass`: name of a Dojo editor, e.g. `"dijit/form/DateTextBox"` or `"epi-cms/contentediting/editors/SelectionEditor"`
2. `EditorConfiguration`: a dictionary of customizations e.g. `"style"` : `"width: 600px;"`
3. `SelectionFactoryType`: a class that implements `ISelectionFactory` to get a list of choices

`ExtendedMetadata` indirectly inherits from Microsoft's `ModelMetadata`:

```
public class ExtendedMetadata : DataAnnotationsModelMetadata
```

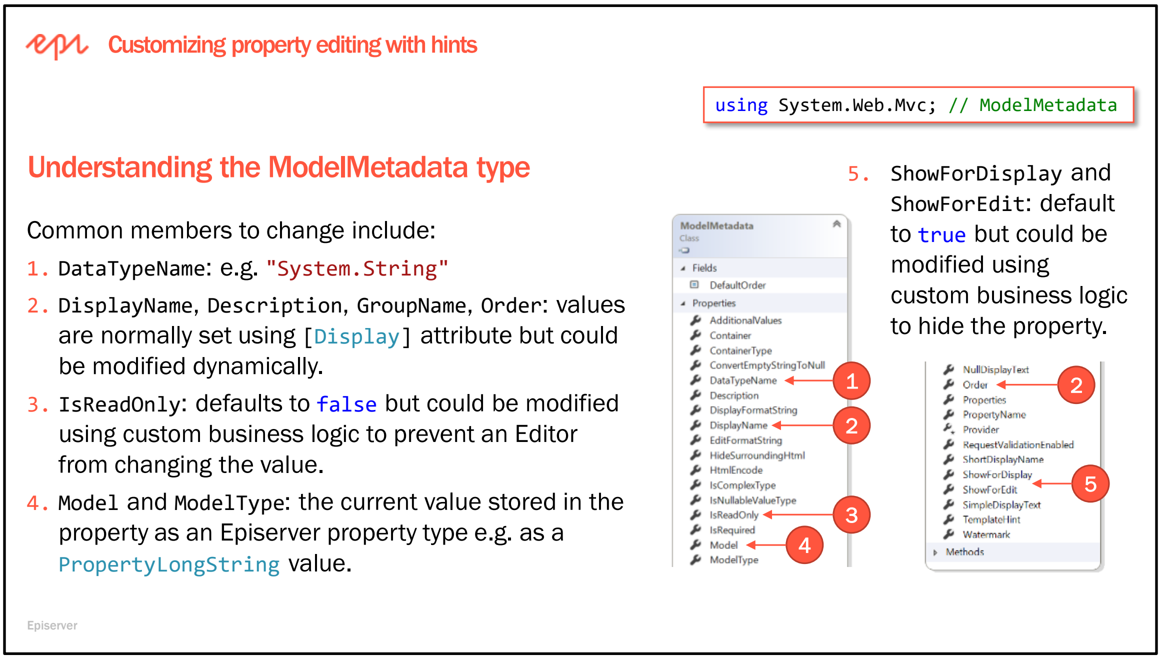
```
public class DataAnnotationsModelMetadata : ModelMetadata
```

ExtendedMetadata
Class
Properties
Attributes
ClientEditingClass
ClientEditingPackage
CustomEditorSettings
CustomMetadataProvider
DefaultValue
EditorConfiguration
GroupName
GroupSettings
InitialValue
LayoutClass
MappedProperties
OverlayConfiguration
Parent
Properties
SelectionFactoryType
UIHint
Methods
1
2
3
Methods
ExtendedMetadata
GetEditorSettings
InitializeFromAttributes
ReadSettingsFromClientSideEditorAttribute
ReadSettingsFromDataTypeAttribute
ReadSettingsFromDisplayAttributes
ReadSettingsFromDisplayFormatAttribute
ReadSettingsFromEditableAttributes
ReadSettingsFromEditorDescriptorAttribute
ReadSettingsFromGroupSettingsAttribute
ReadSettingsFromHiddenInputAttribute
ReadSettingsFromHintAttributes
ReadSettingsFromRequiredAttribute
ReadSettingsFromScaffoldColumnAttribute
ReadSettingsFromUIHintAttributes
ExtendedMetadataExtensions
Static Class
Methods
FindOwnerContent
FindTopMostContentMetadata (+ 1 overload)
Episerver



Locals

Name	Value	Type
metadata	{EPiServer.Cms.Shell.UI.ObjectEditing.ContentDataMetadata}	EPiServer.Shell.ObjectEditi
AdditionalValues	Count = 0	System.Collections.Generi
Attributes	{System.Linq.Enumerable.<ConcatIterator>d__59<System.Attribute>}	System.Collections.Generi
ClientEditingClass	null	string
ClientEditingPackage	null	string
Container	null	object
ContainerType	{Name = "String" FullName = "System.String"}	System.Type {System.Run
ConvertEmptyStringToNull	true	bool
CustomEditorSettings	Count = 0	System.Collections.Generi
CustomMetadataProvider	{EPiServer.Cms.Shell.UI.Internal.ContentDataMetadataProvider}	EPiServer.Shell.ObjectEditi
DataTypeName	"System.String"	string
DefaultValue	"hello"	object {string}
Description	null	string
DisplayFormatString	null	string
DisplayName	"MyHeading"	string
EditFormatString	null	string
EditorConfiguration	Count = 2	System.Collections.Generi
[0]	{[isLanguageSpecific, False]}	System.Collections.Generi
[1]	{[style, width: 600px;]}	System.Collections.Generi
Raw View		
GroupName	"Information"	string
GroupSettings	null	EPiServer.Shell.ObjectEditi
HideSurroundingHtml	false	bool



Customizing property editing with hints

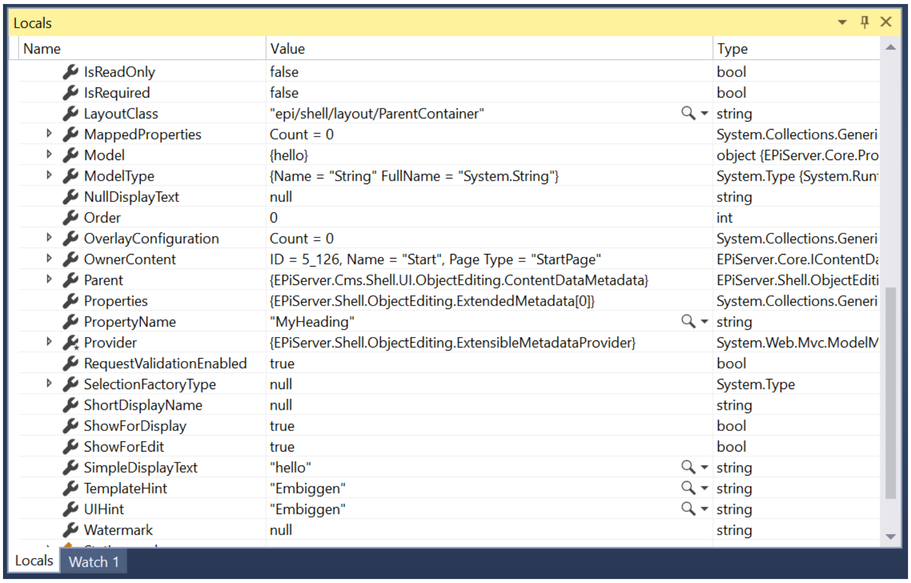
```
using System.Web.Mvc; // ModelMetadata
```

Understanding the ModelMetadata type

Common members to change include:

1. `DataTypeName`: e.g. `"System.String"`
2. `DisplayName`, `Description`, `GroupName`, `Order`: values are normally set using `[Display]` attribute but could be modified dynamically.
3. `IsReadOnly`: defaults to `false` but could be modified using custom business logic to prevent an Editor from changing the value.
4. `Model` and `ModelType`: the current value stored in the property as an Episerver property type e.g. as a `PropertyLongString` value.
5. `ShowForDisplay` and `ShowForEdit`: default to `true` but could be modified using custom business logic to hide the property.

Episerver



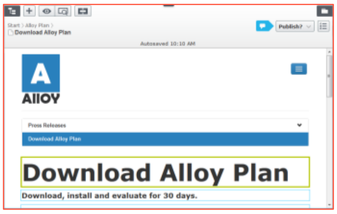
Locals

Name	Value	Type
IsReadOnly	false	bool
IsRequired	false	bool
LayoutClass	"epi/shell/layout/ParentContainer"	string
MappedProperties	Count = 0	System.Collections.Generi
Model	{hello}	object {EPiServer.Core.Pro
ModelType	{Name = "String" FullName = "System.String"}	System.Type {System.Run
NullDisplayText	null	string
Order	0	int
OverlayConfiguration	Count = 0	System.Collections.Generi
OwnerContent	ID = 5_126, Name = "Start", Page Type = "StartPage"	EPiServer.Core.IContentDa
Parent	{EPiServer.Cms.Shell.UI.ObjectEditing.ContentDataMetadata}	EPiServer.Shell.ObjectEditi
Properties	{EPiServer.Shell.ObjectEditing.ExtendedMetadata[0]}	System.Collections.Generi
PropertyName	"MyHeading"	string
Provider	{EPiServer.Shell.ObjectEditing.ExtensibleMetadataProvider}	System.Web.Mvc.ModelM
RequestValidationEnabled	true	bool
SelectionFactoryType	null	System.Type
ShortDisplayName	null	string
ShowForDisplay	true	bool
ShowForEdit	true	bool
SimpleDisplayText	"hello"	string
TemplateHint	"Embiggen"	string
UIHint	"Embiggen"	string
Watermark	null	string

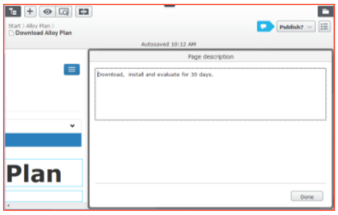
Customizing property editing with hints

Controlling the popup during On-Page Editing

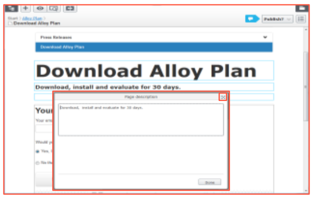
```
public override void ModifyMetadata(
        ExtendedMetadata metadata, IEnumerable<Attribute> attributes)
    {
        metadata.CustomEditorSettings["uiWrapperType"] = UiWrapperType.Flyout;
```



UiWrapperType.ContentEditable

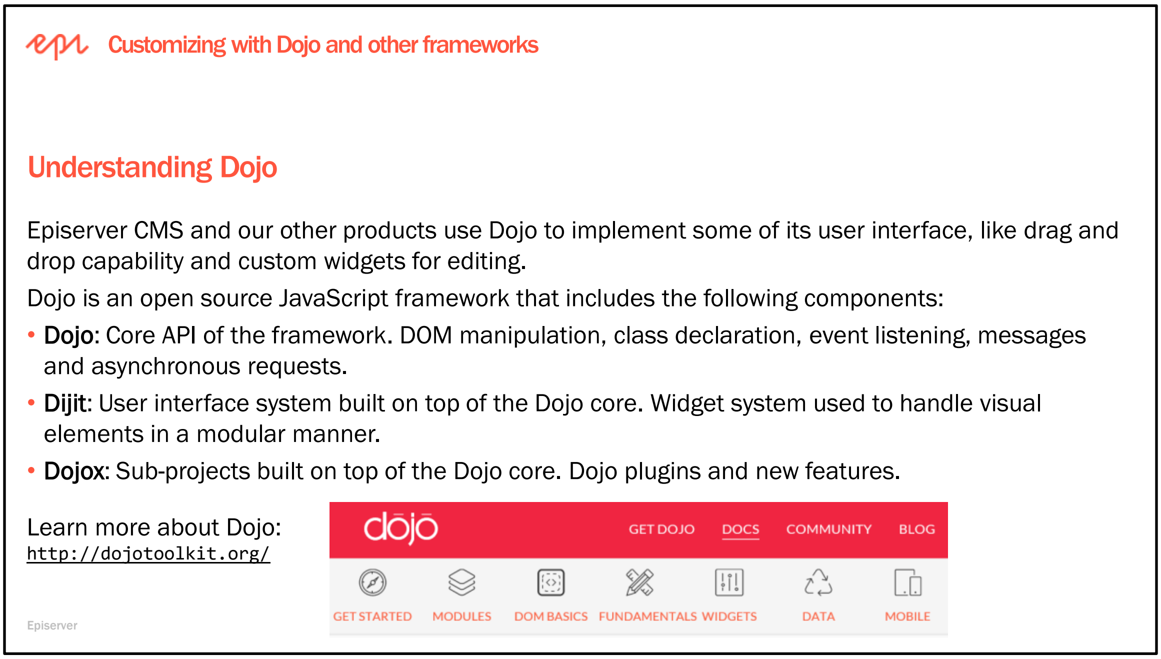


UiWrapperType.Flyout


Publish?
Press Releases
Download Alloy Plan
Download Alloy Plan
Download, install and evaluate for 30 days.
Page description
Download, install and evaluate for 30 days.
Done

UiWrapperType.Floating

Episerver



Customizing with Dojo and other frameworks

Understanding Dojo

Episerver CMS and our other products use Dojo to implement some of its user interface, like drag and drop capability and custom widgets for editing.

Dojo is an open source JavaScript framework that includes the following components:

- **Dojo**: Core API of the framework. DOM manipulation, class declaration, event listening, messages and asynchronous requests.
- **Dijit**: User interface system built on top of the Dojo core. Widget system used to handle visual elements in a modular manner.
- **Dojox**: Sub-projects built on top of the Dojo core. Dojo plugins and new features.

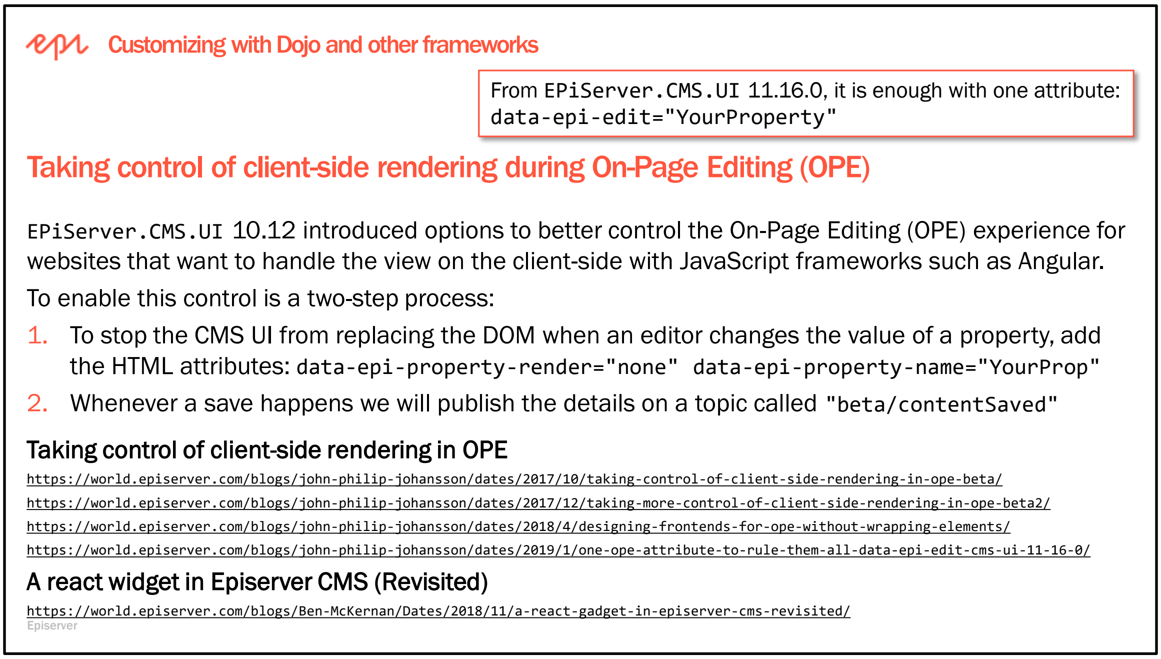
Learn more about Dojo:
`http://dojotoolkit.org/`

Episerver

https://dojotoolkit.org/reference-guide/1.10/dijit/index.html

dijit/form/CurrencyTextBox
A specialized input widget for monetary values, much like the currency type in spreadsheet programs
dijit/form/DateTextBox
An easy-to-use date entry control which allows either typing or choosing a date from any calendar widget
dijit/form/MappedTextBox
A subclass of dijit/form/ValidationTextBox that is designed to be a base class for widgets that have a visible formatted display value, and a serializable value in a hidden input field which is actually sent to the server.
dijit/form/NumberSpinner
An input widget which restricts input to numeric input and offers down and up arrow buttons to “spin” the number up and down
dijit/form/NumberTextBox
A input widget which restricts input to numeric input
dijit/form/RangeBoundTextBox
A base class for textbox form widgets which define a range of valid values.
dijit/form/Textarea
An auto expanding/contracting <textarea>
dijit/form/TimeTextBox
A time input control which allows either typing or choosing a time from any time-picker widget
dijit/form/ValidationTextBox
A class for textbox widgets with the ability to validate various types of content and to provide user feedback.

Inspired by David Knipe’s blog post, **Creating a time picker property for Episerver using a Dojo dijit**:
https://www.david-tec.com/2016/12/creating-a-time-picker-property-for-episerver-using-a-dojo-dijit/



Customizing with Dojo and other frameworks

From `EPiServer.CMS.UI` 11.16.0, it is enough with one attribute: `data-epi-edit="YourProperty"`

Taking control of client-side rendering during On-Page Editing (OPE)

`EPiServer.CMS.UI` 10.12 introduced options to better control the On-Page Editing (OPE) experience for websites that want to handle the view on the client-side with JavaScript frameworks such as Angular.

To enable this control is a two-step process:

1. To stop the CMS UI from replacing the DOM when an editor changes the value of a property, add the HTML attributes: `data-epi-property-render="none"` `data-epi-property-name="YourProp"`
2. Whenever a save happens we will publish the details on a topic called `"beta/contentSaved"`

Taking control of client-side rendering in OPE

https://world.episerver.com/blogs/john-philip-johansson/dates/2017/10/taking-control-of-client-side-rendering-in-ope-beta/
https://world.episerver.com/blogs/john-philip-johansson/dates/2017/12/taking-more-control-of-client-side-rendering-in-ope-beta2/
https://world.episerver.com/blogs/john-philip-johansson/dates/2018/4/designing-frontends-for-ope-without-wrapping-elements/
https://world.episerver.com/blogs/john-philip-johansson/dates/2019/1/one-ope-attribute-to-rule-them-all-data-epi-edit-cms-ui-11-16-0/

A react widget in Episerver CMS (Revisited)

https://world.episerver.com/blogs/Ben-McKernan/Dates/2018/11/a-react-gadget-in-episerver-cms-revisited/

Episerver

Designing frontends for OPE without wrapping elements By John-Philip Johansson

A common scenario I have seen is that a frontend developer or designer implements a design in HTML, CSS, and maybe JS, without worrying about which CMS is used to render it. The code is then copied or moved into Episerver, most often into a Razor view, by an Episerver developer. Then everyone sees the page in On-Page Edit (OPE) and gets a little sad as some of that lovely design is broken. That makes the developers get even sadder as they have to re-do some of the design to work with the extra div elements added by OPE, but at least it will look lovely again.

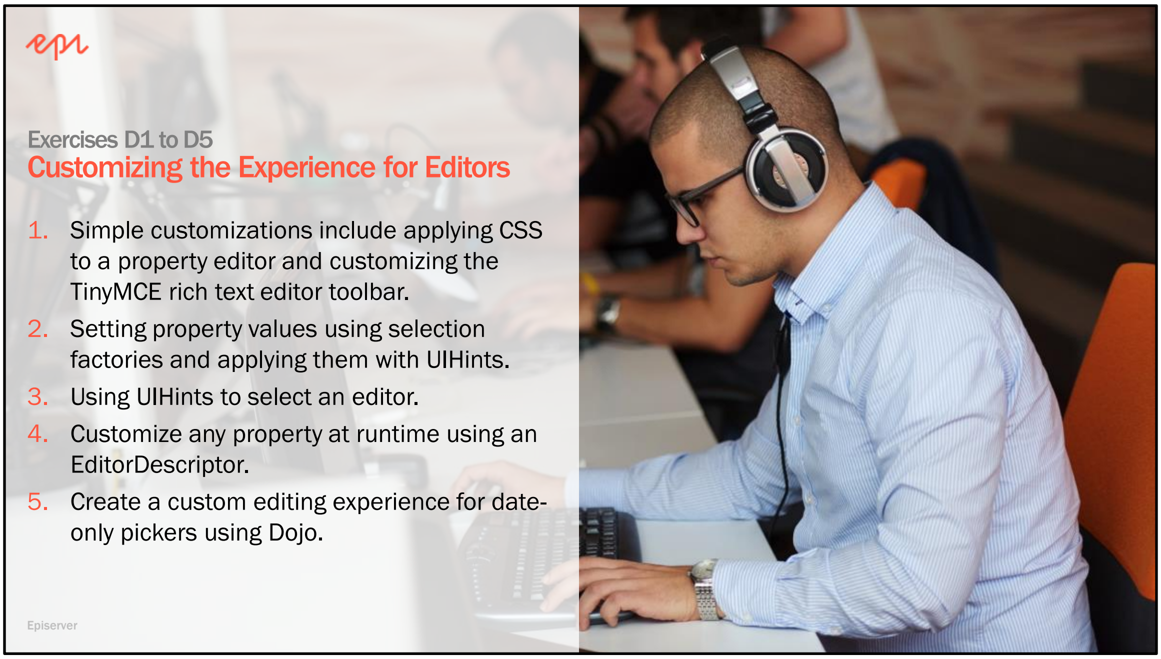
We would like you to use the HTML structure you want. If you are rendering and handling updates purely in your client-side framework of choice, you should already be able to do this. If you are using Razor, then let us discuss two common design implementations that break, and what we can do with them. But first, let us talk about our two HTML helpers @Html.PropertyFor and @Html.EditAttributes.

https://world.episerver.com/blogs/john-philip-johansson/dates/2018/4/designing-frontends-for-ope-without-wrapping-elements/

Introducing a new SPA template site: MusicFestival

To demonstrate some concepts that are useful when creating a SPA with working OPE, we have released a new SPA template site on Github, called MusicFestival.

https://world.episerver.com/blogs/john-philip-johansson/dates/2018/10/introducing-a-new-template-site-for-spas-musicfestival/

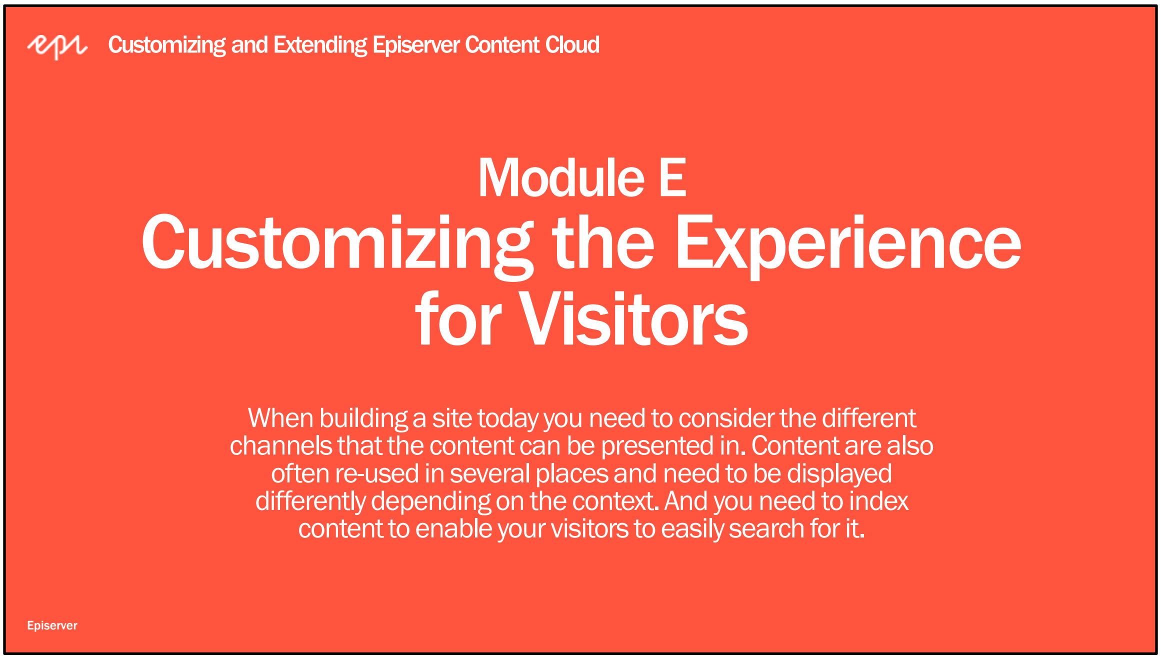


Exercises D1 to D5

Customizing the Experience for Editors

1. Simple customizations include applying CSS to a property editor and customizing the TinyMCE rich text editor toolbar.
2. Setting property values using selection factories and applying them with UIHints.
3. Using UIHints to select an editor.
4. Customize any property at runtime using an EditorDescriptor.
5. Create a custom editing experience for date-only pickers using Dojo.

Episerver



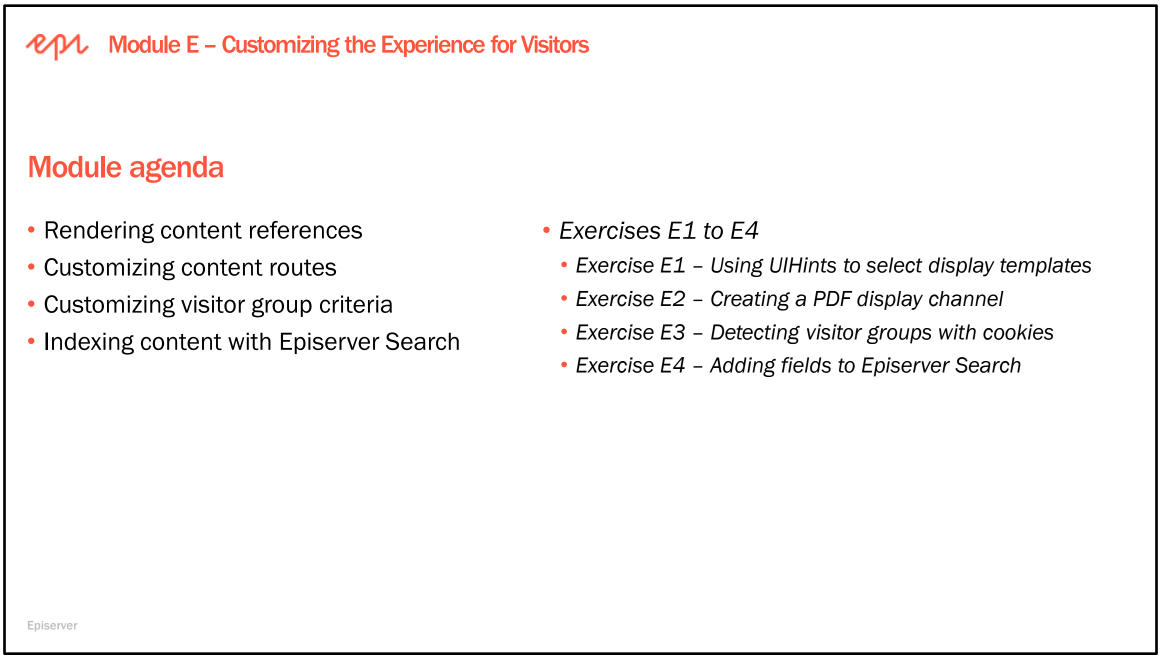
Customizing and Extending Episerver Content Cloud

Module E

Customizing the Experience for Visitors

When building a site today you need to consider the different channels that the content can be presented in. Content are also often re-used in several places and need to be displayed differently depending on the context. And you need to index content to enable your visitors to easily search for it.

Episerver



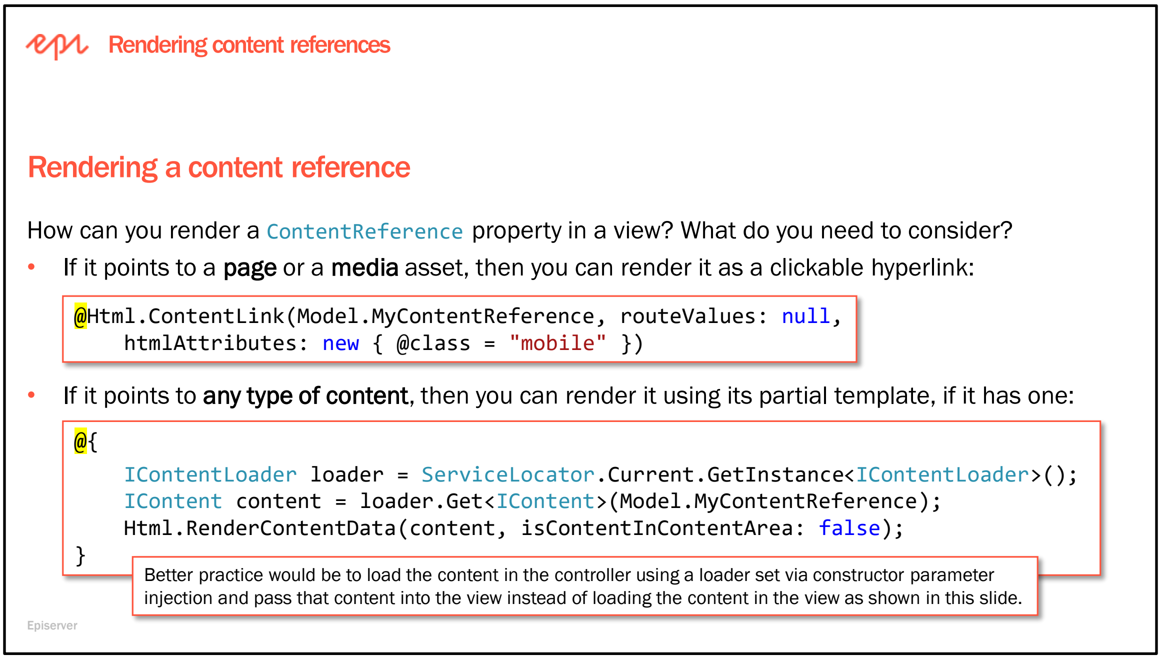
Module E – Customizing the Experience for Visitors

Module agenda

- Rendering content references
- Customizing content routes
- Customizing visitor group criteria
- Indexing content with Episerver Search

- *Exercises E1 to E4*
  - *Exercise E1 – Using UIHints to select display templates*
  - *Exercise E2 – Creating a PDF display channel*
  - *Exercise E3 – Detecting visitor groups with cookies*
  - *Exercise E4 – Adding fields to Episerver Search*

Episerver



Rendering content references

Rendering a content reference

How can you render a `ContentReference` property in a view? What do you need to consider?

- If it points to a **page** or a **media** asset, then you can render it as a clickable hyperlink:

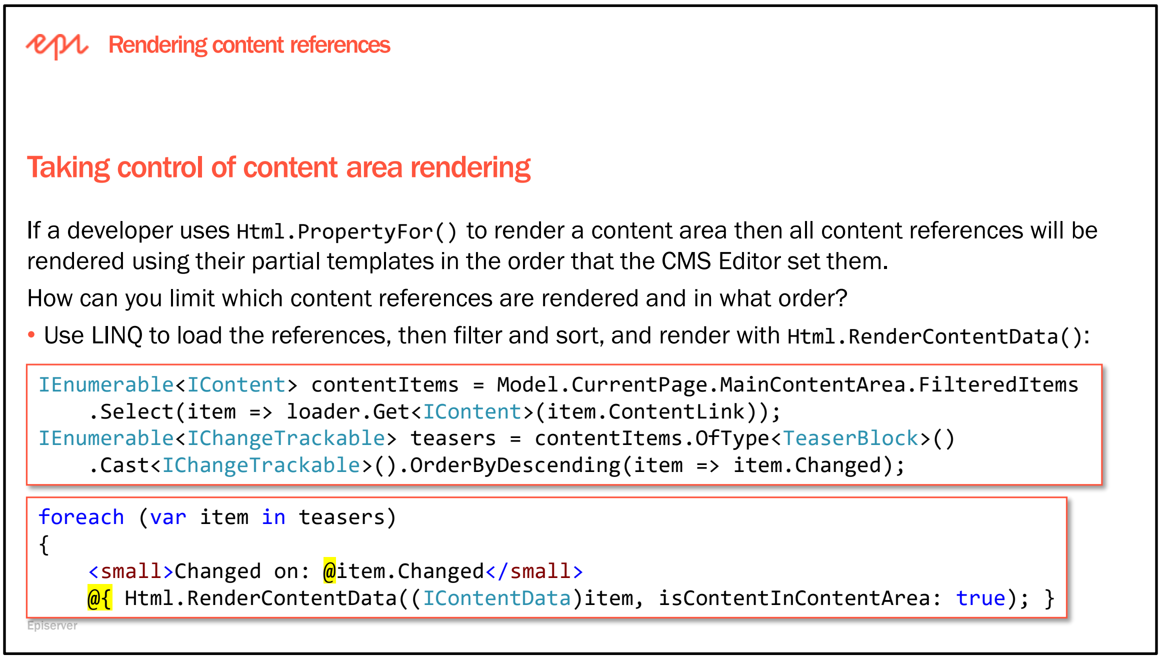
```
@Html.ContentLink(Model.MyContentReference, routeValues: null,
    htmlAttributes: new { @class = "mobile" })
```

- If it points to **any type of content**, then you can render it using its partial template, if it has one:

```
@{
    IContentLoader loader = ServiceLocator.Current.GetInstance<IContentLoader>();
    IContent content = loader.Get<IContent>(Model.MyContentReference);
    Html.RenderContentData(content, isContentInContentArea: false);
}
```

Better practice would be to load the content in the controller using a loader set via constructor parameter injection and pass that content into the view instead of loading the content in the view as shown in this slide.

Episerver



Rendering content references

Taking control of content area rendering

If a developer uses `Html.PropertyFor()` to render a content area then all content references will be rendered using their partial templates in the order that the CMS Editor set them.

How can you limit which content references are rendered and in what order?

- Use LINQ to load the references, then filter and sort, and render with `Html.RenderContentData()`:

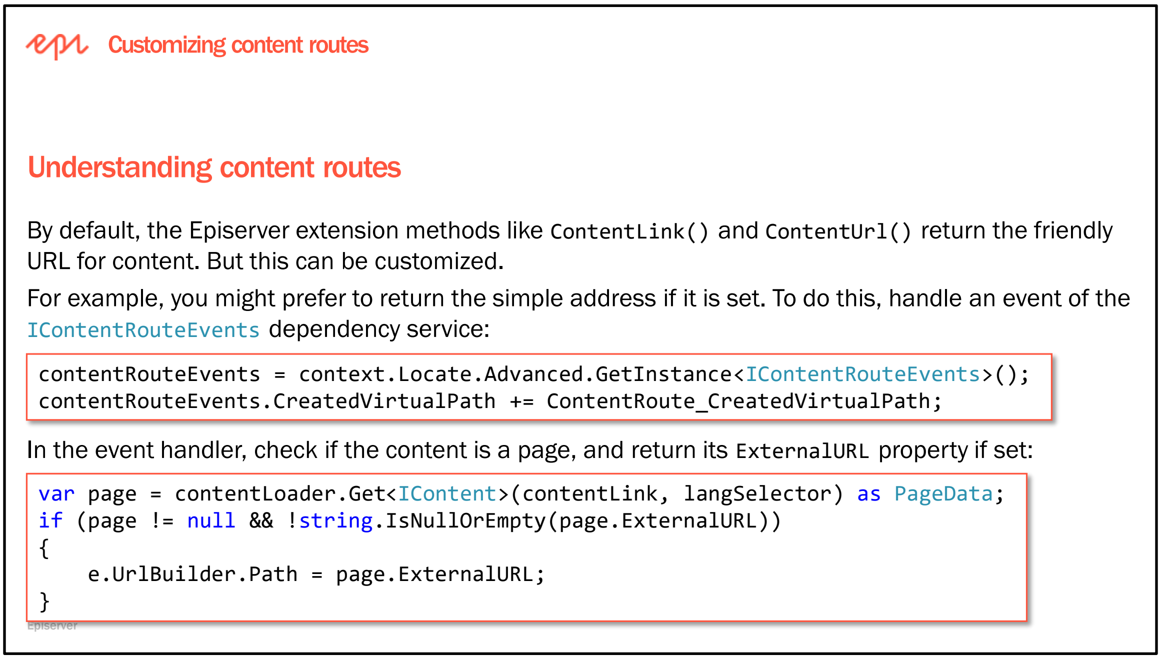
```
IEnumerable<IContent> contentItems = Model.CurrentPage.MainContentArea.FilteredItems
    .Select(item => loader.Get<IContent>(item.ContentLink));
IEnumerable<IChangeTrackable> teasers = contentItems.OfType<TeaserBlock>()
    .Cast<IChangeTrackable>().OrderByDescending(item => item.Changed);
```

```
foreach (var item in teasers)
{
    <small>Changed on: @item.Changed</small>
    @{ Html.RenderContentData((IContentData)item, isContentInContentArea: true); }
```

Episerver

```
@{
    var loader = ServiceLocator.Current.GetInstance<EPiServer.IContentLoader>();

    var contentItems = Model.CurrentPage.MainContentArea.FilteredItems
        .Select(item => loader.Get<IContent>(item.ContentLink));

    var teasers = contentItems.OfType<TeaserBlock>()
        .Cast<IChangeTrackable>()
        .OrderByDescending(item => item.Changed);

    foreach (var item in teasers)
    {
        <div>
            <small>Changed on: @ item.Changed</small>
            @{
                Html.RenderContentData((IContentData)item,
                    isContentInContentArea: true);
            }
        </div>
    }
}
```

Taking control of content area rendering would also allow you to modify the markup used, so you could implement a carousel instead of a stack of blocks:
http://world.episerver.com/Blogs/pezi/Dates/2013/5/Create-an-animating-slider-with-content-area/



Customizing content routes

Understanding content routes

By default, the Episerver extension methods like `ContentLink()` and `ContentUrl()` return the friendly URL for content. But this can be customized.

For example, you might prefer to return the simple address if it is set. To do this, handle an event of the `IContentRouteEvents` dependency service:

```
contentRouteEvents = context.Locate.Advanced.GetInstance<IContentRouteEvents>();
contentRouteEvents.CreatedVirtualPath += ContentRoute_CreatedVirtualPath;
```

In the event handler, check if the content is a page, and return its `ExternalURL` property if set:

```
var page = contentLoader.Get<IContent>(contentLink, langSelector) as PageData;
if (page != null && !string.IsNullOrEmpty(page.ExternalURL))
{
    e.UrlBuilder.Path = page.ExternalURL;
}
```

Episerver

davidknipe/AddTimeStampToImages.cs

Add a hash based on the image timestamp to ensure images are reloaded whenever they are changed.

https://gist.github.com/davidknipe/8a05d807dc73c198c51b

EpiCdnHandler

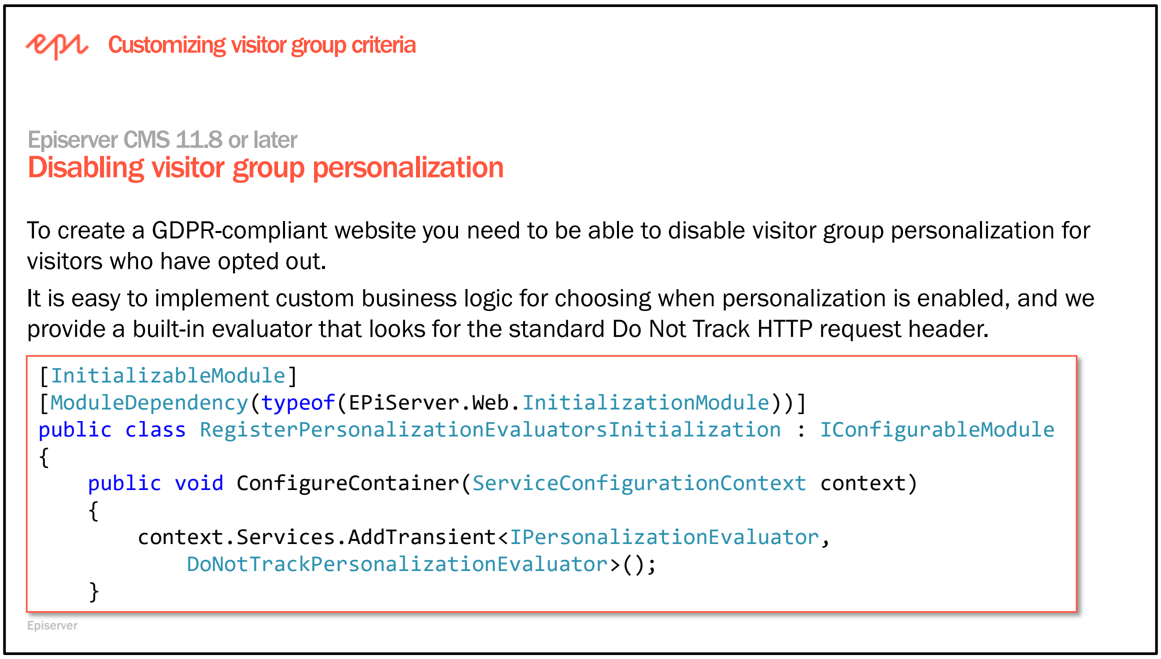
Customer origin CDN support for EpiServer 7.5 (or newer). The module will rewrite and handle all image urls. It will add a version hash to the urls. The module will set http headers on the requests to "permanently" cache the image files on the client.

https://github.com/torjue/EpiCdnHandler/blob/master/EpiCdnHandler/UrlBuilder.cs

CanonicalLink extension method

In Episerver CMS version 11.11.2, this method outputs a relative URL, but recently Google's guidance for canonical links recommends including your domain i.e. use absolute URLs, as discussed in the following link:

https://support.google.com/webmasters/answer/139066?hl=en



Customizing visitor group criteria

Episerver CMS 11.8 or later

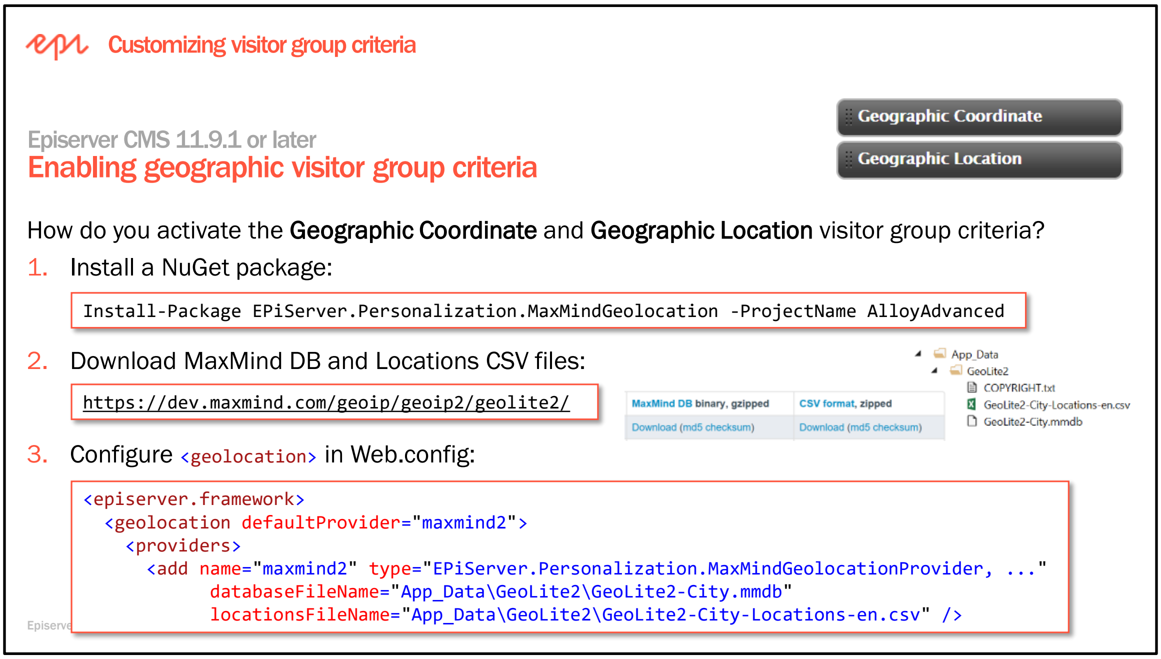
Disabling visitor group personalization

To create a GDPR-compliant website you need to be able to disable visitor group personalization for visitors who have opted out.

It is easy to implement custom business logic for choosing when personalization is enabled, and we provide a built-in evaluator that looks for the standard Do Not Track HTTP request header.

```
[InitializableModule]
[ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
public class RegisterPersonalizationEvaluatorsInitialization : IConfigurableModule
{
    public void ConfigureContainer(ServiceConfigurationContext context)
    {
        context.Services.AddTransient<IPersonalizationEvaluator,
            DoNotTrackPersonalizationEvaluator>();
    }
```

Episerver



Customizing visitor group criteria

Episerver CMS 11.9.1 or later

Enabling geographic visitor group criteria

How do you activate the **Geographic Coordinate** and **Geographic Location** visitor group criteria?

1. Install a NuGet package:

```
Install-Package EPiServer.Personalization.MaxMindGeolocation -ProjectName AlloyAdvanced
```

2. Download MaxMind DB and Locations CSV files:

https://dev.maxmind.com/geoip/geoip2/geolite2/

MaxMind DB binary, gzipped	CSV format, zipped
Download (md5 checksum)	Download (md5 checksum)

```
App_Data
  GeoLite2
    COPYRIGHT.txt
    GeoLite2-City-Locations-en.csv
    GeoLite2-City.mmdb
```

3. Configure `<geolocation>` in Web.config:

```
<episerver.framework>
  <geolocation defaultProvider="maxmind2">
    <providers>
      <add name="maxmind2" type="EPiServer.Personalization.MaxMindGeolocationProvider, ..."
            databaseFileName="App_Data\GeoLite2\GeoLite2-City.mmdb"
            locationsFileName="App_Data\GeoLite2\GeoLite2-City-Locations-en.csv" />
```

Episerve

Geolocation provider changes

Episerver CMS used to come with built-in Geolocation support for MaxMind's GeoLite database, but MaxMind has decided to discontinue our GeoLite Legacy databases effective January 2, 2019.

https://support.maxmind.com/geolite-legacy-discontinuation-notice/

As a replacement, MaxMind is instead offering the free GeoLite2 or the commercial GeoIP2 database which both comes with IPv6 support.

A new provider for GeoLite2 databases

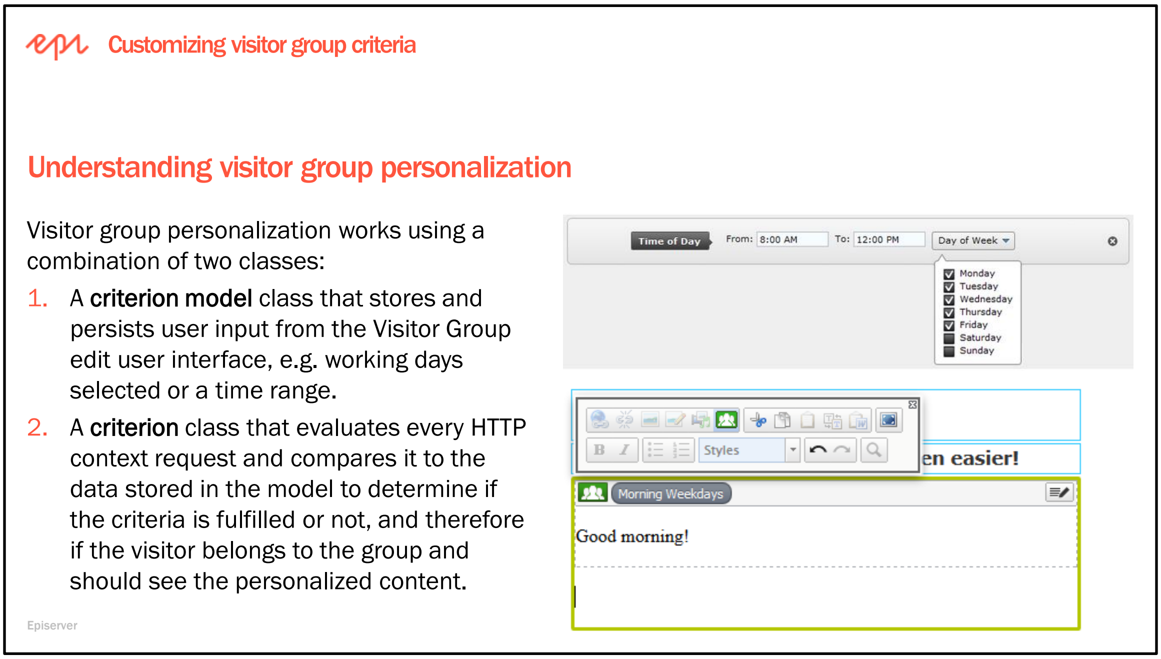
A new separate NuGet package called `EPiServer.Personalization.MaxMindGeolocation` has been released. This package includes a Geolocation provider with support for MaxMind's GeoLite2 database. The package is distributed without a MaxMind Geolocation database.

You can acquire a Geolocation database from MaxMind by downloading the free GeoLite2 database at:

https://dev.maxmind.com/geoip/geoip2/geolite2/

You will also need to download the database in CSV format for the provider to be able to list all available Locations. The provider will work with both the Country and City database types.

https://world.episerver.com/blogs/Henrik-Nystrom/Dates/2018/6/geolocation-provider-changes/



Customizing visitor group criteria

Understanding visitor group personalization

Visitor group personalization works using a combination of two classes:

1. A **criterion model** class that stores and persists user input from the Visitor Group edit user interface, e.g. working days selected or a time range.
2. A **criterion** class that evaluates every HTTP context request and compares it to the data stored in the model to determine if the criteria is fulfilled or not, and therefore if the visitor belongs to the group and should see the personalized content.

Time of Day From: 8:00 AM To: 12:00 PM Day of Week

☑ Monday
☑ Tuesday
☑ Wednesday
☑ Thursday
☑ Friday
☐ Saturday
☐ Sunday

Good morning!

Episerver

Disable visitor group personalization

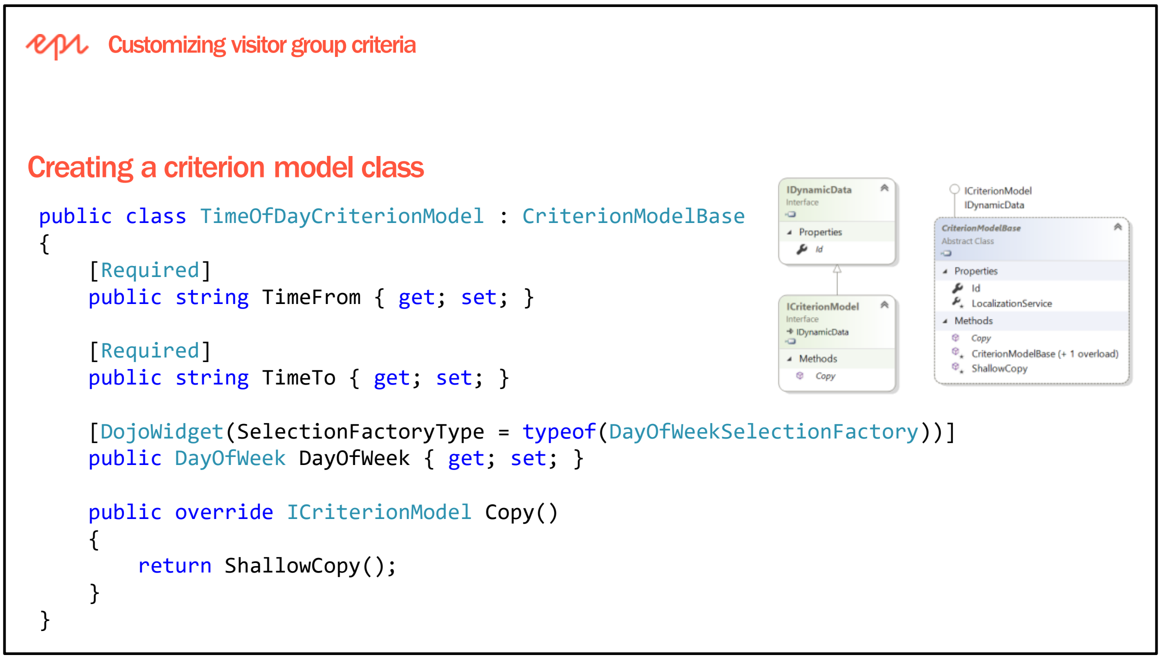
`IPersonalizationEvaluator` is an interface that can be implemented to control whether personalization should occur or not. Episerver CMS includes an implementation that checks for presence of a Do Not Track header. If the header is present, no personalization is done for the request and no cookies are stored.

https://world.episerver.com/documentation/developer-guides/CMS/personalization/disable-visitor-group-personalization/

Session handling in visitor group criteria

You can use visitor group criteria without requiring session state by disabling ASP.NET Session state. The visitor group system will autodetect this configuration and switch to a cookie-based approach instead. You can also customize your own storage of users' visitor group sessions.

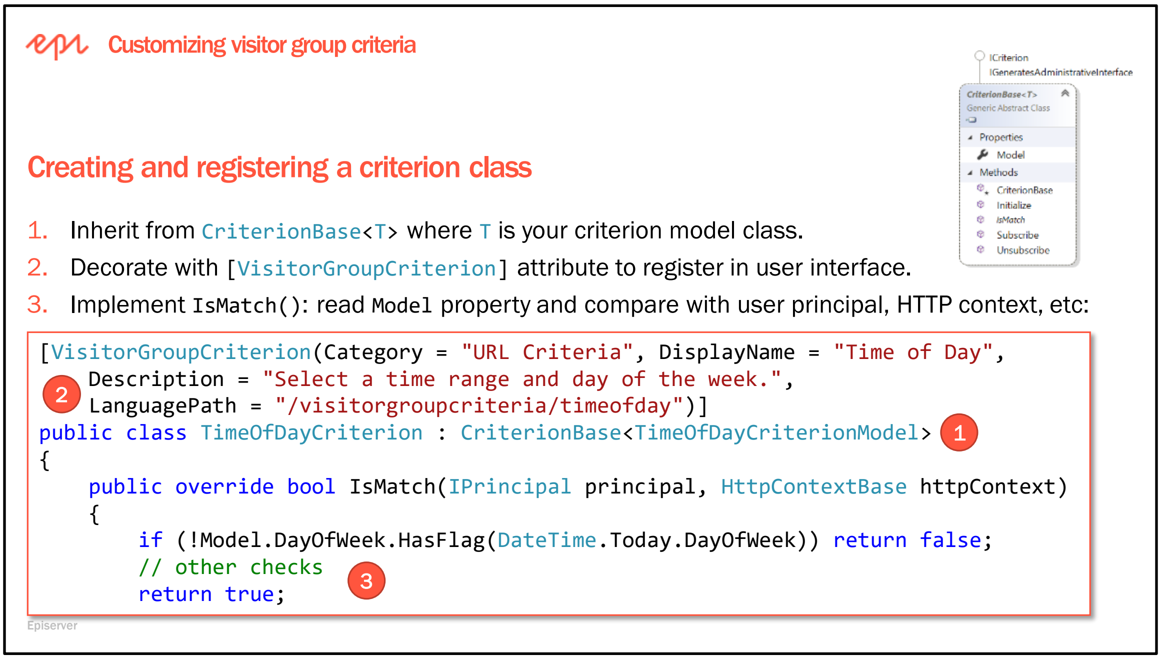
https://world.episerver.com/documentation/developer-guides/CMS/personalization/session-handling-in-visitor-group-criteria/



Customizing visitor group criteria

Creating a criterion model class

```
public class TimeOfDayCriterionModel : CriterionModelBase
{
    [Required]
    public string TimeFrom { get; set; }

    [Required]
    public string TimeTo { get; set; }

    [DojoWidget(SelectionFactoryType = typeof(DayOfWeekSelectionFactory))]
    public DayOfWeek DayOfWeek { get; set; }

    public override ICriterionModel Copy()
    {
        return ShallowCopy();
    }
}
```



Customizing visitor group criteria

Creating and registering a criterion class

1. Inherit from `CriterionBase<T>` where `T` is your criterion model class.
2. Decorate with `[VisitorGroupCriterion]` attribute to register in user interface.
3. Implement `IsMatch()`: read `Model` property and compare with user principal, HTTP context, etc:

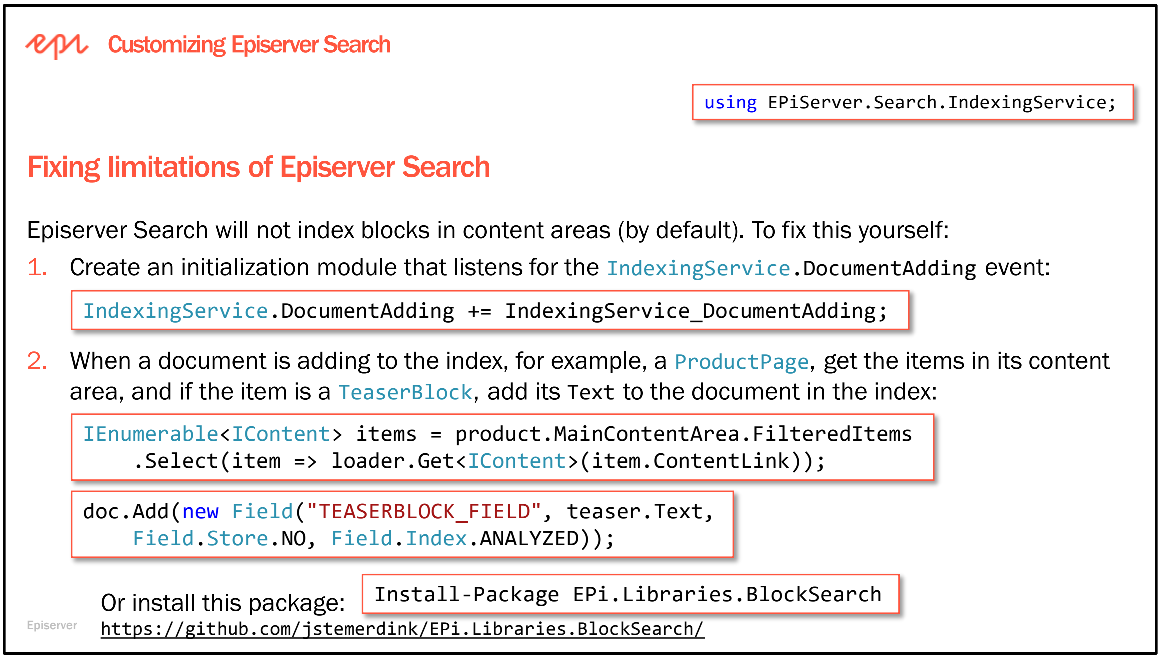
```
[VisitorGroupCriterion(Category = "URL Criteria", DisplayName = "Time of Day",
    Description = "Select a time range and day of the week.",
    LanguagePath = "/visitorgroupcriteria/timeofday")]
public class TimeOfDayCriterion : CriterionBase<TimeOfDayCriterionModel>
{
    public override bool IsMatch(IPrincipal principal, HttpContextBase httpContext)
    {
        if (!Model.DayOfWeek.HasFlag(DateTime.Today.DayOfWeek)) return false;
        // other checks
        return true;
```

Your custom criterion class must evaluate the HTTP context and the data stored in the model to determine if the criteria is fulfilled or not. The connection between the criterion and model classes is created via CriterionBase – the base class that must be used for the criterion class – which is a generic class that accepts ICriterionModel parameters.

The only method you must override is CriterionBase.IsMatch which is the central method for a criterion, it is the method that will be called when evaluating if a user is a member of a visitor group.

The criterion class must also be decorated with VisitorGroupCriterion attribute, which identifies your class as a criterion and makes it available for use.

- **Category**: The name of the group in the criteria picker UI where this criterion will be located. Criteria with the same Category value will be grouped together.
- **DisplayName**: A short name that is used to identify the criterion in menus and visitor groups.



Customizing Episerver Search

```
using EPiServer.Search.IndexingService;
```

Fixing limitations of Episerver Search

Episerver Search will not index blocks in content areas (by default). To fix this yourself:

1. Create an initialization module that listens for the `IndexingService.DocumentAdding` event:

```
IndexingService.DocumentAdding += IndexingService_DocumentAdding;
```

2. When a document is adding to the index, for example, a `ProductPage`, get the items in its content area, and if the item is a `TeaserBlock`, add its `Text` to the document in the index:

```
IEnumerable<IContent> items = product.MainContentArea.FilteredItems
    .Select(item => loader.Get<IContent>(item.ContentLink));
```

```
doc.Add(new Field("TEASERBLOCK_FIELD", teaser.Text,
    Field.Store.NO, Field.Index.ANALYZED));
```

Or install this package:

```
Install-Package EPi.Libraries.BlockSearch
```

https://github.com/jstemerdink/EPi.Libraries.BlockSearch/

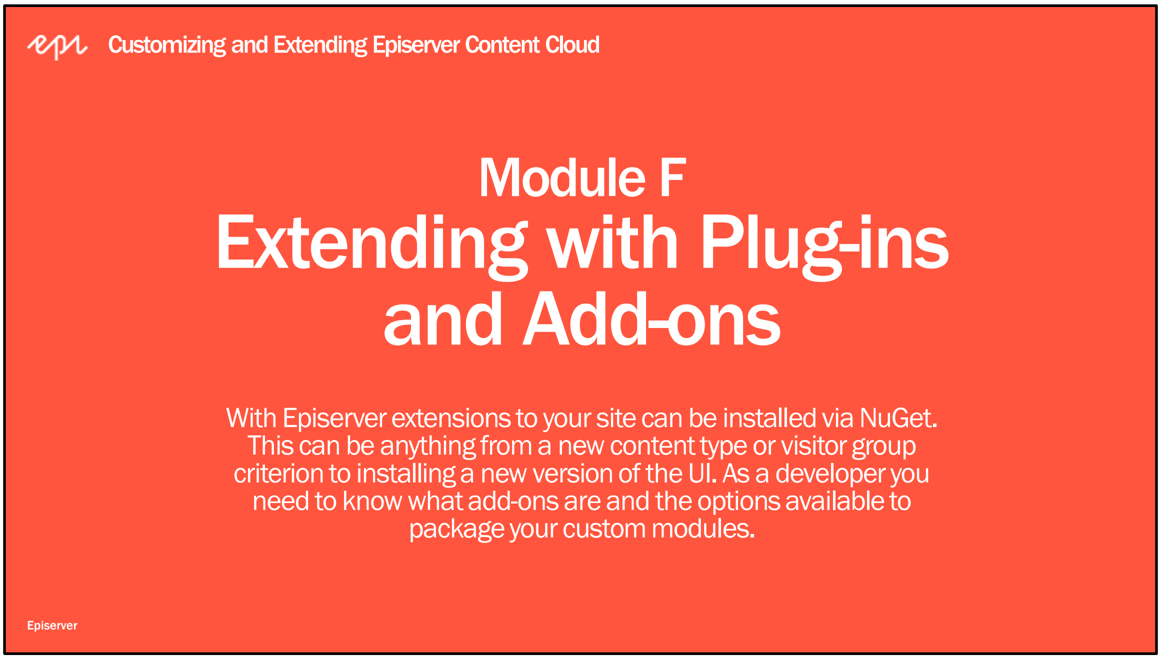
Episerver

Exercises E1 to E4

Customizing the Experience for Visitors

1. Using UIHints to apply display templates
2. Creating a PDF display channel
3. Detecting visitor groups with cookies
4. Adding fields to Episerver Search

Episerver

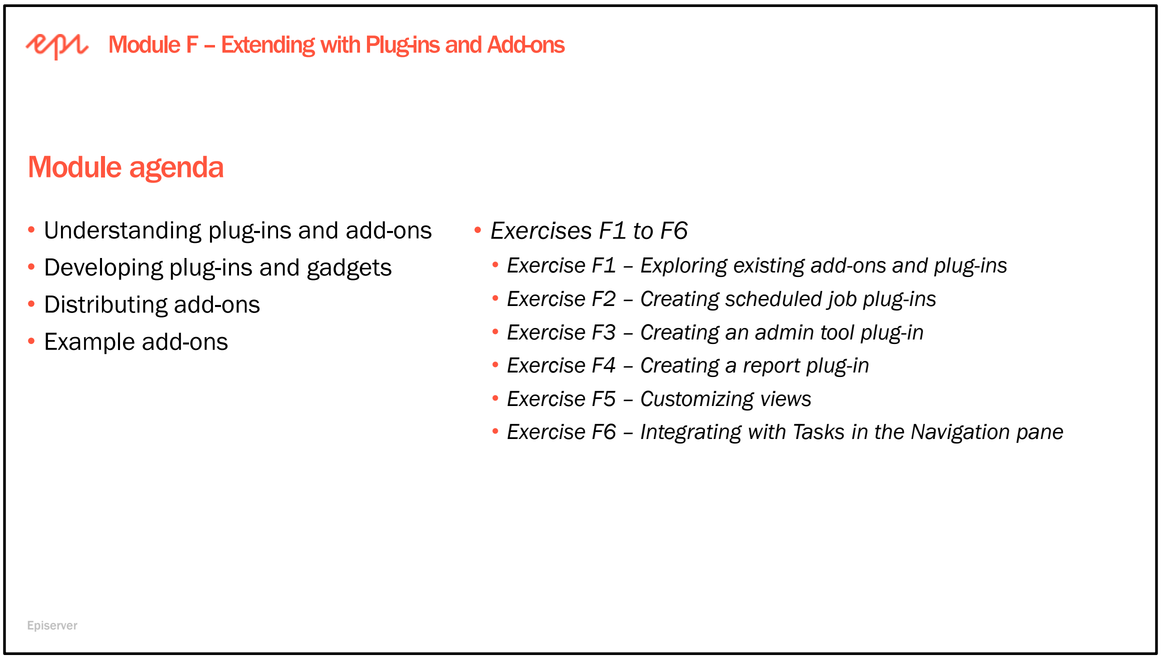


Customizing and Extending Episerver Content Cloud

Module F
Extending with Plug-ins and Add-ons

With Episerver extensions to your site can be installed via NuGet. This can be anything from a new content type or visitor group criterion to installing a new version of the UI. As a developer you need to know what add-ons are and the options available to package your custom modules.

Episerver



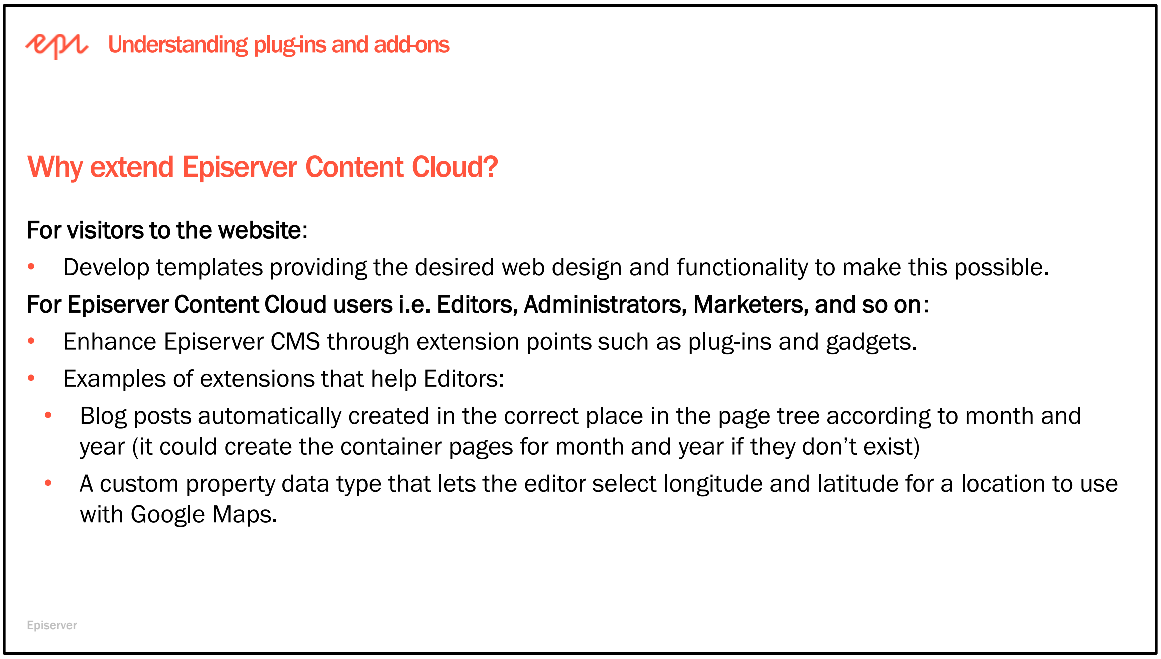
Module F – Extending with Plug-ins and Add-ons

Module agenda

- Understanding plug-ins and add-ons
- Developing plug-ins and gadgets
- Distributing add-ons
- Example add-ons

- *Exercises F1 to F6*
  - *Exercise F1 – Exploring existing add-ons and plug-ins*
  - *Exercise F2 – Creating scheduled job plug-ins*
  - *Exercise F3 – Creating an admin tool plug-in*
  - *Exercise F4 – Creating a report plug-in*
  - *Exercise F5 – Customizing views*
  - *Exercise F6 – Integrating with Tasks in the Navigation pane*

Episerver



Understanding plug-ins and add-ons

Why extend Episerver Content Cloud?

For visitors to the website:

- Develop templates providing the desired web design and functionality to make this possible.

For Episerver Content Cloud users i.e. Editors, Administrators, Marketers, and so on:

- Enhance Episerver CMS through extension points such as plug-ins and gadgets.
- Examples of extensions that help Editors:
  - Blog posts automatically created in the correct place in the page tree according to month and year (it could create the container pages for month and year if they don't exist)
  - A custom property data type that lets the editor select longitude and latitude for a location to use with Google Maps.

Episerver

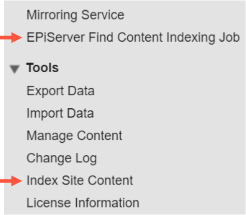
Understanding plug-ins and add-ons

Terms for extension points

Plug-in: an extension that is available to all CMS users, e.g. scheduled job, custom tool, report, and so on.

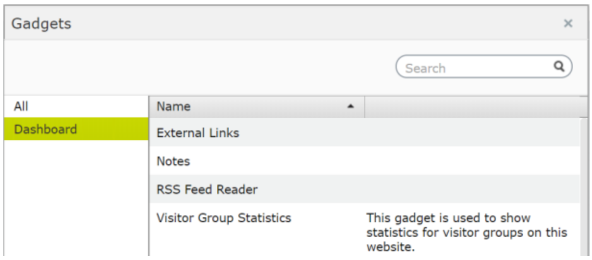
Gadget: an extension that CMS users can choose to add to their Dashboard or Edit view panes. Each user has their own configuration of gadgets.

Add-on: a way to package an extension for distribution via Episerver's NuGet feed. Plug-ins and gadgets do not need to be packaged as an add-on if you are deploying internally.



Scheduled job plug-in

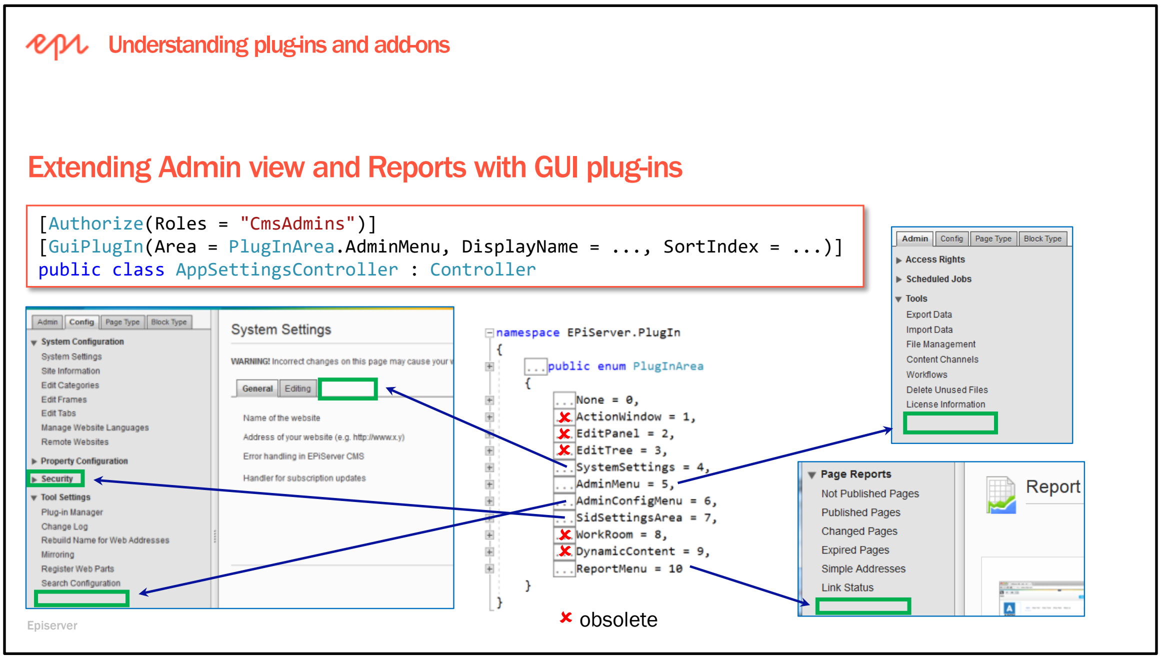
Custom tool plug-in



Name	
External Links	
Notes	
RSS Feed Reader	
Visitor Group Statistics	This gadget is used to show statistics for visitor groups on this website.

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

Episerver



Understanding plug-ins and add-ons

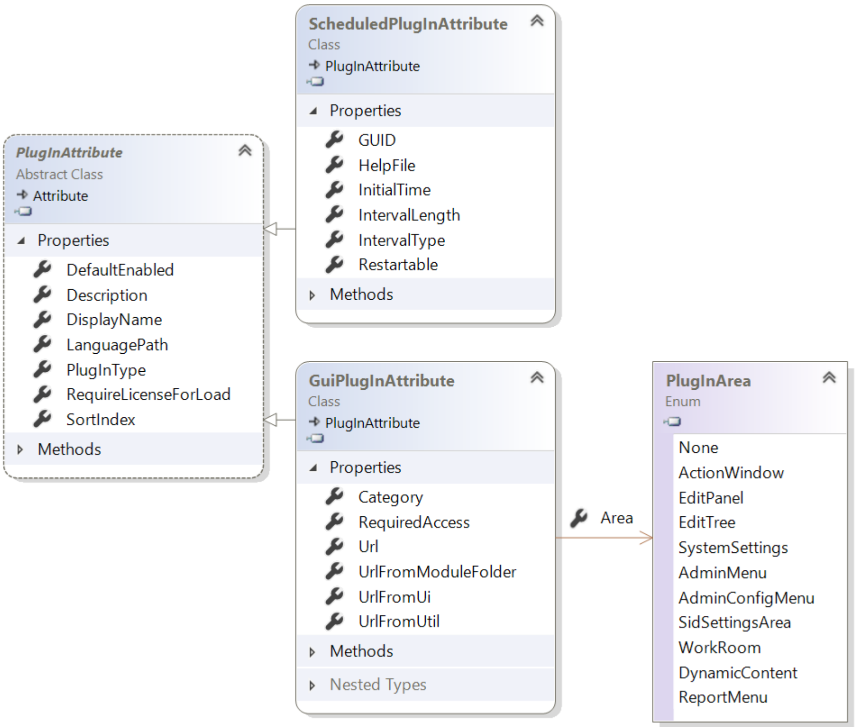
Extending Admin view and Reports with GUI plug-ins

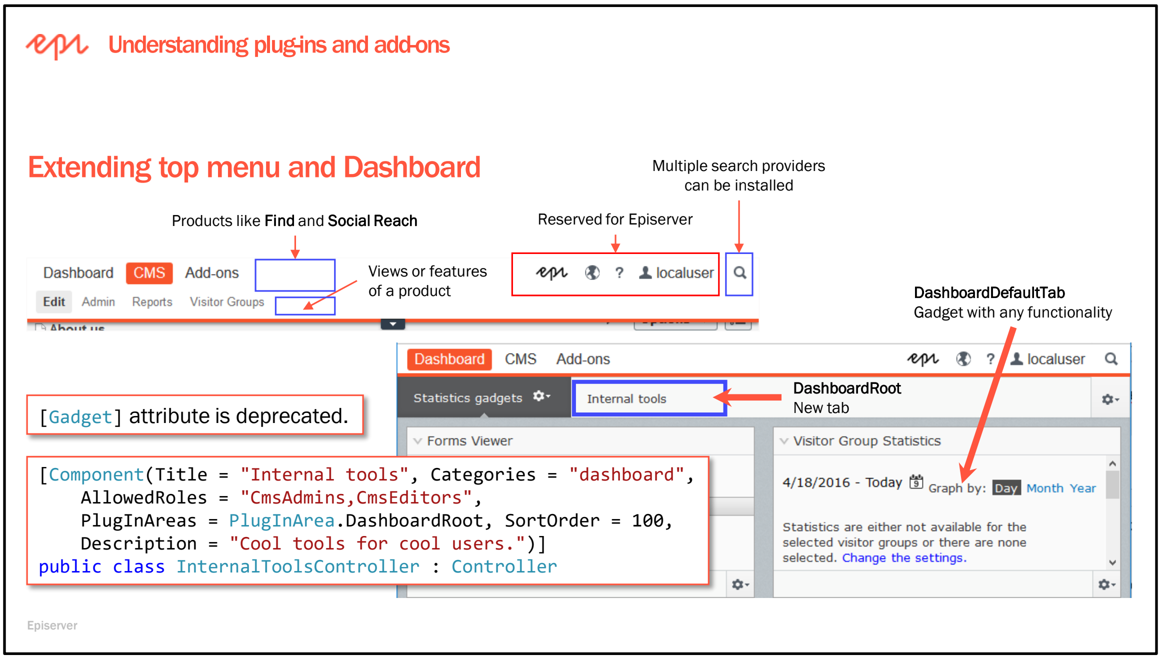
```
[Authorize(Roles = "CmsAdmins")]
[GuiPlugIn(Area = PlugInArea.AdminMenu, DisplayName = ..., SortIndex = ...)]
public class AppSettingsController : Controller
```

```
namespace EPiServer.PlugIn
{
    public enum PlugInArea
    {
        None = 0,
        ActionWindow = 1,
        EditPanel = 2,
        EditTree = 3,
        SystemSettings = 4,
        AdminMenu = 5,
        AdminConfigMenu = 6,
        SidSettingsArea = 7,
        WorkRoom = 8,
        DynamicContent = 9,
        ReportMenu = 10
    }
}
```

✘ obsolete

Episerver





Understanding plug-ins and add-ons

Extending top menu and Dashboard

Products like **Find** and **Social Reach**

Reserved for Episerver

Multiple search providers can be installed

Views or features of a product

DashboardDefaultTab
Gadget with any functionality

DashboardRoot
New tab

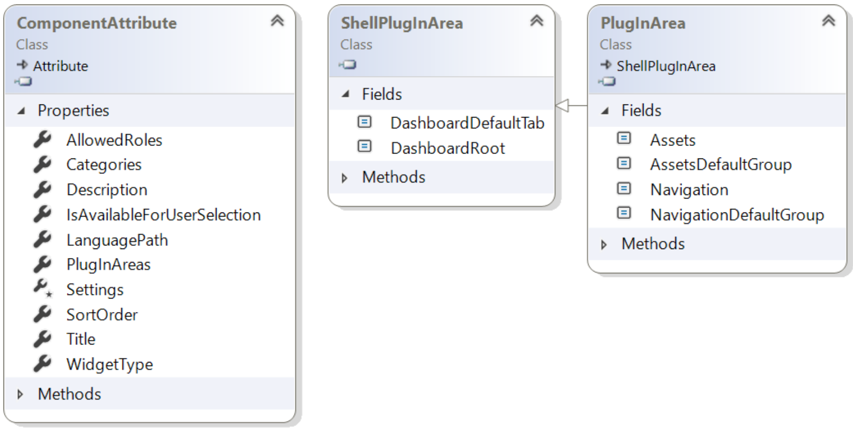
Statistics are either not available for the selected visitor groups or there are none selected. Change the settings.

[Gadget] attribute is deprecated.

```
[Component(Title = "Internal tools", Categories = "dashboard",
    AllowedRoles = "CmsAdmins,CmsEditors",
    PlugInAreas = PlugInArea.DashboardRoot, SortOrder = 100,
    Description = "Cool tools for cool users.")]
public class InternalToolsController : Controller
```

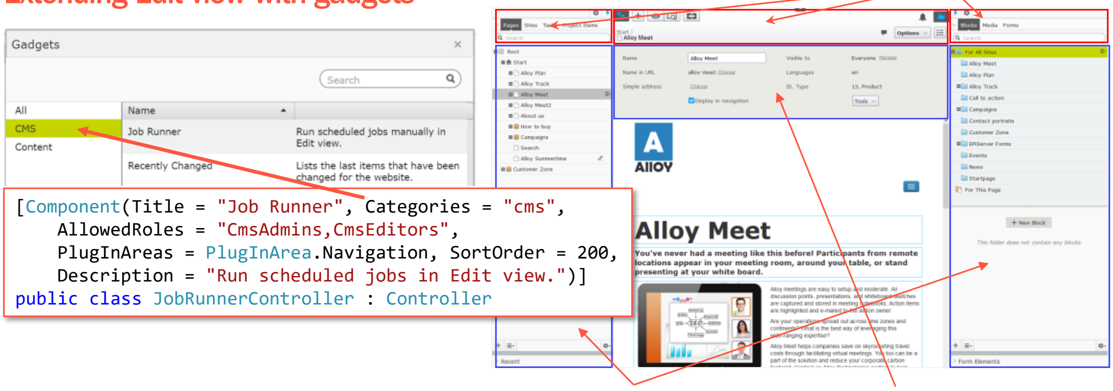
Episerver

Create gadgets with the following classes



Understanding plug-ins and add-ons

Extending Edit view with gadgets



Gadgets

All	Name	
CMS	Job Runner	Run scheduled jobs manually in Edit view.
Content	Recently Changed	Lists the last items that have been changed for the website.

```
[Component(Title = "Job Runner", Categories = "cms",
    AllowedRoles = "CmsAdmins,CmsEditors",
    PlugInAreas = PlugInArea.Navigation, SortOrder = 200,
    Description = "Run scheduled jobs in Edit view.")]
public class JobRunnerController : Controller
```

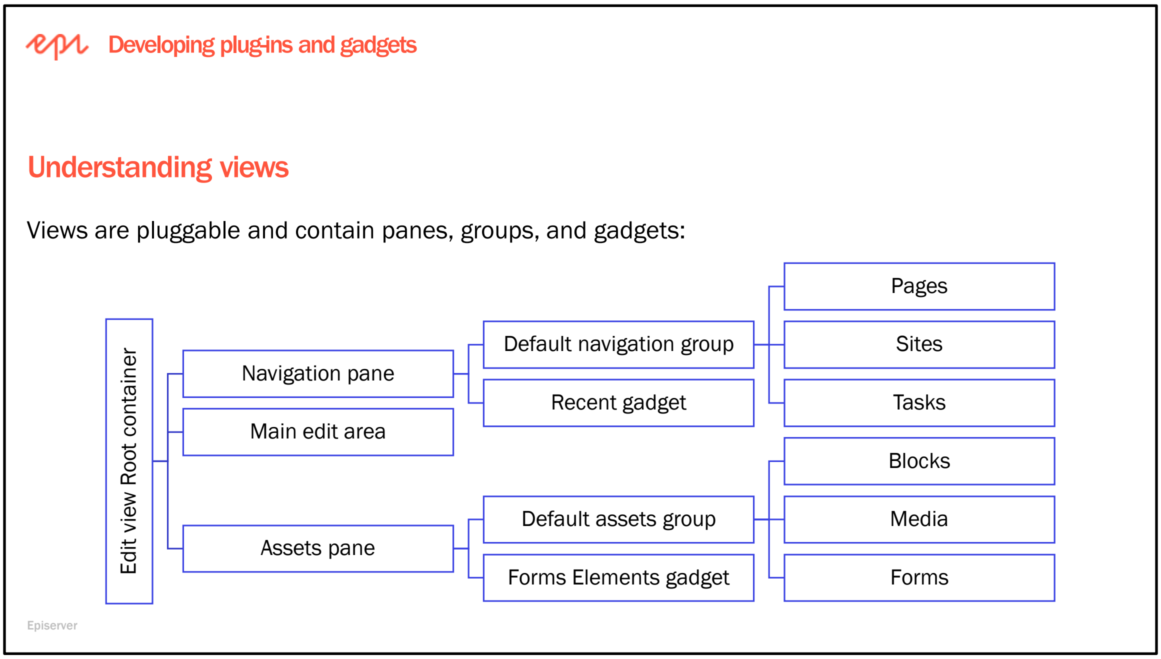
You've never had a meeting like this before! Participants from remote locations appear in your meeting room, around your table, or stand presenting at your white board.

Tab bars and main toolbar
Can be extended but it is not recommended

Asset Pane and Navigation Pane
Information related to content

Settings header
Properties that are frequently used

Episerver



Developing plug-ins and gadgets

Understanding views

Views are pluggable and contain panes, groups, and gadgets:

Edit view Root container

Navigation pane

Main edit area

Assets pane

Default navigation group

Recent gadget

Default assets group

Forms Elements gadget

Pages

Sites

Tasks

Blocks

Media

Forms

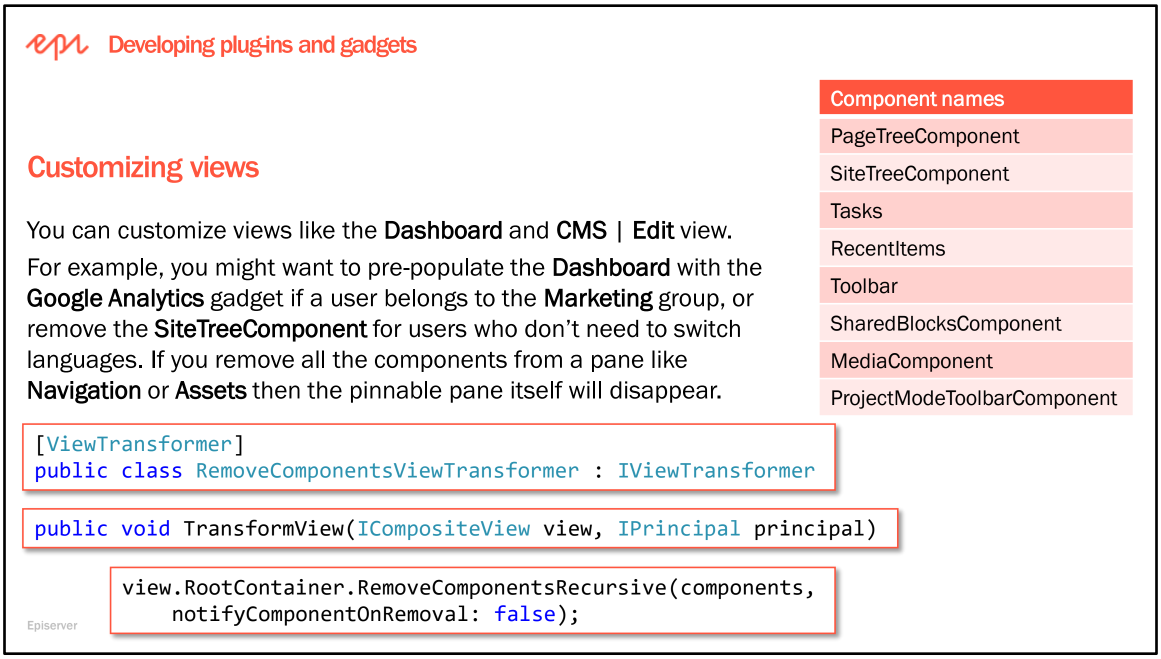
Episerver

Episerver CMS - Edit

```
View: /episerver/cms/home, Title: EPiServer CMS - Edit, 3/29/2018 10:28:00 AM
  Container: EPiServer.Shell.ViewComposition.Containers.BorderContainer, dijit/layout/BorderContainer
    Container: EPiServer.Shell.ViewComposition.Containers.PinnablePane, epi/shell/layout/PinnablePane
      Container: EPiServer.Shell.ViewComposition.Containers.ComponentPaneContainer, epi/shell/widget/layout/ComponentPaneContainer
        Container: EPiServer.Shell.ViewComposition.Containers.ComponentGroup, epi/shell/widget/layout/ComponentTabContainer
          Component: EPiServer.Cms.Shell.UI.Components.PageTreeComponent, epi-cms/component/MainNavigationComponent
          Component: EPiServer.Cms.Shell.UI.Components.SiteTreeComponent, epi-cms/component/SiteTree
          Component: EPiServer.Cms.Shell.UI.Components.Tasks, epi-cms/component/Tasks
        Component: EPiServer.Cms.Shell.UI.Components.RecentItems, epi-cms/component/ContextHistory
    Container: EPiServer.Shell.ViewComposition.Containers.BorderContainer, dijit/layout/BorderContainer
      Container: EPiServer.Shell.ViewComposition.Containers.ContentPane, dijit/layout/ContentPane
        Component: EPiServer.Cms.Shell.UI.Components.Toolbar, epi-cms/component/GlobalToolbar
      Container: EPiServer.Shell.ViewComposition.Containers.ContentPane, dijit/layout/ContentPane
        Component: EPiServer.Cms.Shell.UI.Components.WidgetSwitcher, epi/shell/widget/WidgetSwitcher
    Container: EPiServer.Shell.ViewComposition.Containers.PinnablePane, epi/shell/layout/PinnablePane
      Container: EPiServer.Shell.ViewComposition.Containers.ComponentPaneContainer, epi/shell/widget/layout/ComponentPaneContainer
        Container: EPiServer.Shell.ViewComposition.Containers.ComponentGroup, epi/shell/widget/layout/ComponentTabContainer
          Component: EPiServer.Cms.Shell.UI.Components.MediaComponent, epi-cms/component/Media
          Component: EPiServer.Cms.Shell.UI.Components.SharedBlocksComponent, epi-cms/component/SharedBlocks
    Container: EPiServer.Shell.ViewComposition.Containers.ContentPane, dijit/layout/ContentPane
      Component: EPiServer.Cms.Shell.UI.Components.ProjectModeToolbarComponent, epi-cms/project/ProjectModeToolbar
```

Episerver - Dashboard

```
View: /episerver/dashboard, Title: EPiServer - Dashboard, 3/29/2018 10:29:41 AM
  Container: EPiServer.Shell.ViewComposition.Containers.BorderContainer, dijit/layout/BorderContainer
    Container: EPiServer.Shell.ViewComposition.Containers.TabContainer, epi/shell/widget/TabContainer
      Container: EPiServer.Shell.ViewComposition.Containers.ComponentContainer, epi/shell/widget/layout/ComponentContainer
```



Developing plug-ins and gadgets

Customizing views

You can customize views like the **Dashboard** and **CMS | Edit** view.

For example, you might want to pre-populate the **Dashboard** with the **Google Analytics** gadget if a user belongs to the **Marketing** group, or remove the **SiteTreeComponent** for users who don't need to switch languages. If you remove all the components from a pane like **Navigation** or **Assets** then the pinnable pane itself will disappear.

Component names
PageTreeComponent
SiteTreeComponent
Tasks
RecentItems
Toolbar
SharedBlocksComponent
MediaComponent
ProjectModeToolbarComponent

```
[ViewTransformer]
public class RemoveComponentsViewTransformer : IViewTransformer
```

```
public void TransformView(ICompositeView view, IPrincipal principal)
```

```
view.RootContainer.RemoveComponentsRecursive(components,
    notifyComponentOnRemoval: false);
```

Episerver

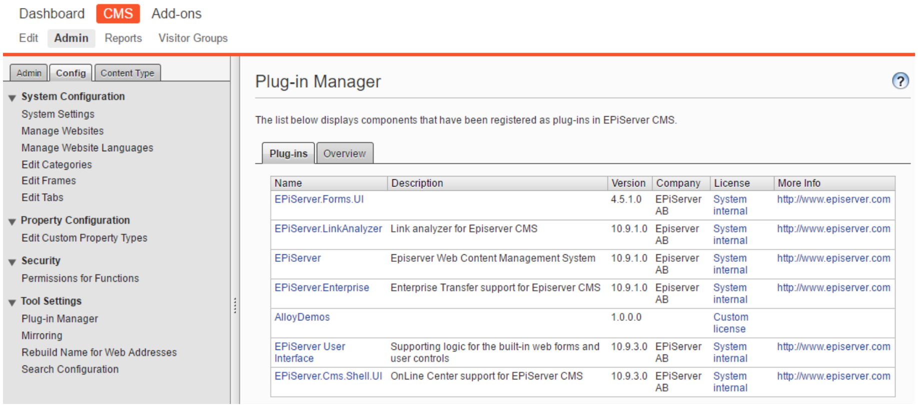
Learn more from these articles:
https://world.episerver.com/blogs/Ben-McKernan/Dates/2015/6/modifying-the-default-view-components/
https://www.david-tec.com/2016/05/remove-episerver-ui-components-for-certain-editors/
https://world.episerver.com/blogs/Linus-Ekstrom/Dates/2013/2/Modifying-the-EPiServer-UI-views/

Developing plug-ins and gadgets

Plug-in manager

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

Shows Episerver CMS version e.g. 10.9.1.0, and other shell modules deployed as plug-ins.

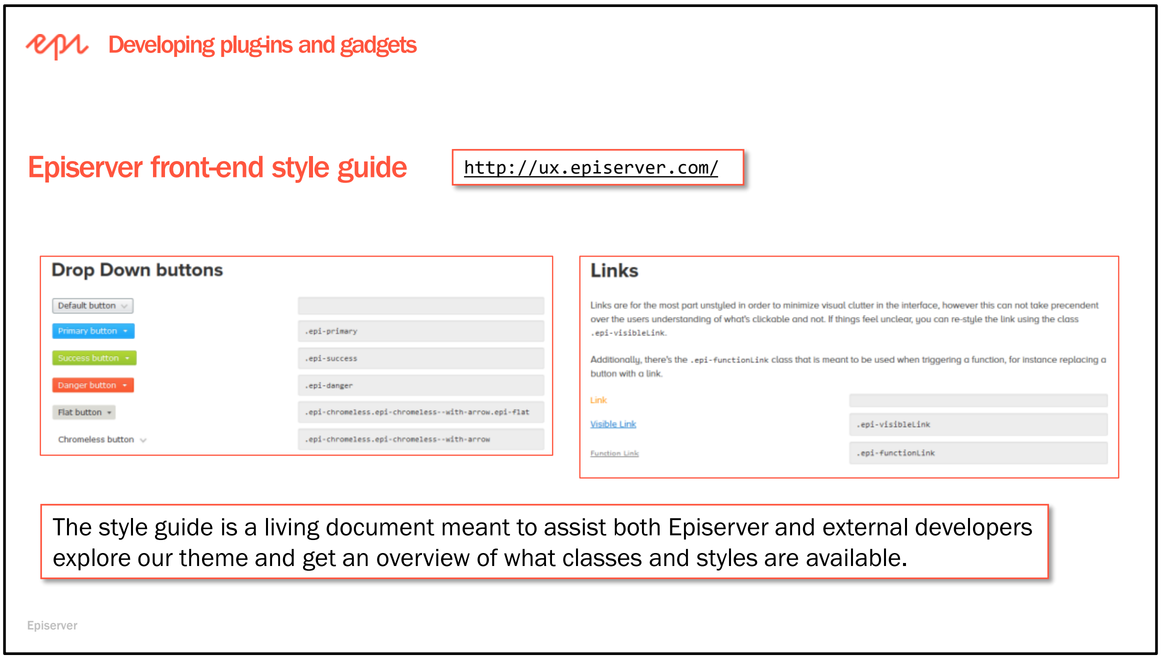


Plug-in Manager

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

Name	Description	Version	Company	License	More Info
EPiServer.Forms.UI		4.5.1.0	EPiServer AB	System internal	http://www.episerver.com
EPiServer.LinkAnalyzer	Link analyzer for Episerver CMS	10.9.1.0	Episerver AB	System internal	http://www.episerver.com
EPiServer	Episerver Web Content Management System	10.9.1.0	Episerver AB	System internal	http://www.episerver.com
EPiServer.Enterprise	Enterprise Transfer support for Episerver CMS	10.9.1.0	Episerver AB	System internal	http://www.episerver.com
AlloyDemos		1.0.0.0		Custom license	
EPiServer User Interface	Supporting logic for the built-in web forms and user controls	10.9.3.0	EPiServer AB	System internal	http://www.episerver.com
EPiServer.Cms.Shell.UI	OnLine Center support for EPiServer CMS	10.9.3.0	EPiServer AB	System internal	http://www.episerver.com

Episerver



Developing plug-ins and gadgets

Episerver front-end style guide

http://ux.episerver.com/

Drop Down buttons

Default button	
Primary button	.epi-primary
Success button	.epi-success
Danger button	.epi-danger
Flat button	.epi-chromeless.epi-chromeless--with-arrow.epi-flat
Chromeless button	.epi-chromeless.epi-chromeless--with-arrow

Links

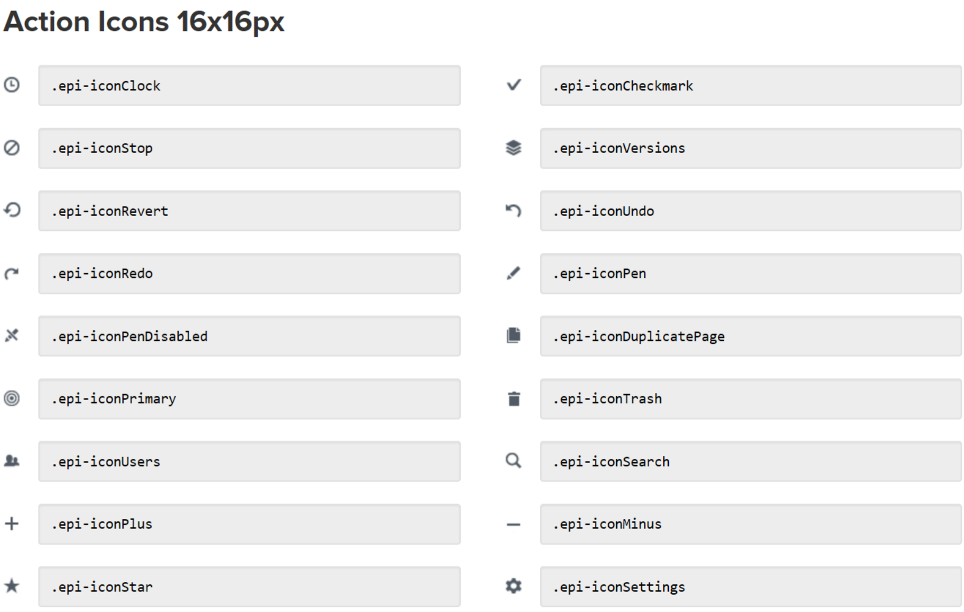
Links are for the most part unstyled in order to minimize visual clutter in the interface, however this can not take precendent over the users understanding of what's clickable and not. If things feel unclear, you can re-style the link using the class .epi-visibleLink.

Additionally, there's the .epi-functionLink class that is meant to be used when triggering a function, for instance replacing a button with a link.

Link	
Visible Link	.epi-visibleLink
Function Link	.epi-functionLink

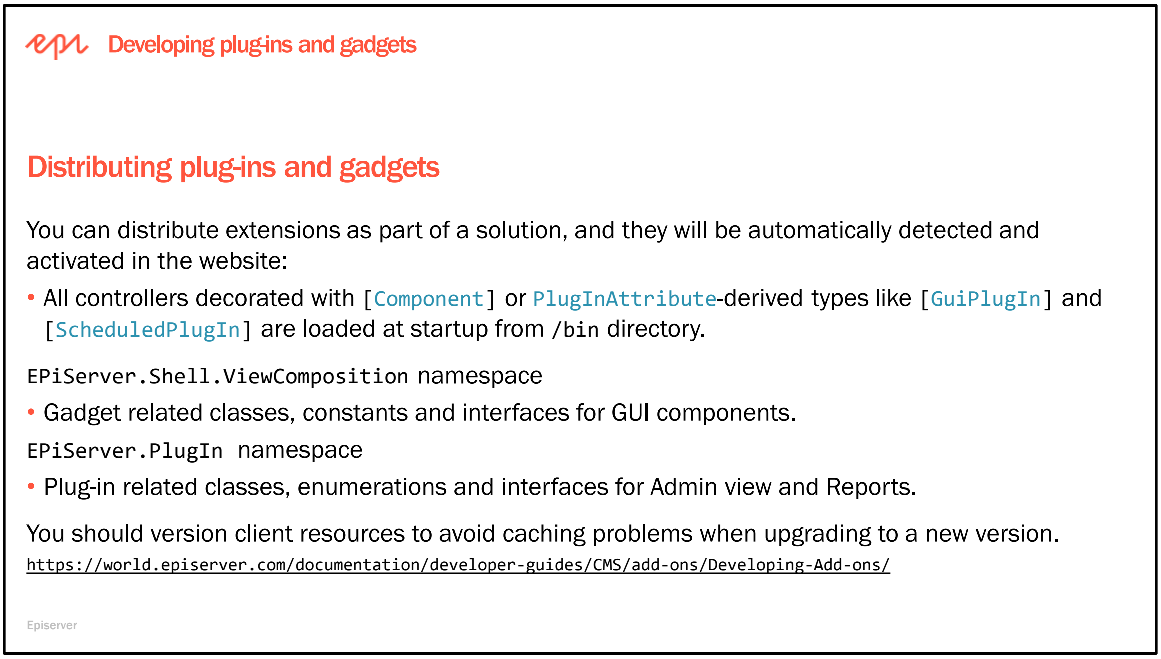
The style guide is a living document meant to assist both Episerver and external developers explore our theme and get an overview of what classes and styles are available.

Episerver



Action Icons 16x16px

🕒	.epi-iconClock	✓	.epi-iconCheckmark
⊘	.epi-iconStop	≋	.epi-iconVersions
↺	.epi-iconRevert	↶	.epi-iconUndo
↷	.epi-iconRedo	✎	.epi-iconPen
✗	.epi-iconPenDisabled	⧉	.epi-iconDuplicatePage
◎	.epi-iconPrimary	🗑	.epi-iconTrash
👥	.epi-iconUsers	🔍	.epi-iconSearch
+	.epi-iconPlus	−	.epi-iconMinus
★	.epi-iconStar	⚙	.epi-iconSettings



Developing plug-ins and gadgets

Distributing plug-ins and gadgets

You can distribute extensions as part of a solution, and they will be automatically detected and activated in the website:

- All controllers decorated with `[Component]` or `PlugInAttribute`-derived types like `[GuiPlugIn]` and `[ScheduledPlugIn]` are loaded at startup from `/bin` directory.

`EPiServer.Shell.ViewComposition` namespace

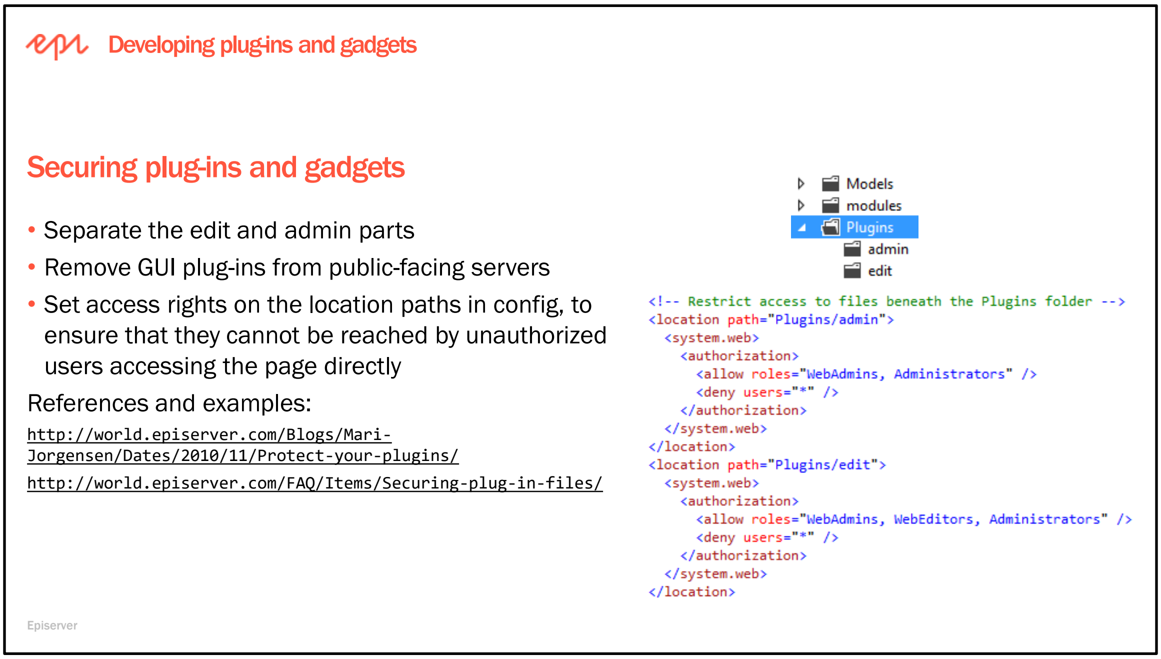
- Gadget related classes, constants and interfaces for GUI components.

`EPiServer.PlugIn` namespace

- Plug-in related classes, enumerations and interfaces for Admin view and Reports.

You should version client resources to avoid caching problems when upgrading to a new version.
https://world.episerver.com/documentation/developer-guides/CMS/add-ons/Developing-Add-ons/

Episerver



Developing plug-ins and gadgets

Securing plug-ins and gadgets

- Separate the edit and admin parts
- Remove GUI plug-ins from public-facing servers
- Set access rights on the location paths in config, to ensure that they cannot be reached by unauthorized users accessing the page directly

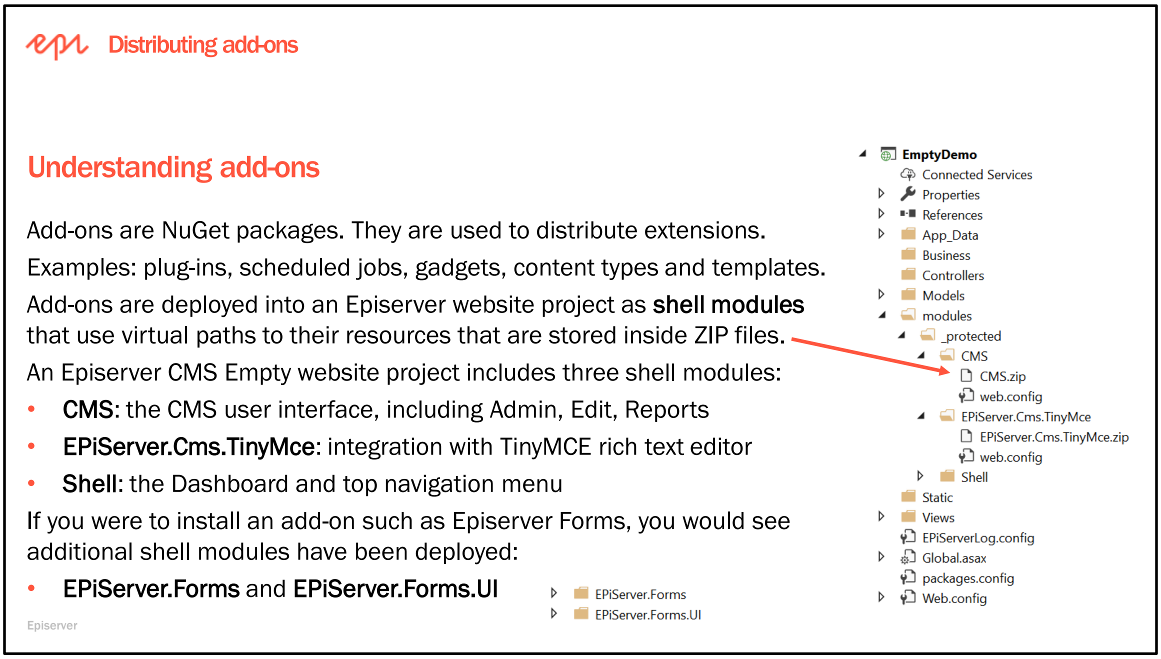
References and examples:

http://world.episerver.com/Blogs/Mari-Jorgensen/Dates/2010/11/Protect-your-plugins/

http://world.episerver.com/FAQ/Items/Securing-plug-in-files/

```
<!-- Restrict access to files beneath the Plugins folder -->
<location path="Plugins/admin">
  <system.web>
    <authorization>
      <allow roles="WebAdmins, Administrators" />
      <deny users="*" />
    </authorization>
  </system.web>
</location>
<location path="Plugins/edit">
  <system.web>
    <authorization>
      <allow roles="WebAdmins, WebEditors, Administrators" />
      <deny users="*" />
    </authorization>
  </system.web>
</location>
```

Episerver



Distributing add-ons

Understanding add-ons

Add-ons are NuGet packages. They are used to distribute extensions.

Examples: plug-ins, scheduled jobs, gadgets, content types and templates.

Add-ons are deployed into an Episerver website project as **shell modules** that use virtual paths to their resources that are stored inside ZIP files.

An Episerver CMS Empty website project includes three shell modules:

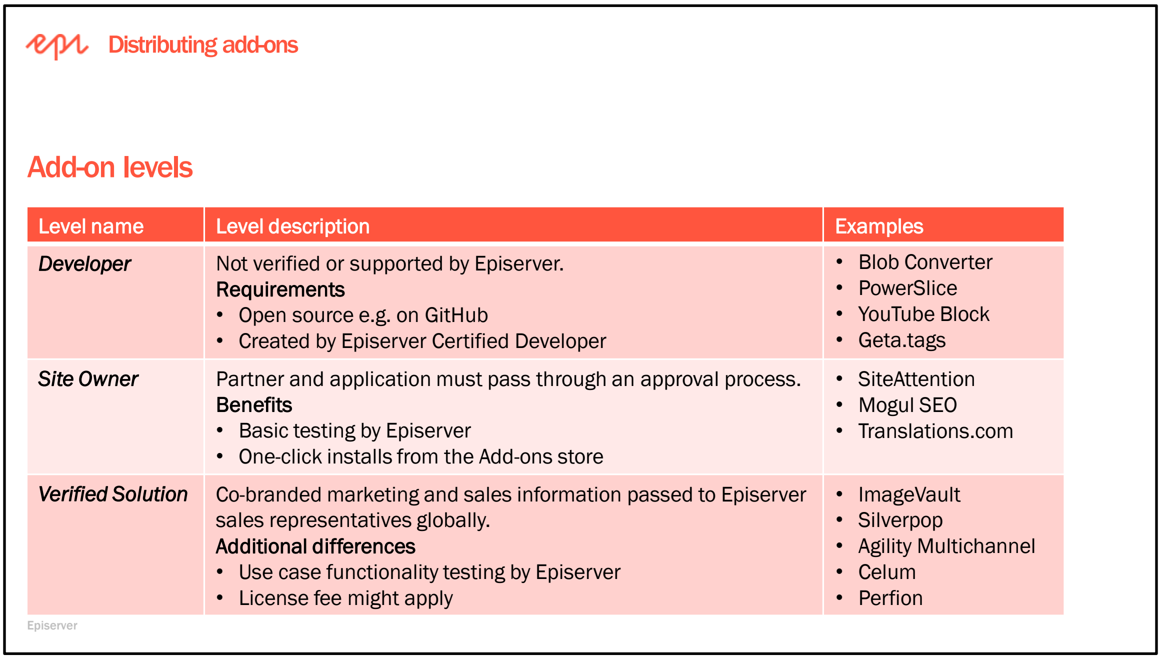
- **CMS**: the CMS user interface, including Admin, Edit, Reports
- **EPiServer.Cms.TinyMce**: integration with TinyMCE rich text editor
- **Shell**: the Dashboard and top navigation menu

If you were to install an add-on such as Episerver Forms, you would see additional shell modules have been deployed:

- **EPiServer.Forms** and **EPiServer.Forms.UI**

How to package add-ons video (90 minutes):
http://fast.wistia.net/embed/iframe/4dhm5342lt?videoFoam=true

Create, update and deploy Nuget Packages with a GUI
https://github.com/NuGetPackageExplorer/NuGetPackageExplorer

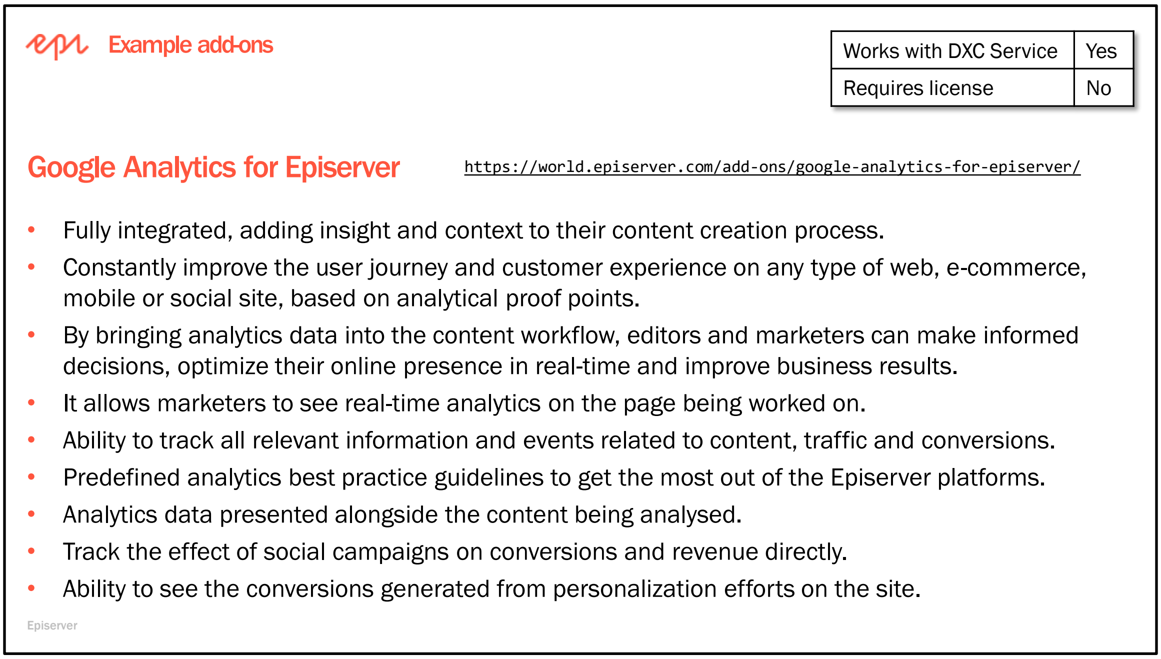


Distributing add-ons

Add-on levels

Level name	Level description	Examples
Developer	Not verified or supported by Episerver. **Requirements** • Open source e.g. on GitHub • Created by Episerver Certified Developer	• Blob Converter • PowerSlice • YouTube Block • Geta.tags
Site Owner	Partner and application must pass through an approval process. **Benefits** • Basic testing by Episerver • One-click installs from the Add-ons store	• SiteAttention • Mogul SEO • Translations.com
Verified Solution	Co-branded marketing and sales information passed to Episerver sales representatives globally. **Additional differences** • Use case functionality testing by Episerver • License fee might apply	• ImageVault • Silverpop • Agility Multichannel • Celum • Perfion

Episerver



Example add-ons

Works with DXC Service	Yes
Requires license	No

Google Analytics for Episerver

https://world.episerver.com/add-ons/google-analytics-for-episerver/

- Fully integrated, adding insight and context to their content creation process.
- Constantly improve the user journey and customer experience on any type of web, e-commerce, mobile or social site, based on analytical proof points.
- By bringing analytics data into the content workflow, editors and marketers can make informed decisions, optimize their online presence in real-time and improve business results.
- It allows marketers to see real-time analytics on the page being worked on.
- Ability to track all relevant information and events related to content, traffic and conversions.
- Predefined analytics best practice guidelines to get the most out of the Episerver platforms.
- Analytics data presented alongside the content being analysed.
- Track the effect of social campaigns on conversions and revenue directly.
- Ability to see the conversions generated from personalization efforts on the site.

Episerver

Example add-ons

Works with DXC Service	Yes
Requires license	No

Episerver Social Reach

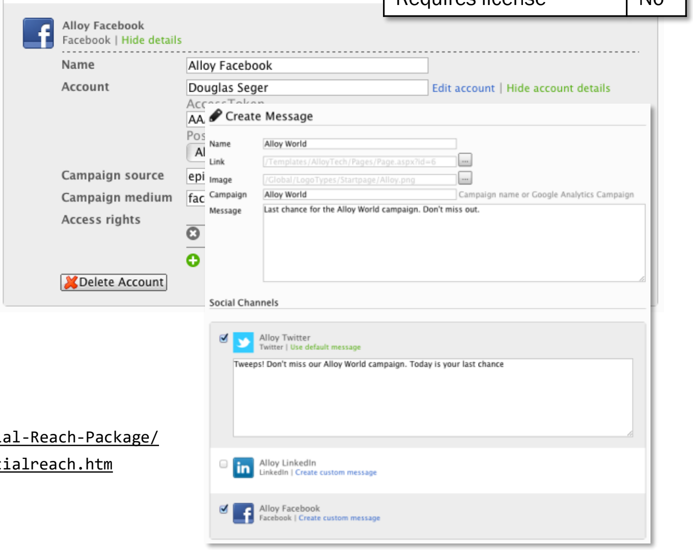
With **Episerver Social Reach** you can set up your social channels and configure which editors and marketers can use them.

When an article or product is to be promoted, create a social message and decide which social channels you want to target it to.

Introducing EPiServer Social Reach

http://world.episerver.com/Articles/Items/Social-Reach-Package/

http://webhelp.episerver.com/latest/addons/socialreach.htm



Alloy Facebook
Facebook | Hide details

Name: Alloy Facebook
Account: Douglas Seger — Edit account | Hide account details
AccessToken: AA
Pos
Al
Campaign source: epi
Campaign medium: fac
Access rights

Delete Account

Create Message

Name: Alloy World
Link: /Templates/AlloyTech/Pages/Page.aspx?id=6
Image: /Global/LogoTypes/Startpage/Alloy.png
Campaign: Alloy World — Campaign name or Google Analytics Campaign
Message: Last chance for the Alloy World campaign. Don't miss out.

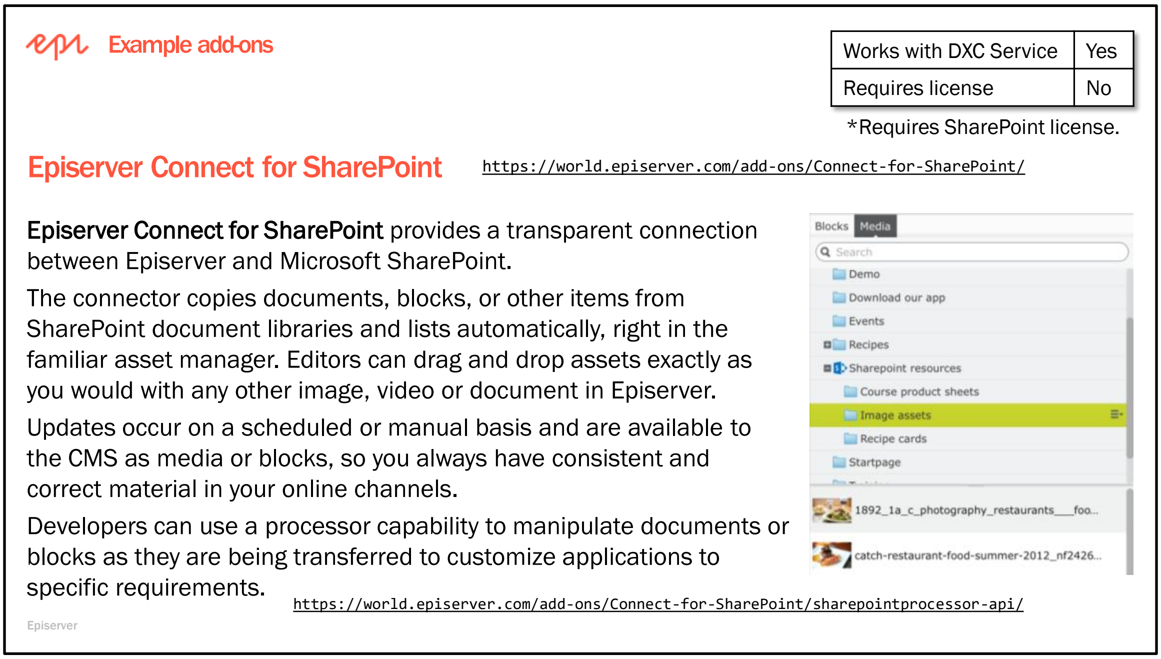
Social Channels

☑ Alloy Twitter
Twitter | Use default message
Tweeps! Don't miss our Alloy World campaign. Today is your last chance

□ Alloy LinkedIn
LinkedIn | Create custom message

☑ Alloy Facebook
Facebook | Create custom message

Episerver



Example add-ons

Works with DXC Service	Yes
Requires license	No

*Requires SharePoint license.

Episerver Connect for SharePoint

https://world.episerver.com/add-ons/Connect-for-SharePoint/

Episerver Connect for SharePoint provides a transparent connection between Episerver and Microsoft SharePoint.

The connector copies documents, blocks, or other items from SharePoint document libraries and lists automatically, right in the familiar asset manager. Editors can drag and drop assets exactly as you would with any other image, video or document in Episerver.

Updates occur on a scheduled or manual basis and are available to the CMS as media or blocks, so you always have consistent and correct material in your online channels.

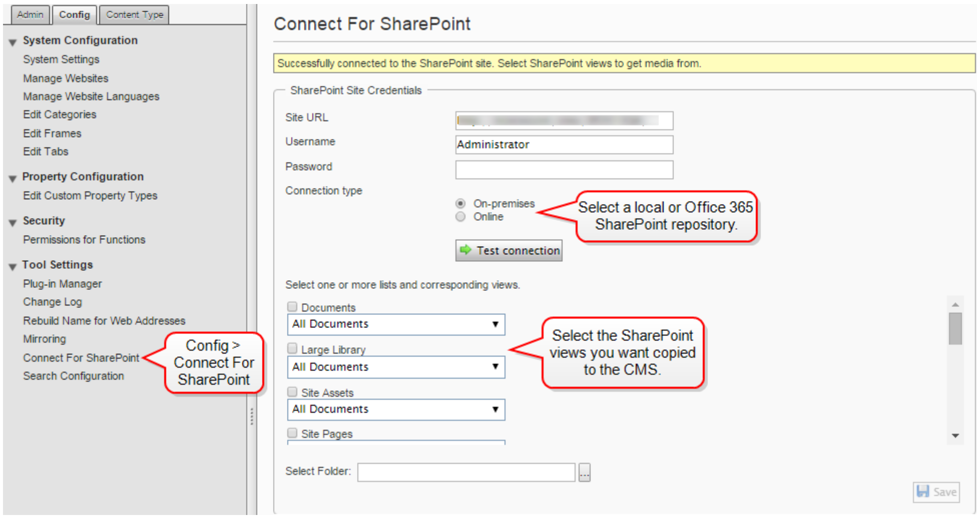
Developers can use a processor capability to manipulate documents or blocks as they are being transferred to customize applications to specific requirements.

https://world.episerver.com/add-ons/Connect-for-SharePoint/sharepointprocessor-api/

Episerver

Episerver Connect for SharePoint in User Guide

http://webhelp.episerver.com/15-5/EN/addons/sharepoint.htm



Config > Connect For SharePoint

Connect For SharePoint

Successfully connected to the SharePoint site. Select SharePoint views to get media from.

SharePoint Site Credentials

Site URL	[redacted]
Username	Administrator
Password	
Connection type	● On-premises ○ Online

Select a local or Office 365 SharePoint repository.

Test connection

Select one or more lists and corresponding views.

☐ Documents
All Documents ▼

☐ Large Library
All Documents ▼

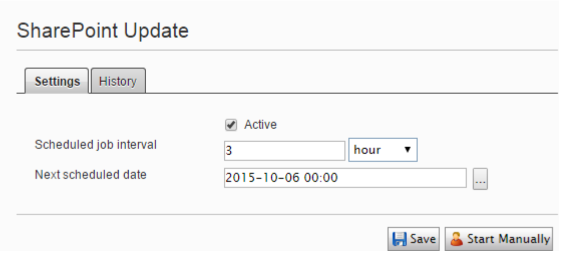
☐ Site Assets
All Documents ▼

☐ Site Pages

Select the SharePoint views you want copied to the CMS.

Select Folder: ...

Save



SharePoint Update

☑ Active

Scheduled job interval	3 hour ▼
Next scheduled date	2015-10-06 00:00 ...

Save Start Manually

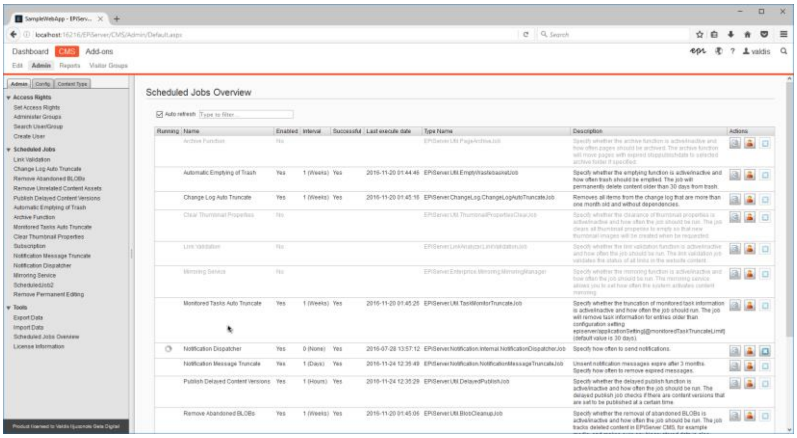
Example add-ons

Works with DXC Service	Yes
Requires license	No

TechFellow ScheduledJobOverview

Gives you an easy way to tell details of the job, which of them is enabled, which of them failed last time, what is the schedule interval, and so on.

Open source on GitHub so you can learn how to build your own Admin view plug-ins.



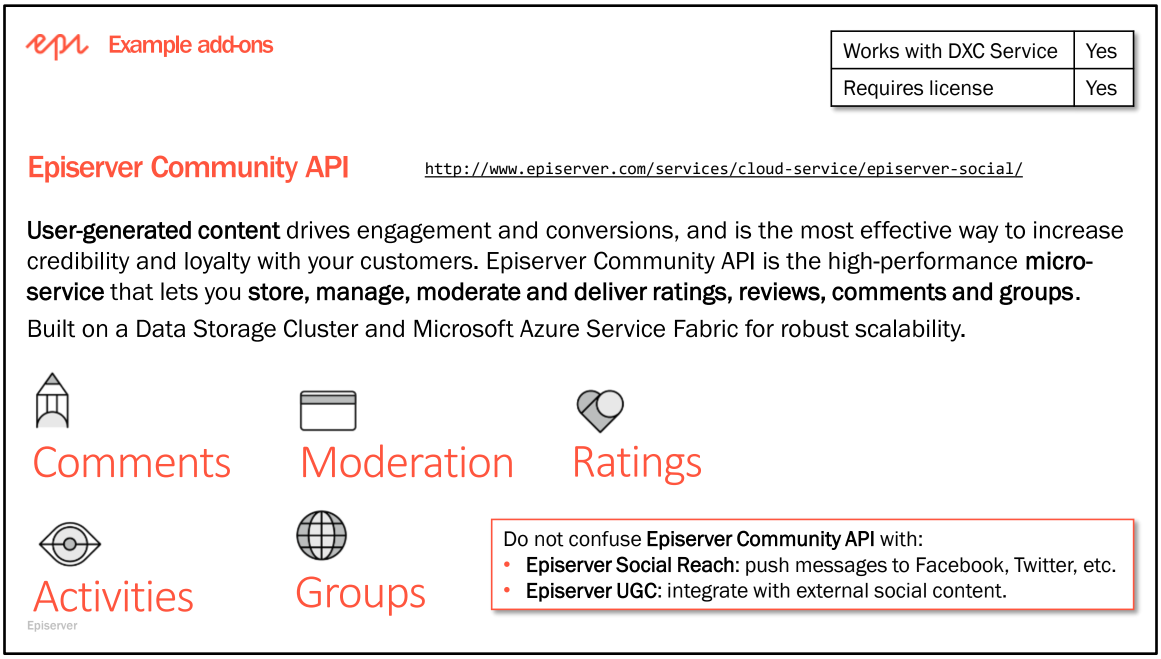
Scheduled Jobs Overview

☑ Auto refresh Type to filter...

Running	Name	Enabled	Interval	Successful	Last execute date	Type Name	Description	Actions
	Archive Function	No				EPiServer.Util.PageArchiveJob	Specify whether the archive function is active/inactive and how often pages should be archived. The archive function will move pages with expired stoppublishdate to selected archive folder if specified.	
	Automatic Emptying of Trash	Yes	1 (Weeks)	Yes	2016-11-20 01:44:46	EPiServer.Util.EmptyWastebasketJob	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.	
	Change Log Auto Truncate	Yes	1 (Weeks)	Yes	2016-11-20 01:45:16	EPiServer.ChangeLog.ChangeLogAutoTruncateJob	Removes all items from the change log that are more than one month old and without dependencies.	
	Clear Thumbnail Properties	No				EPiServer.Util.ThumbnailPropertiesClearJob	Specify whether the clearance of thumbnail properties is active/inactive and how often the job should be run. The job clears all thumbnail properties to empty so that new thumbnail images will be created when be requested.	
	Link Validation	No				EPiServer.LinkAnalyzer.LinkValidationJob	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.	
	Mirroring Service	No				EPiServer.Enterprise.Mirroring.MirroringManager	Specify whether the mirroring function is active/inactive and how often the job should be run. The mirroring service allows you to set how often the system activates content mirroring.	
	Monitored Tasks Auto Truncate	Yes	1 (Weeks)	Yes	2016-11-20 01:45:26	EPiServer.Util.TaskMonitorTruncateJob	Specify whether the truncation of monitored task information is active/inactive and how often the job should run. The job will remove task information for entries older than configuration setting episerver/applicationSettings[@monitoredTaskTruncateLimit] (default value is 30 days).	
◌	Notification Dispatcher	Yes	0 (None)	Yes	2016-07-28 13:57:12	EPiServer.Notification.Internal.NotificationDispatcherJob	Specify how often to send notifications.	
	Notification Message Truncate	Yes	1 (Days)	Yes	2016-11-24 12:35:49	EPiServer.Notification.NotificationMessageTruncateJob	Unsent notification messages expire after 3 months. Specify how often to remove expired messages.	
	Publish Delayed Content Versions	Yes	1 (Hours)	Yes	2016-11-24 12:35:29	EPiServer.Util.DelayedPublishJob	Specify whether the delayed publish function is active/inactive and how often the job should be run. The delayed publish job checks if there are content versions that are set to be published at a certain time.	
	Remove Abandoned BLOBs	Yes	1 (Weeks)	Yes	2016-11-20 01:45:06	EPiServer.Util.BlobCleanupJob	Specify whether the removal of abandoned BLOBs is active/inactive and how often the job should be run. The job tracks deleted content in EPiServer CMS, for example	

https://github.com/valdisiljuconoks/TechFellow.ScheduledJobOverview/blob/master/README.md

Episerver



Example add-ons

Works with DXC Service	Yes
Requires license	Yes

Episerver Community API

http://www.episerver.com/services/cloud-service/episerver-social/

User-generated content drives engagement and conversions, and is the most effective way to increase credibility and loyalty with your customers. Episerver Community API is the high-performance **micro-service** that lets you **store, manage, moderate and deliver ratings, reviews, comments and groups.**

Built on a Data Storage Cluster and Microsoft Azure Service Fabric for robust scalability.

Comments

Moderation

Ratings

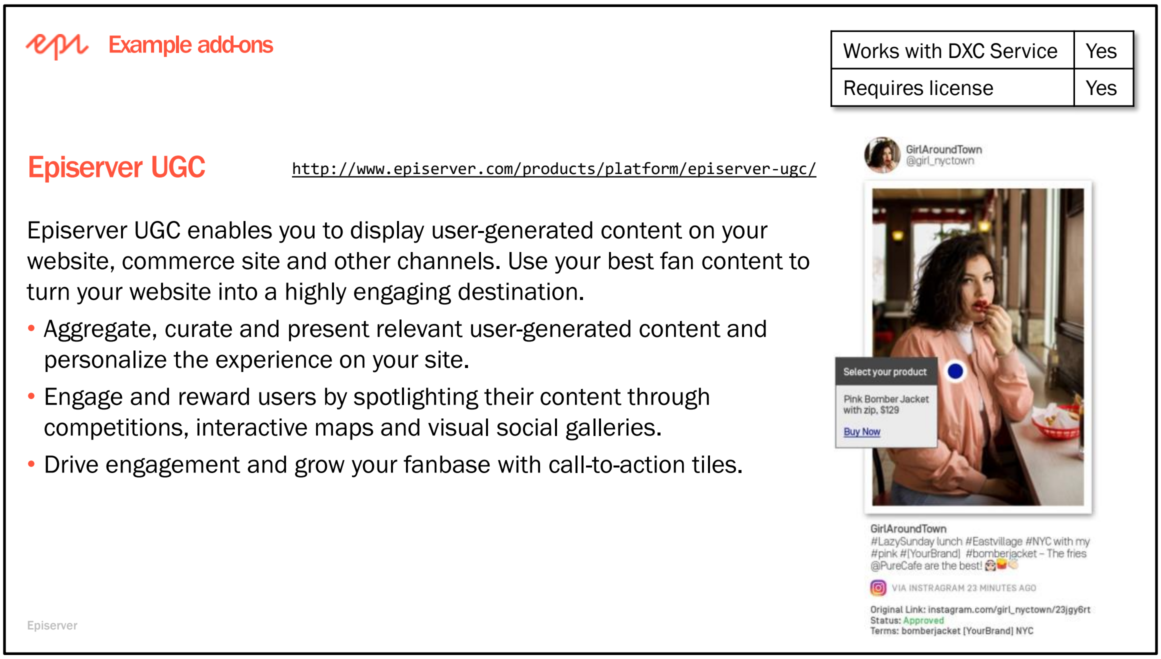
Activities

Groups

Do not confuse **Episerver Community API** with:

- **Episerver Social Reach:** push messages to Facebook, Twitter, etc.
- **Episerver UGC:** integrate with external social content.

Episerver



Example add-ons

Works with DXC Service	Yes
Requires license	Yes

Episerver UGC

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

Episerver UGC enables you to display user-generated content on your website, commerce site and other channels. Use your best fan content to turn your website into a highly engaging destination.

- Aggregate, curate and present relevant user-generated content and personalize the experience on your site.
- Engage and reward users by spotlighting their content through competitions, interactive maps and visual social galleries.
- Drive engagement and grow your fanbase with call-to-action tiles.

GirlAroundTown
@girl_nyctown

Select your product

Pink Bomber Jacket with zip, $129

Buy Now

GirlAroundTown
#LazySunday lunch #Eastvillage #NYC with my #pink #[YourBrand] #bomberjacket – The fries @PureCafe are the best! 🙈🍟👏

VIA INSTRAGRAM 23 MINUTES AGO

Original Link: instagram.com/girl_nyctown/23jgy6rt
Status: Approved
Terms: bomberjacket [YourBrand] NYC

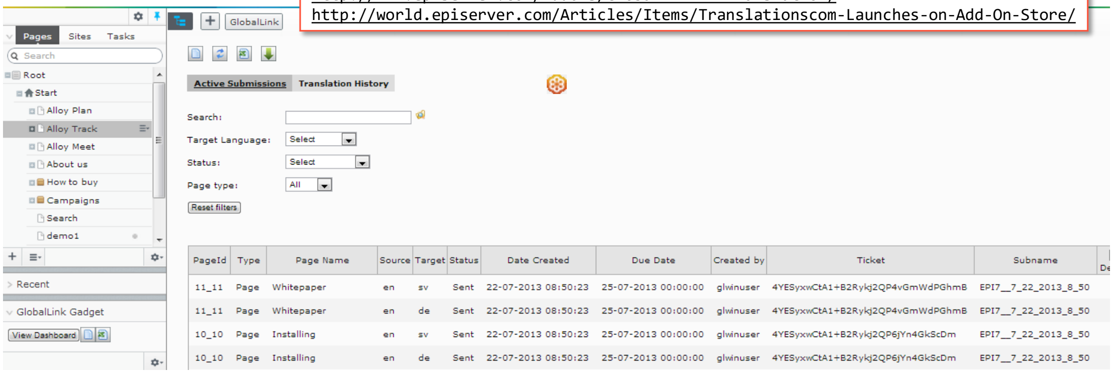
Episerver

Example add-ons

Works with DXC Service	Yes
Requires license	Yes

GlobalLink® Localization Suite from translation.com

http://translations.com/products/globallink-episerver-adaptor
http://www.episerver.com/AddOns/GlobalLink---Translation/
http://world.episerver.com/Articles/Items/Translationscom-Launches-on-Add-On-Store/



Search:

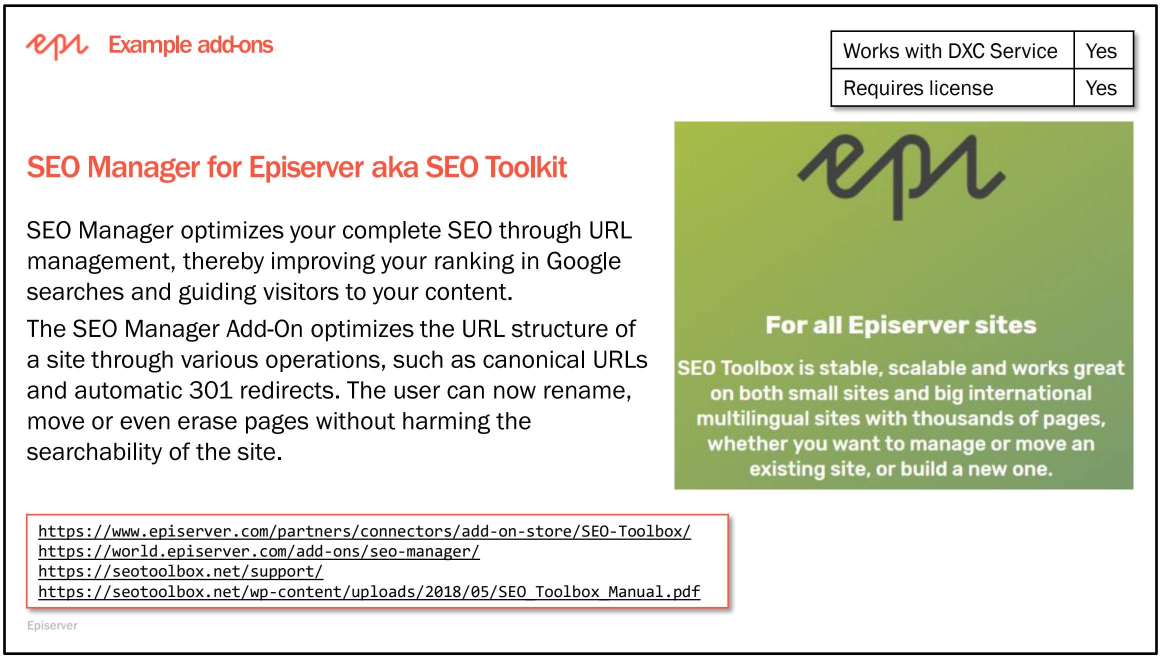
Target Language: Select

Status: Select

Page type: All

PageId	Type	Page Name	Source	Target	Status	Date Created	Due Date	Created by	Ticket	Subname
11_11	Page	Whitepaper	en	sv	Sent	22-07-2013 08:50:23	25-07-2013 00:00:00	glwinuser	4YESyxwCtA1+B2Rykj2QP4vGmWdPGhmB	EPI7__7_22_2013_8_50
11_11	Page	Whitepaper	en	de	Sent	22-07-2013 08:50:23	25-07-2013 00:00:00	glwinuser	4YESyxwCtA1+B2Rykj2QP4vGmWdPGhmB	EPI7__7_22_2013_8_50
10_10	Page	Installing	en	sv	Sent	22-07-2013 08:50:23	25-07-2013 00:00:00	glwinuser	4YESyxwCtA1+B2Rykj2QP6jYn4GkScDm	EPI7__7_22_2013_8_50
10_10	Page	Installing	en	de	Sent	22-07-2013 08:50:23	25-07-2013 00:00:00	glwinuser	4YESyxwCtA1+B2Rykj2QP6jYn4GkScDm	EPI7__7_22_2013_8_50

Episerver



Example add-ons

Works with DXC Service	Yes
Requires license	Yes

SEO Manager for Episerver aka SEO Toolkit

SEO Manager optimizes your complete SEO through URL management, thereby improving your ranking in Google searches and guiding visitors to your content.

The SEO Manager Add-On optimizes the URL structure of a site through various operations, such as canonical URLs and automatic 301 redirects. The user can now rename, move or even erase pages without harming the searchability of the site.

For all Episerver sites

SEO Toolbox is stable, scalable and works great on both small sites and big international multilingual sites with thousands of pages, whether you want to manage or move an existing site, or build a new one.

https://www.episerver.com/partners/connectors/add-on-store/SEO-Toolbox/
https://world.episerver.com/add-ons/seo-manager/
https://seotoolbox.net/support/
https://seotoolbox.net/wp-content/uploads/2018/05/SEO_Toolbox_Manual.pdf

Episerver

Consolidate duplicate URLs - Define a canonical page for similar or duplicate pages

”If you have a single page accessible by multiple URLs, or different pages with similar content (for example, a page with both a mobile and a desktop version), Google sees these as duplicate versions of the same page. Google will choose one URL as the canonical version and crawl that, and all other URLs will be considered duplicate URLs and crawled less often. If you don't explicitly tell Google which URL is canonical, Google will make the choice for you, or might consider them both of equal weight, which might lead to unwanted behaviour.”

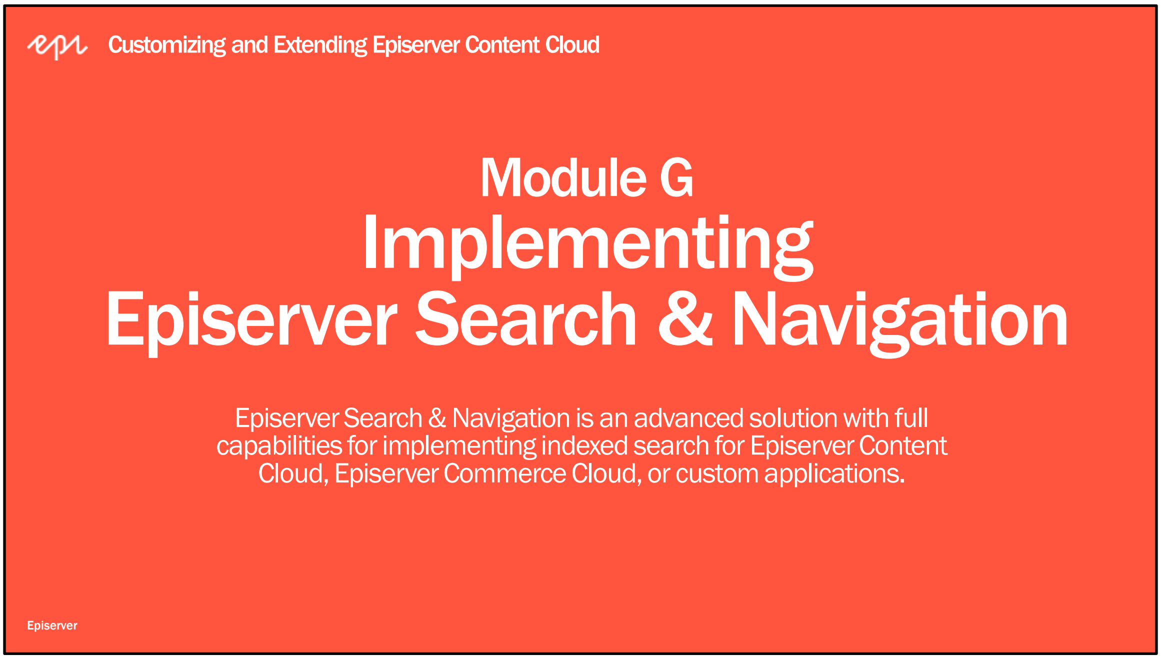
https://support.google.com/webmasters/answer/139066?hl=en

Exercises F1 to F6

Extending with Plug-ins and Add-ons

1. Exploring existing add-ons and plug-ins
2. Creating scheduled job plug-ins
3. Creating an admin tool plug-in
4. Creating a report plug-in
5. Customizing views
6. Integrating with Tasks in the Navigation pane

Episerver



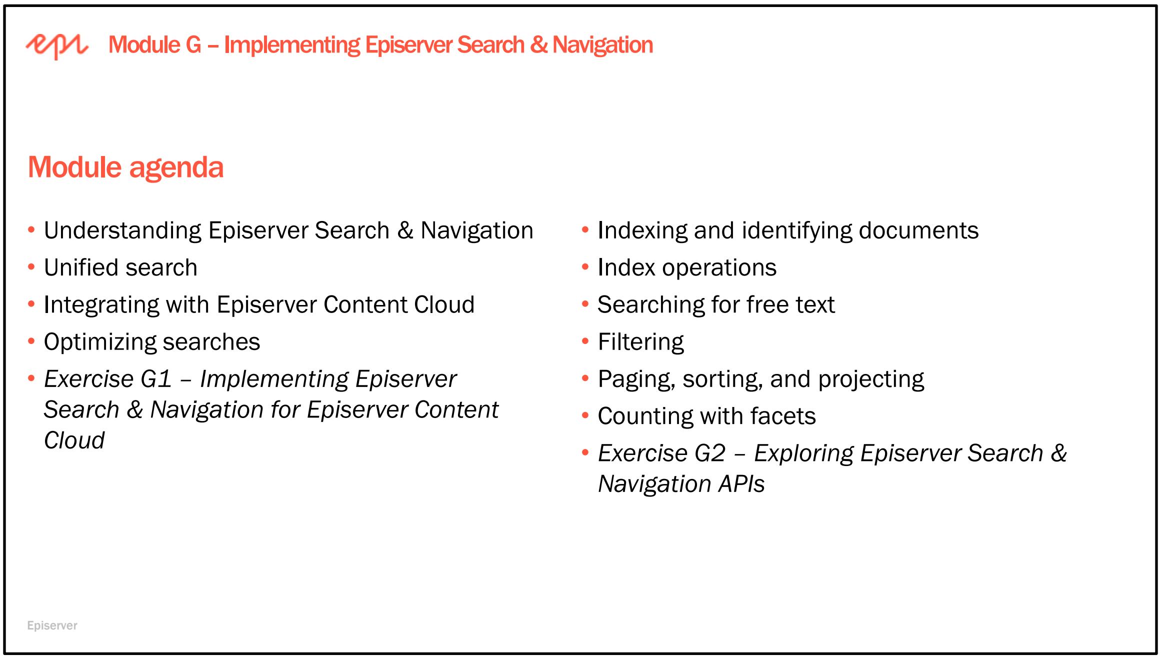
Customizing and Extending Episerver Content Cloud

Module G

Implementing Episerver Search & Navigation

Episerver Search & Navigation is an advanced solution with full capabilities for implementing indexed search for Episerver Content Cloud, Episerver Commerce Cloud, or custom applications.

Episerver



Module G – Implementing Episerver Search & Navigation

Module agenda

- Understanding Episerver Search & Navigation
- Unified search
- Integrating with Episerver Content Cloud
- Optimizing searches
- *Exercise G1 – Implementing Episerver Search & Navigation for Episerver Content Cloud*
- Indexing and identifying documents
- Index operations
- Searching for free text
- Filtering
- Paging, sorting, and projecting
- Counting with facets
- *Exercise G2 – Exploring Episerver Search & Navigation APIs*

Episerver

Understanding Episerver Search & Navigation

Understanding Episerver Search & Navigation (formerly Find)

Episerver Search & Navigation 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 Search & Navigation?

- **Integration with Episerver Content Cloud and Commerce Cloud**: it integrates closely with our other products so as soon as content is published it is immediately indexed and appears in results.
- **Admin view**: it has an easy-to-use interface to view statistics and optimize results.
- **Managed Services**: it is a cloud solution fully managed by Episerver experts to keep your indexed searches running smoothly 24/7/365.
- **Friendly .NET API**: it has an easy-to-use API that wraps the underlying complexity of the Elasticsearch REST indexing service.
- **Personalized Search**: it provides smart machine learning optimized search results.

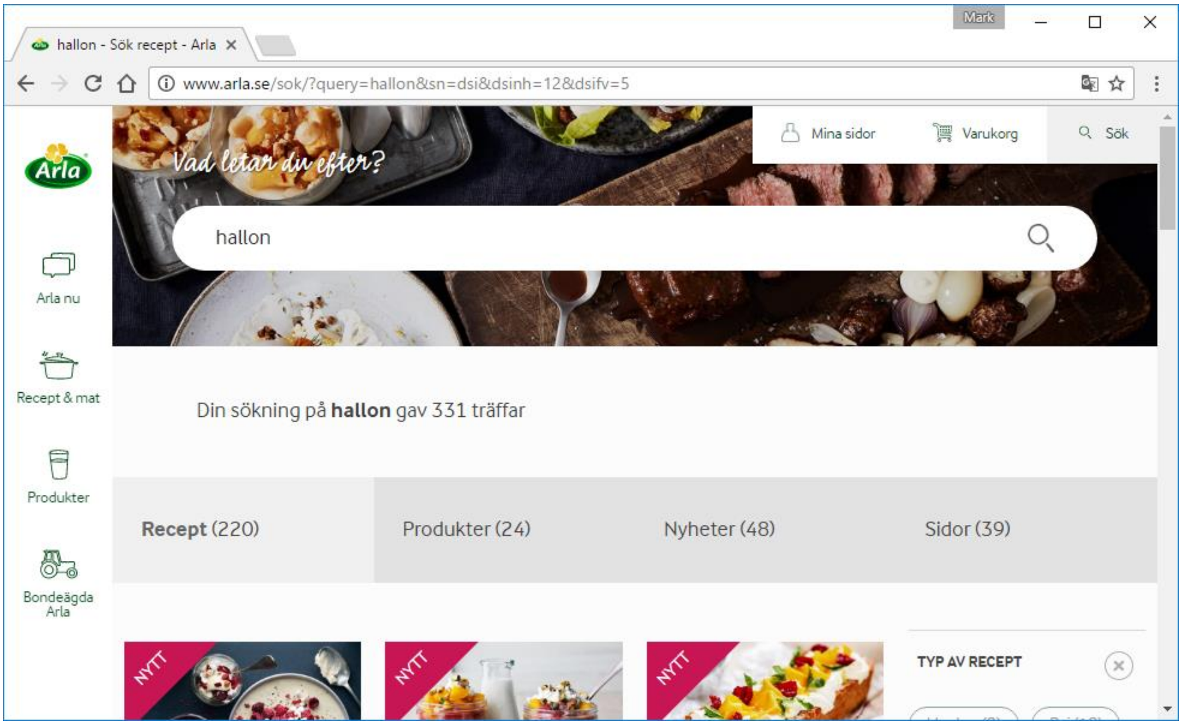
Episerver

Sites that use Episerver Search & Navigation

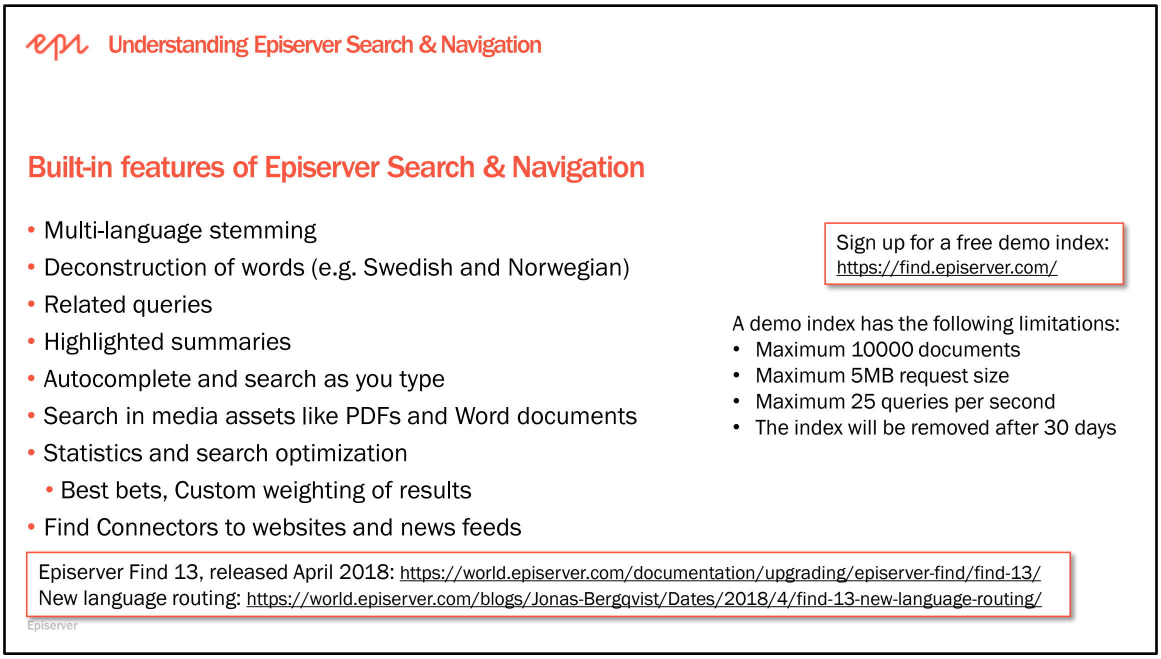
Arla
http://www.arla.se/

Small Luxury Hotels of the World
http://www.slh.com/

SMALL LUXURY HOTELS OF THE WORLD™
Independently minded



Din sökning på **hallon** gav 331 träffar



Understanding Episerver Search & Navigation

Built-in features of Episerver Search & Navigation

- Multi-language stemming
- Deconstruction of words (e.g. Swedish and Norwegian)
- Related queries
- Highlighted summaries
- Autocomplete and search as you type
- Search in media assets like PDFs and Word documents
- 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 30 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

Decompounding

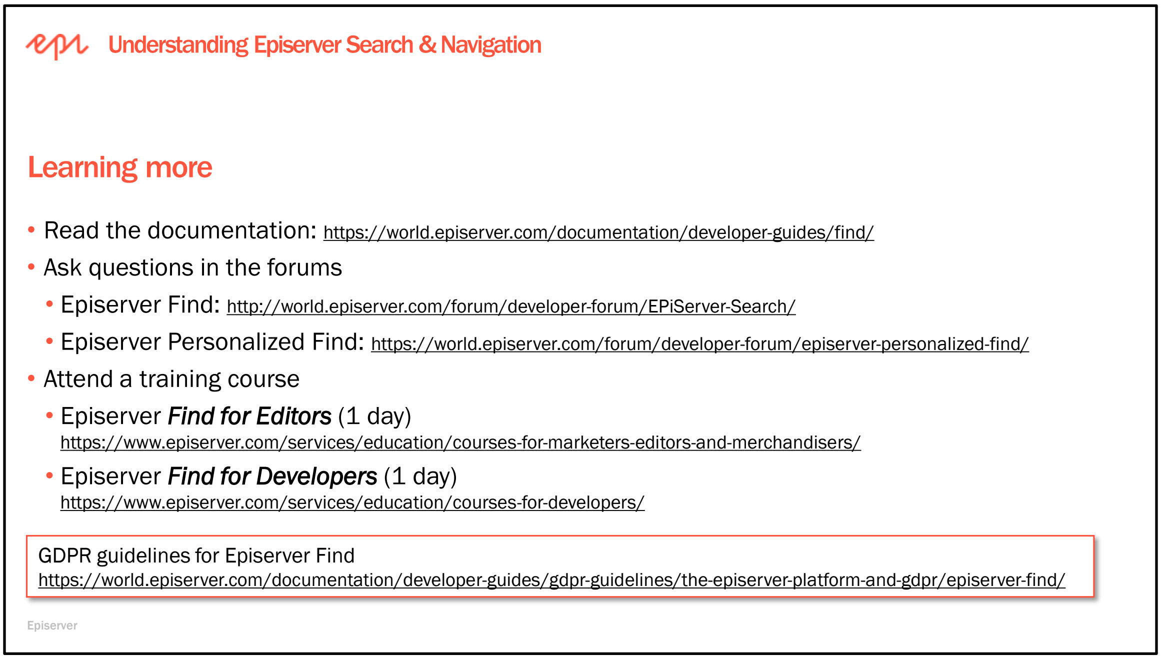
- cheeseburger → cheese burger
- football → foot ball
- blårutigskjortan → blå rutig skjorta n (the blue checkered shirt)
- banan → bana n (the trajectory)
- banan → banana

Unstable Episerver Find developer (demo) indexes

Intermittently the free Episerver Find demo indexes can stop working for a short period. This article shows how you can create Episerver websites that can start up even if its Find index is temporarily unavailable.
https://www.brianweet.com/2018/03/20/unstable-episerver-find-developer-indexes.html

Installing Episerver Find

- Installed through NuGet
- Requires additional license + create an index in cloud service
- Works with Episerver CMS 6 and higher
- Works with Episerver Commerce
- Requires the full .NET framework (not Client Profile)
- Depends on JSON.NET (Newtonsoft.Json.dll)



Understanding Episerver Search & Navigation

Learning more

- Read the documentation: https://world.episerver.com/documentation/developer-guides/find/
- Ask questions in the forums
  - Episerver Find: http://world.episerver.com/forum/developer-forum/EPiServer-Search/
  - Episerver Personalized Find: https://world.episerver.com/forum/developer-forum/episerver-personalized-find/
- Attend a training course
  - Episerver *Find for Editors* (1 day) https://www.episerver.com/services/education/courses-for-marketers-editors-and-merchandisers/
  - Episerver *Find for Developers* (1 day) https://www.episerver.com/services/education/courses-for-developers/

GDPR guidelines for Episerver Find
https://world.episerver.com/documentation/developer-guides/gdpr-guidelines/the-episerver-platform-and-gdpr/episerver-find/

Episerver

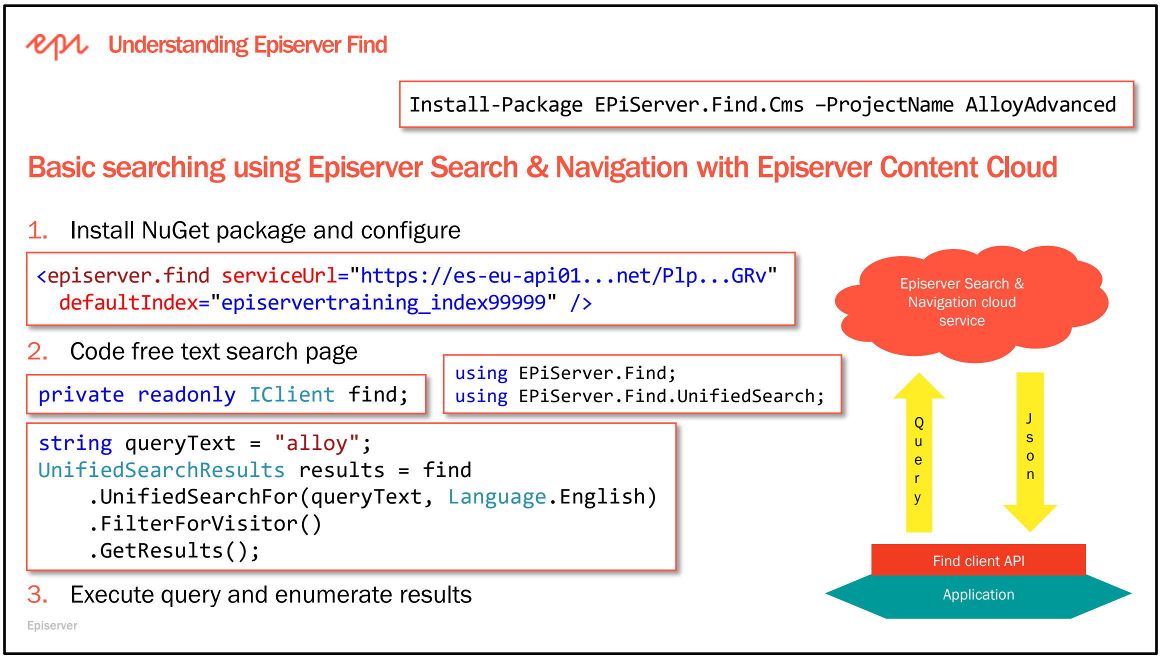
InspectInIndex
A quick and easy way to inspect Episerver content in an Episerver Find index.
https://github.com/BVNetwork/InspectInIndex/

```
Install-Package EPiCode.InspectInIndex
```

How to increase the Term Facet Count from default of 10
http://world.episerver.com/forum/developer-forum/EPiServer-Search/Thread-Container/2013/6/Term-facet-count/

Indexing content in a content area
http://world.episerver.com/documentation/developer-guides/find/Integration/episerver-cms-7-5-with-updates/Indexing-content-in-a-content-area/

Searching in blocks
http://world.episerver.com/Modules/Forum/Pages/Thread.aspx?id=65052



Understanding Episerver Find

```
Install-Package EPiServer.Find.Cms –ProjectName AlloyAdvanced
```

Basic searching using Episerver Search & Navigation with Episerver Content Cloud

1. Install NuGet package and configure

```
<episerver.find serviceUrl="https://es-eu-api01...net/Plp...GRv"
  defaultIndex="episervertraining_index99999" />
```

2. Code free text search page

```
private readonly IClient find;
```

```
using EPiServer.Find;
using EPiServer.Find.UnifiedSearch;
```

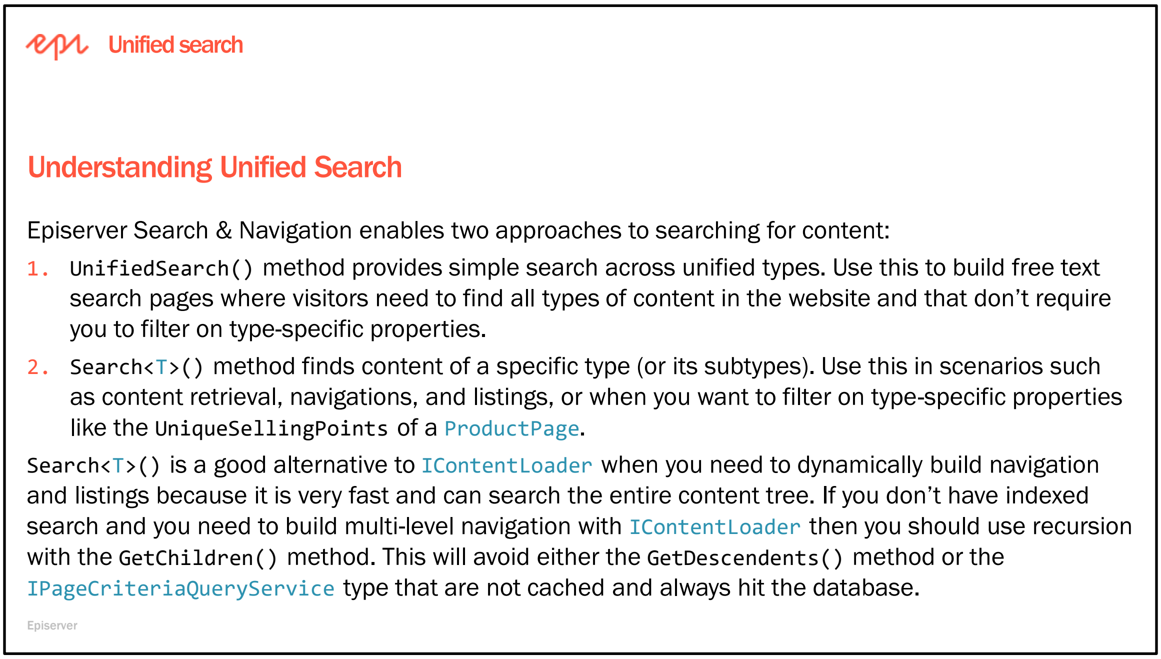
```
string queryText = "alloy";
UnifiedSearchResults results = find
    .UnifiedSearchFor(queryText, Language.English)
    .FilterForVisitor()
    .GetResults();
```

3. Execute query and enumerate results

Episerver

Good Practice

<title> and <meta name="description"> needs to be properly filled for Episerver Find to index an external page correctly. If the meta description is missing, Find will use the nearest <h2> (or <p> tag if <h2> is missing).



Unified search

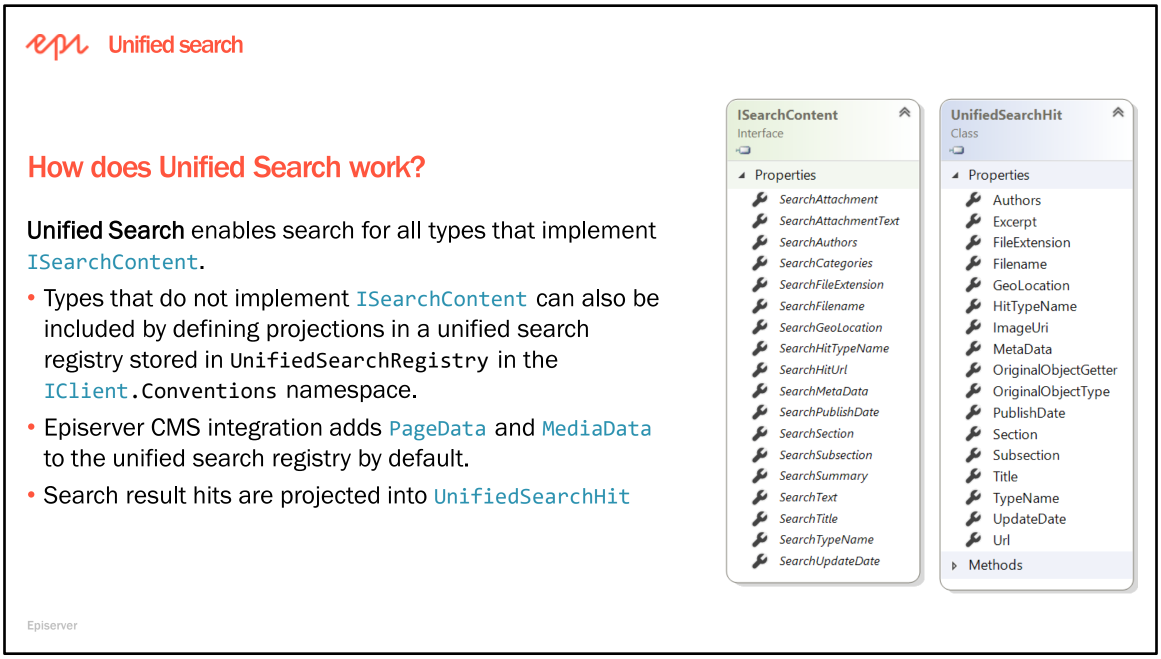
Understanding Unified Search

Episerver Search & Navigation enables two approaches to searching for content:

1. `UnifiedSearch()` method provides simple search across unified types. Use this to build free text search pages where visitors need to find all types of content in the website and that don't require you to filter on type-specific properties.
2. `Search<T>()` method finds content of a specific type (or its subtypes). Use this in scenarios such as content retrieval, navigations, and listings, or when you want to filter on type-specific properties like the `UniqueSellingPoints` of a `ProductPage`.

`Search<T>()` is a good alternative to `IContentLoader` when you need to dynamically build navigation and listings because it is very fast and can search the entire content tree. If you don't have indexed search and you need to build multi-level navigation with `IContentLoader` then you should use recursion with the `GetChildren()` method. This will avoid either the `GetDescendents()` method or the `IPageCriteriaQueryService` type that are not cached and always hit the database.

Episerver



Unified search

How does Unified Search work?

Unified Search enables search for all types that implement `ISearchContent`.

- Types that do not implement `ISearchContent` can also be included by defining projections in a unified search registry stored in `UnifiedSearchRegistry` in the `IClient.Conventions` namespace.
- Episerver CMS integration adds `PageData` and `MediaData` to the unified search registry by default.
- Search result hits are projected into `UnifiedSearchHit`

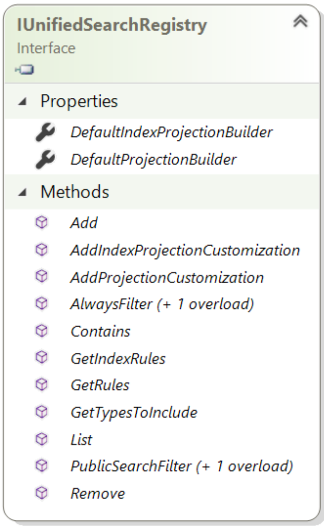
Episerver

`ISearchContent` interface defines quite a lot of properties allowing it to cover most scenarios when building a search page. Most notable are `SearchTitle`, `SearchText` and `SearchHitUrl` as they are typically the most frequently used when building a search page.

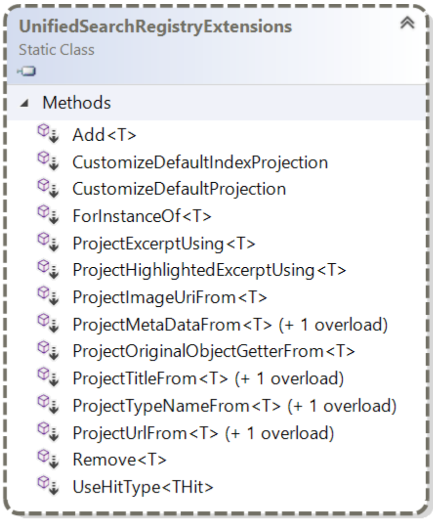
For both PageData and MediaData-derived content types, SearchSection is set to the Name of the ancestor below the start page and SearchSubSection to the ancestor below the SearchSection page.

The unified search registry can be used to map any property to one of the `ISearchContent` properties for inclusion in unified search results.

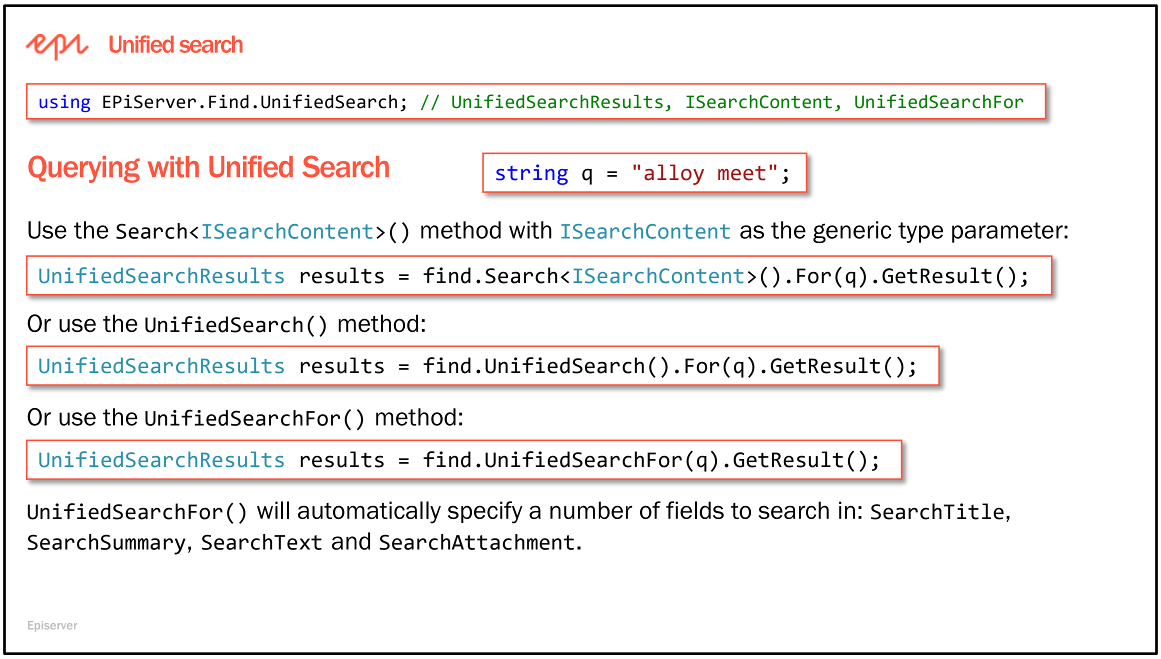
```
find.Conventions.UnifiedSearchRegistry
    .ForInstanceOf<StandardPage>()
    .ProjectTitleFrom(spec => spec.MetaTitle);
```



IUnifiedSearchRegistry
Interface
Properties
DefaultIndexProjectionBuilder
DefaultProjectionBuilder
Methods
Add
AddIndexProjectionCustomization
AddProjectionCustomization
AlwaysFilter (+ 1 overload)
Contains
GetIndexRules
GetRules
GetTypesToInclude
List
PublicSearchFilter (+ 1 overload)
Remove



UnifiedSearchRegistryExtensions
Static Class
Methods
Add<T>
CustomizeDefaultIndexProjection
CustomizeDefaultProjection
ForInstanceOf<T>
ProjectExcerptUsing<T>
ProjectHighlightedExcerptUsing<T>
ProjectImageUriFrom<T>
ProjectMetaDataFrom<T> (+ 1 overload)
ProjectOriginalObjectGetterFrom<T>
ProjectTitleFrom<T> (+ 1 overload)
ProjectTypeNameFrom<T> (+ 1 overload)
ProjectUrlFrom<T> (+ 1 overload)
Remove<T>
UseHitType<THit>



Unified search

```
using EPiServer.Find.UnifiedSearch; // UnifiedSearchResults, ISearchContent, UnifiedSearchFor
```

Querying with Unified Search

```
string q = "alloy meet";
```

Use the `Search<ISearchContent>()` method with `ISearchContent` as the generic type parameter:

```
UnifiedSearchResults results = find.Search<ISearchContent>().For(q).GetResult();
```

Or use the `UnifiedSearch()` method:

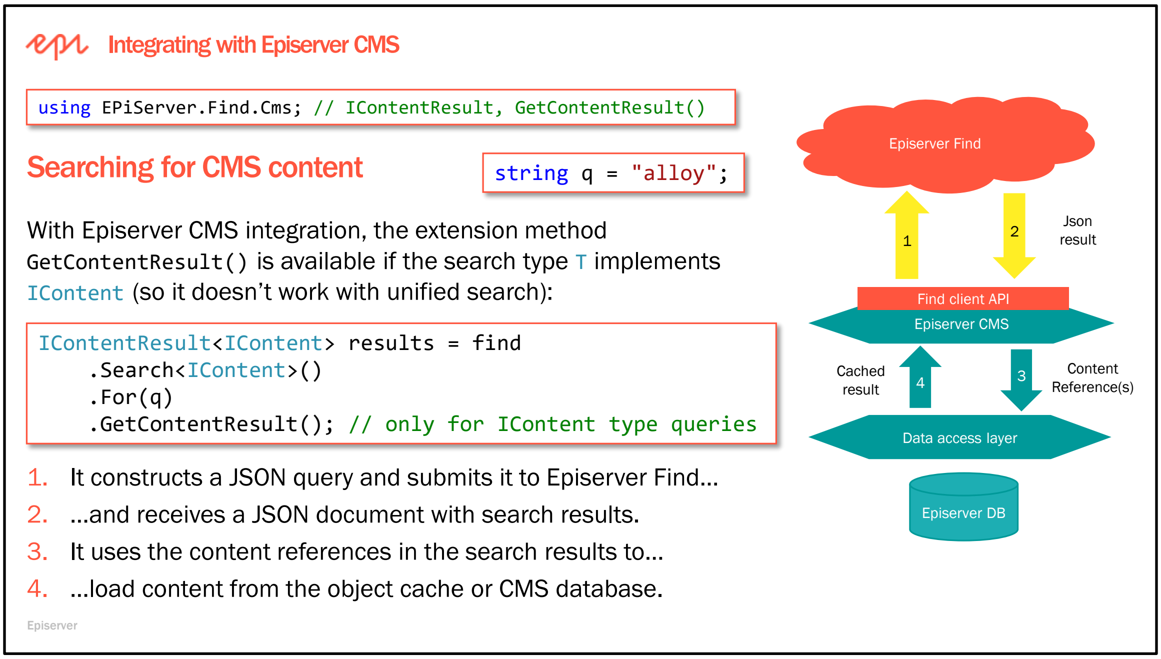
```
UnifiedSearchResults results = find.UnifiedSearch().For(q).GetResult();
```

Or use the `UnifiedSearchFor()` method:

```
UnifiedSearchResults results = find.UnifiedSearchFor(q).GetResult();
```

`UnifiedSearchFor()` will automatically specify a number of fields to search in: `SearchTitle`, `SearchSummary`, `SearchText` and `SearchAttachment`.

Episerver



Integrating with Episerver CMS

```
using EPiServer.Find.Cms; // IContentResult, GetContentResult()
```

Searching for CMS content

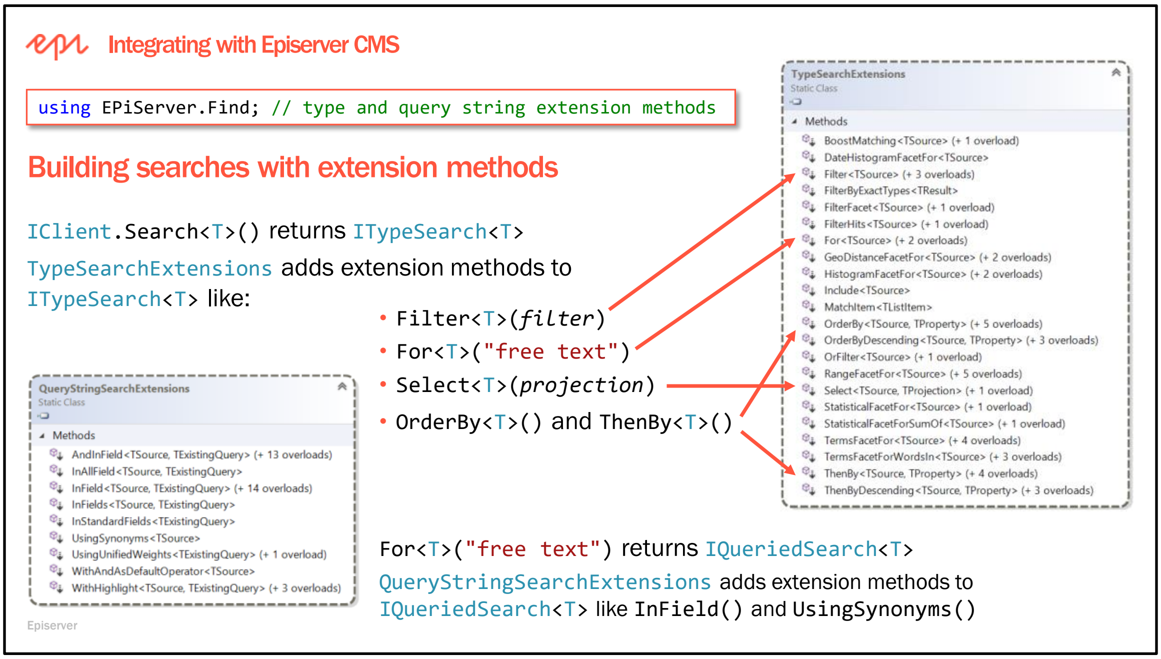
```
string q = "alloy";
```

With Episerver CMS integration, the extension method `GetContentResult()` is available if the search type `T` implements `IContent` (so it doesn't work with unified search):

```
IContentResult<IContent> results = find
    .Search<IContent>()
    .For(q)
    .GetContentResult(); // only for IContent type queries
```

1. It constructs a JSON query and submits it to Episerver Find…
2. …and receives a JSON document with search results.
3. It uses the content references in the search results to…
4. …load content from the object cache or CMS database.

Episerver



Integrating with Episerver CMS

```
using EPiServer.Find; // type and query string extension methods
```

Building searches with extension methods

`IClient.Search<T>()` returns `ITypeSearch<T>`

`TypeSearchExtensions` adds extension methods to `ITypeSearch<T>` like:

- `Filter<T>(filter)`
- `For<T>("free text")`
- `Select<T>(projection)`
- `OrderBy<T>()` and `ThenBy<T>()`

TypeSearchExtensions
Static Class

Methods
BoostMatching<TSource> (+ 1 overload)
DateHistogramFacetFor<TSource>
Filter<TSource> (+ 3 overloads)
FilterByExactTypes<TResult>
FilterFacet<TSource> (+ 1 overload)
FilterHits<TSource> (+ 1 overload)
For<TSource> (+ 2 overloads)
GeoDistanceFacetFor<TSource> (+ 2 overloads)
HistogramFacetFor<TSource> (+ 2 overloads)
Include<TSource>
MatchItem<TListItem>
OrderBy<TSource, TProperty> (+ 5 overloads)
OrderByDescending<TSource, TProperty> (+ 3 overloads)
OrFilter<TSource> (+ 1 overload)
RangeFacetFor<TSource> (+ 5 overloads)
Select<TSource, TProjection> (+ 1 overload)
StatisticalFacetFor<TSource> (+ 1 overload)
StatisticalFacetForSumOf<TSource> (+ 1 overload)
TermsFacetFor<TSource> (+ 4 overloads)
TermsFacetForWordsIn<TSource> (+ 3 overloads)
ThenBy<TSource, TProperty> (+ 4 overloads)
ThenByDescending<TSource, TProperty> (+ 3 overloads)

QueryStringSearchExtensions
Static Class

Methods
AndInField<TSource, TExistingQuery> (+ 13 overloads)
InAllField<TSource, TExistingQuery>
InField<TSource, TExistingQuery> (+ 14 overloads)
InFields<TSource, TExistingQuery>
InStandardFields<TExistingQuery>
UsingSynonyms<TSource>
UsingUnifiedWeights<TExistingQuery> (+ 1 overload)
WithAndAsDefaultOperator<TSource>
WithHighlight<TSource, TExistingQuery> (+ 3 overloads)

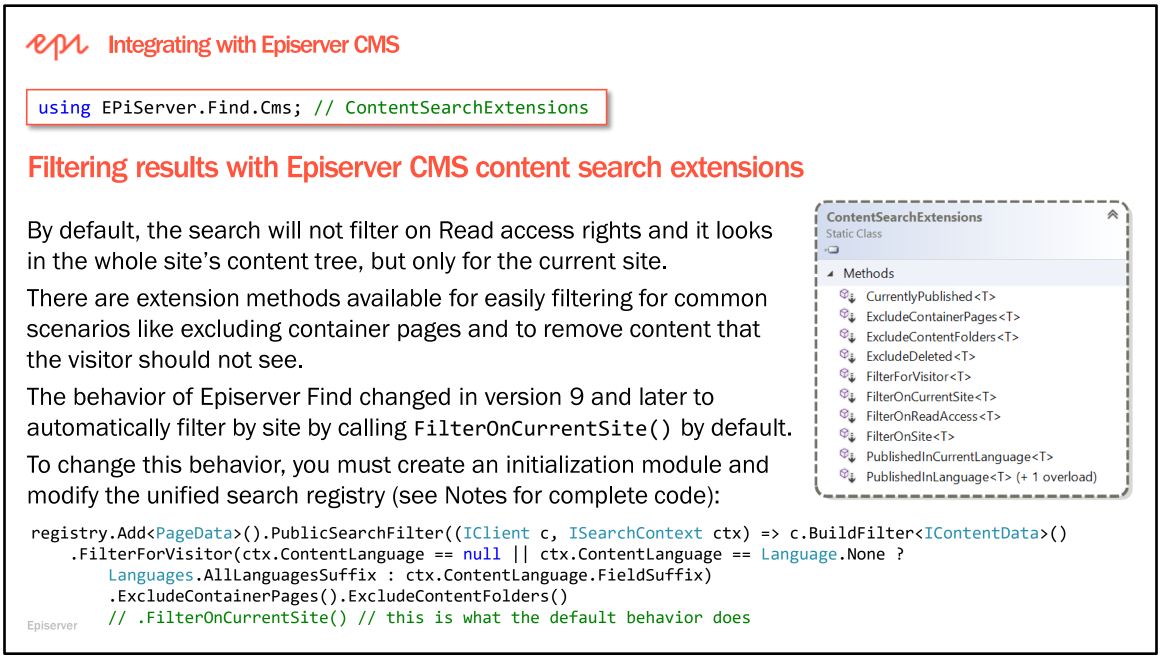
Episerver

`For<T>("free text")` returns `IQueriedSearch<T>`

`QueryStringSearchExtensions` adds extension methods to `IQueriedSearch<T>` like `InField()` and `UsingSynonyms()`

Be careful to check what type is returned from an extension method, for example, `ITypeSearch<T>` or `IQueriedSearch<T>`. Some do not have some extension methods so you must call the extension methods in the correct order. For example:

```
IContentResult<StandardPage> results = find
     .Search<StandardPage>() // returns ITypeSearch<StandardPage>
     .For("alloy")           // returns IQueriedSearch<T>
     .UsingSynonyms()        // only available on IQueriedSearch<T>
     .Filter(                // available on ITypeSearch<T>
         page => page.RolesWithReadAccess().Match("Everyone"))
     .Track()                // available on ITypeSearch<T>
     .ApplyBestBets()        // available on ITypeSearch<T>
     .GetContentResult();    // only available on ITypeSearch<IContent>
```



Integrating with Episerver CMS

```
using EPiServer.Find.Cms; // ContentSearchExtensions
```

Filtering results with Episerver CMS content search extensions

By default, the search will not filter on Read access rights and it looks in the whole site's content tree, but only for the current site.

There are extension methods available for easily filtering for common scenarios like excluding container pages and to remove content that the visitor should not see.

The behavior of Episerver Find changed in version 9 and later to automatically filter by site by calling `FilterOnCurrentSite()` by default.

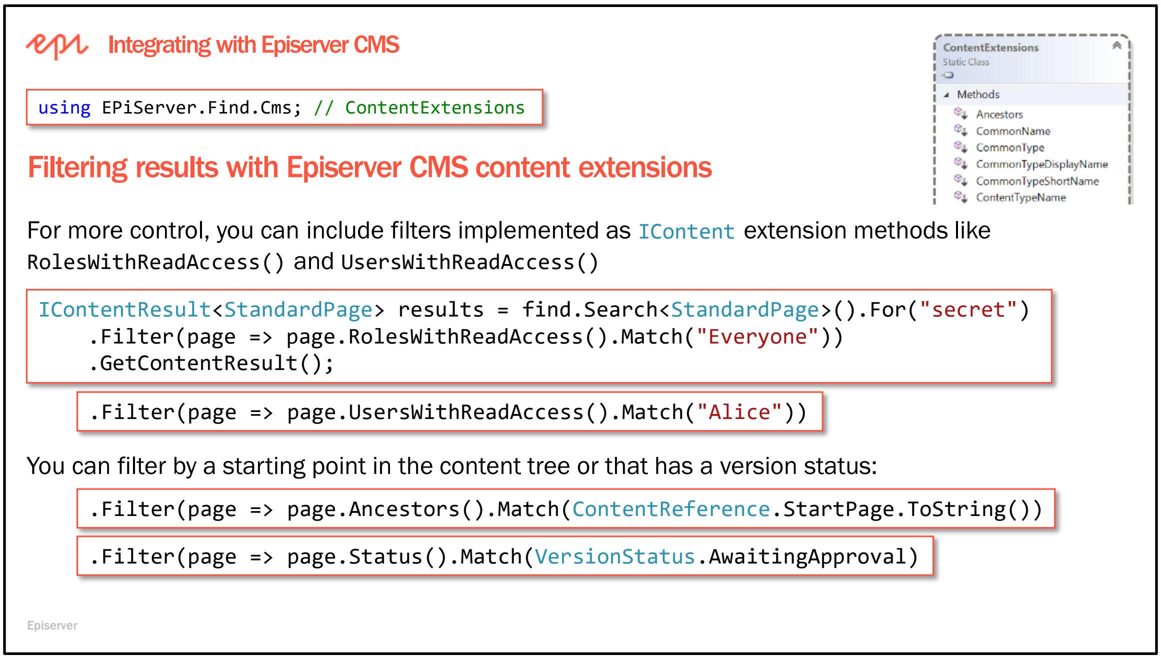
To change this behavior, you must create an initialization module and modify the unified search registry (see Notes for complete code):

ContentSearchExtensions
Static Class

Methods
CurrentlyPublished<T>
ExcludeContainerPages<T>
ExcludeContentFolders<T>
ExcludeDeleted<T>
FilterForVisitor<T>
FilterOnCurrentSite<T>
FilterOnReadAccess<T>
FilterOnSite<T>
PublishedInCurrentLanguage<T>
PublishedInLanguage<T> (+ 1 overload)

```
registry.Add<PageData>().PublicSearchFilter((IClient c, ISearchContext ctx) => c.BuildFilter<IContentData>()
    .FilterForVisitor(ctx.ContentLanguage == null || ctx.ContentLanguage == Language.None ?
        Languages.AllLanguagesSuffix : ctx.ContentLanguage.FieldSuffix)
        .ExcludeContainerPages().ExcludeContentFolders()
        // .FilterOnCurrentSite() // this is what the default behavior does
```

```
using EPiServer.Core;
using EPiServer.Find;
using EPiServer.Find.Cms;
using EPiServer.Find.Framework;
using EPiServer.Find.UnifiedSearch;
using EPiServer.Framework;
using EPiServer.Framework.Initialization;

namespace AlloyAdvanced.Business.Initialization
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Find.Cms.Module.IndexingModule))]
    public class MultisiteFindInitializationModule : IInitializableModule
    {
        public void Initialize(InitializationEngine context)
        {
            var setup = new CmsUnifiedSearchSetUp();
            IUnifiedSearchRegistry registry = SearchClient.Instance.Conventions.UnifiedSearchRegistry;

            registry.Add<PageData>()
                .PublicSearchFilter((IClient c, ISearchContext ctx) => c.BuildFilter<IContentData>()
                    .FilterForVisitor(
                        ctx.ContentLanguage == null || ctx.ContentLanguage == Language.None ?
                        Languages.AllLanguagesSuffix : ctx.ContentLanguage.FieldSuffix)
                    .ExcludeContainerPages()
                    .ExcludeContentFolders()
                    // .FilterOnCurrentSite() // this is what the default behavior does
                    )
                .CustomizeIndexProjection(setup.CustomizeIndexProjectionForPageData)
                .CustomizeProjection(setup.CustomizeProjectionForPageData);

            registry.Add<MediaData>()
                .PublicSearchFilter((c, ctx) => c.BuildFilter<IContentData>()
                    .FilterForVisitor(
                        ctx.ContentLanguage == null || ctx.ContentLanguage == Language.None ?
                        Languages.AllLanguagesSuffix : ctx.ContentLanguage.FieldSuffix)
                    .ExcludeContentFolders())
                .CustomizeIndexProjection(setup.CustomizeIndexProjectionForMediaData)
                .CustomizeProjection(setup.CustomizeProjectionForMediaData);
        }

        public void Uninitialize(InitializationEngine context) { }
    }
}
```



Integrating with Episerver CMS

```
using EPiServer.Find.Cms; // ContentExtensions
```

ContentExtensions
Static Class

Methods
- Ancestors
- CommonName
- CommonType
- CommonTypeDisplayName
- CommonTypeShortName
- ContentTypeName

Filtering results with Episerver CMS content extensions

For more control, you can include filters implemented as `IContent` extension methods like `RolesWithReadAccess()` and `UsersWithReadAccess()`

```
IContentResult<StandardPage> results = find.Search<StandardPage>().For("secret")
    .Filter(page => page.RolesWithReadAccess().Match("Everyone"))
    .GetContentResult();
```

```
.Filter(page => page.UsersWithReadAccess().Match("Alice"))
```

You can filter by a starting point in the content tree or that has a version status:

```
.Filter(page => page.Ancestors().Match(ContentReference.StartPage.ToString())
```

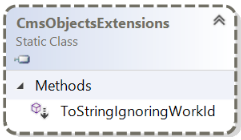
```
.Filter(page => page.Status().Match(VersionStatus.AwaitingApproval)
```

Episerver

After importing the `EPiServer.Find.Cms` namespace, an extension method named `Ancestors()` is included when indexing `IContent` (pages, shared blocks, and so on).

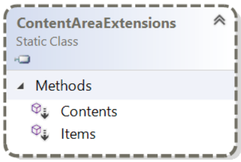
The `Ancestors()` method returns a list containing the string representation of the `ContentLink` property of each of the indexed contents ancestors in the content tree. This can be used to filter for content located below a certain node in the content tree.

Episerver Find will automatically index all sites in a multi-site setup and you can filter the results per-site so that you will by default only get results for the site you are currently browsing.



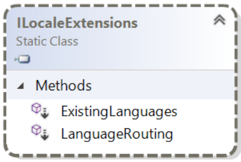
CmsObjectsExtensions
Static Class

Methods
- ToStringIgnoringWorkId



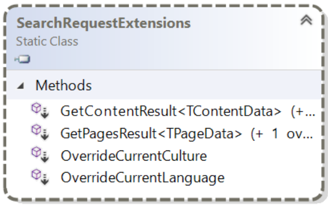
ContentAreaExtensions
Static Class

Methods
- Contents
- Items



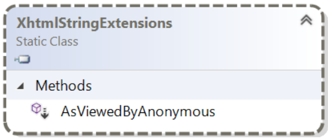
ILocaleExtensions
Static Class

Methods
- ExistingLanguages
- LanguageRouting



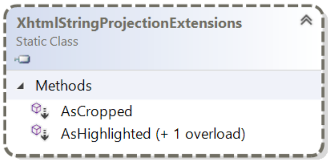
SearchRequestExtensions
Static Class

Methods
- GetContentResult<TContentData> (+...
- GetPagesResult<TPageData> (+ 1 ov...
- OverrideCurrentCulture
- OverrideCurrentLanguage



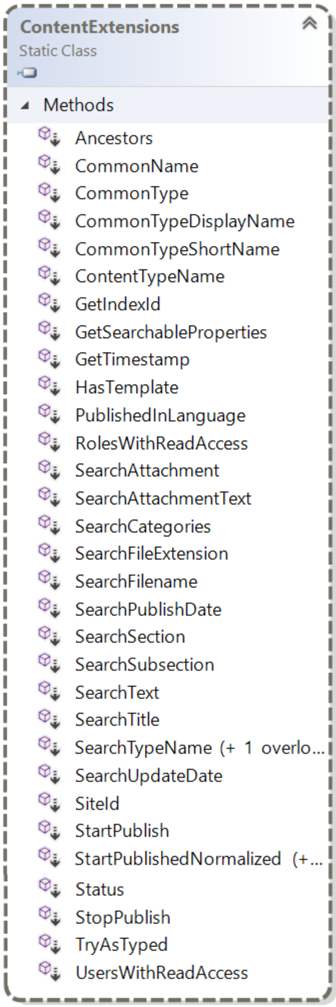
XhtmlStringExtensions
Static Class

Methods
- AsViewedByAnonymous



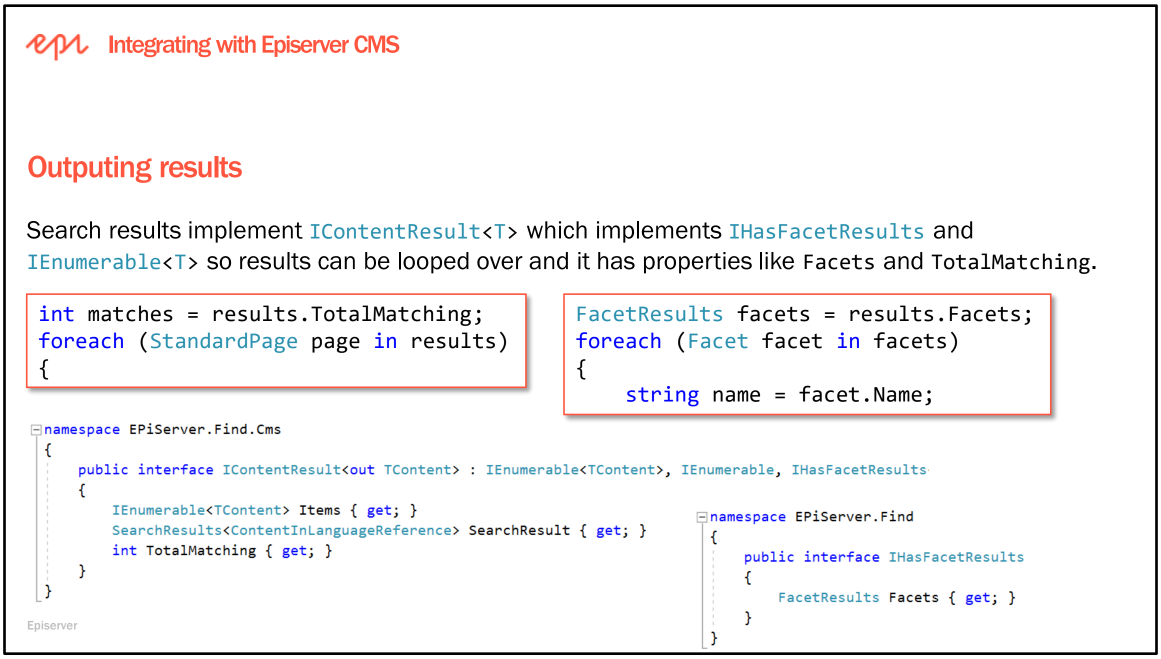
XhtmlStringProjectionExtensions
Static Class

Methods
- AsCropped
- AsHighlighted (+ 1 overload)



ContentExtensions
Static Class

Methods
- Ancestors
- CommonName
- CommonType
- CommonTypeDisplayName
- CommonTypeShortName
- ContentTypeName
- GetIndexId
- GetSearchableProperties
- GetTimestamp
- HasTemplate
- PublishedInLanguage
- RolesWithReadAccess
- SearchAttachment
- SearchAttachmentText
- SearchCategories
- SearchFileExtension
- SearchFilename
- SearchPublishDate
- SearchSection
- SearchSubsection
- SearchText
- SearchTitle
- SearchTypeName (+ 1 overlo...
- SearchUpdateDate
- SiteId
- StartPublish
- StartPublishedNormalized (+...
- Status
- StopPublish
- TryAsTyped
- UsersWithReadAccess



Integrating with Episerver CMS

Outputing results

Search results implement `IContentResult<T>` which implements `IHasFacetResults` and `IEnumerable<T>` so results can be looped over and it has properties like `Facets` and `TotalMatching`.

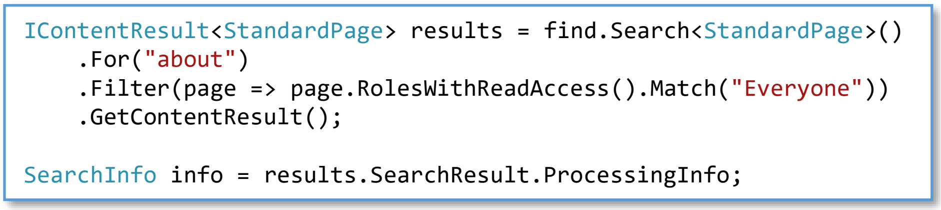
```
int matches = results.TotalMatching;
foreach (StandardPage page in results)
{
```

```
FacetResults facets = results.Facets;
foreach (Facet facet in facets)
{
    string name = facet.Name;
```

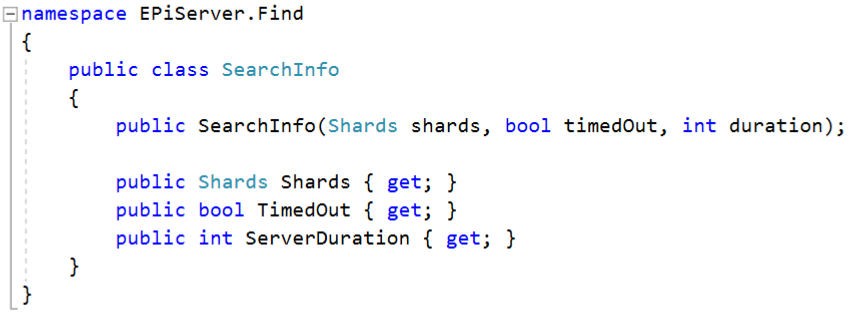
```
namespace EPiServer.Find.Cms
{
    public interface IContentResult<out TContent> : IEnumerable<TContent>, IEnumerable, IHasFacetResults
    {
        IEnumerable<TContent> Items { get; }
        SearchResults<ContentInLanguageReference> SearchResult { get; }
        int TotalMatching { get; }
    }
}
```

```
namespace EPiServer.Find
{
    public interface IHasFacetResults
    {
        FacetResults Facets { get; }
    }
}
```

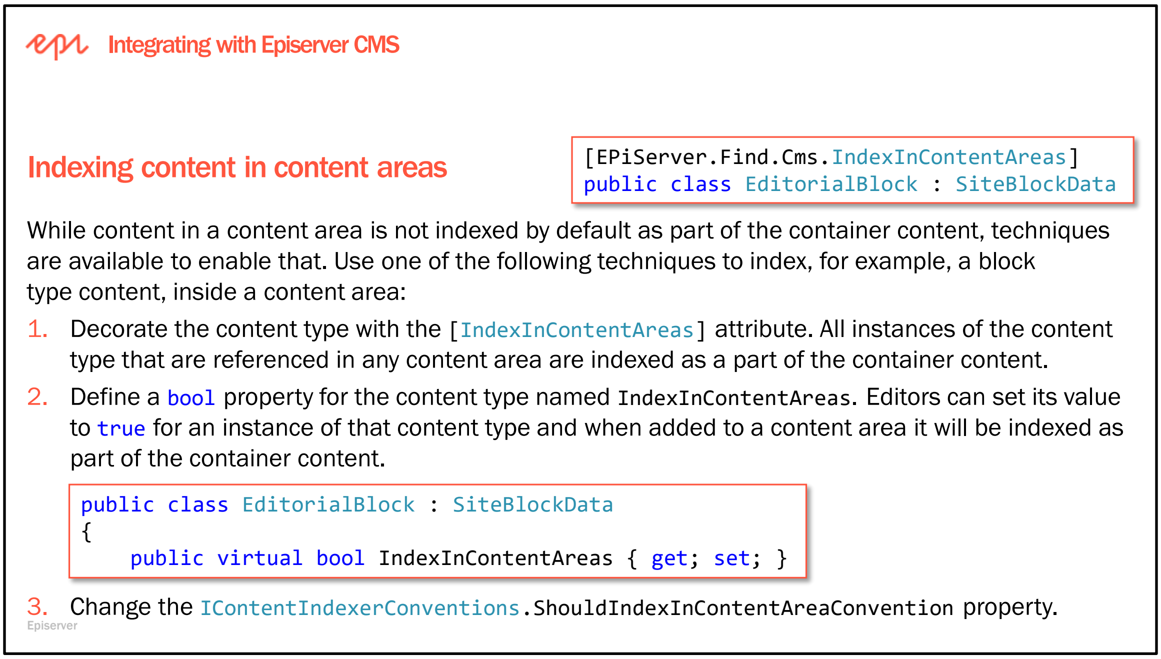
Episerver



```
IContentResult<StandardPage> results = find.Search<StandardPage>()
    .For("about")
    .Filter(page => page.RolesWithReadAccess().Match("Everyone"))
    .GetContentResult();

SearchInfo info = results.SearchResult.ProcessingInfo;
```



```
namespace EPiServer.Find
{
    public class SearchInfo
    {
        public SearchInfo(Shards shards, bool timedOut, int duration);

        public Shards Shards { get; }
        public bool TimedOut { get; }
        public int ServerDuration { get; }
    }
}
```



Integrating with Episerver CMS

Indexing content in content areas

```
[EPiServer.Find.Cms.IndexInContentAreas]
public class EditorialBlock : SiteBlockData
```

While content in a content area is not indexed by default as part of the container content, techniques are available to enable that. Use one of the following techniques to index, for example, a block type content, inside a content area:

1. Decorate the content type with the `[IndexInContentAreas]` attribute. All instances of the content type that are referenced in any content area are indexed as a part of the container content.
2. Define a `bool` property for the content type named `IndexInContentAreas`. Editors can set its value to `true` for an instance of that content type and when added to a content area it will be indexed as part of the container content.

   ```
   public class EditorialBlock : SiteBlockData
   {
       public virtual bool IndexInContentAreas { get; set; }
   ```

3. Change the `IContentIndexerConventions.ShouldIndexInContentAreaConvention` property.

Episerver

Indexing Block's Content to make it searchable

By default, the content of a block (that is added to ContentArea on a page) is not indexed and therefore you can’t search for the content of that block instance in your site.

To index a particular block type, create a class and inherit it with interface IShouldIndexInContentAreaConvention:

```
public class ShouldIndexInContentAreaConvention : IShouldIndexInContentAreaConvention
{
    public bool? ShouldIndexInContentArea(IContent content)
    {
        return content is CopyBlock;
    }
}
```

https://world.episerver.com/blogs/pjangid/dates/2019/4/indexing-blocks-content-to-make-it-searchable/

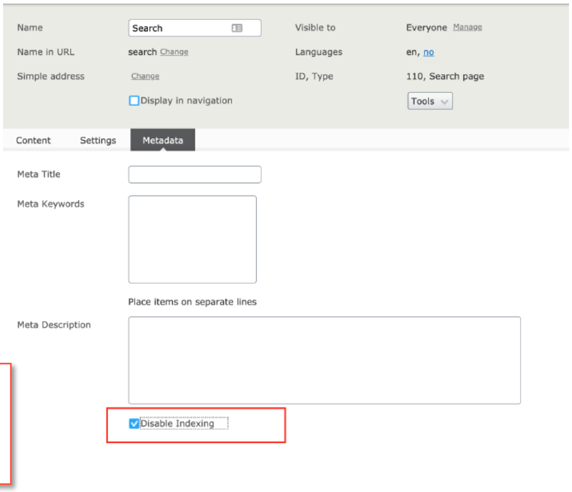
Integrating with Episerver CMS

Disabling indexing of content

The Alloy (MVC) project template includes a property to disable a page from being indexed but it is not implemented!

For external search engines, you could add the following code to the shared layout <head>:

```
@if (Model.CurrentPage.DisableIndexing)
{
  <meta name="robots" content="noindex" />
}
```

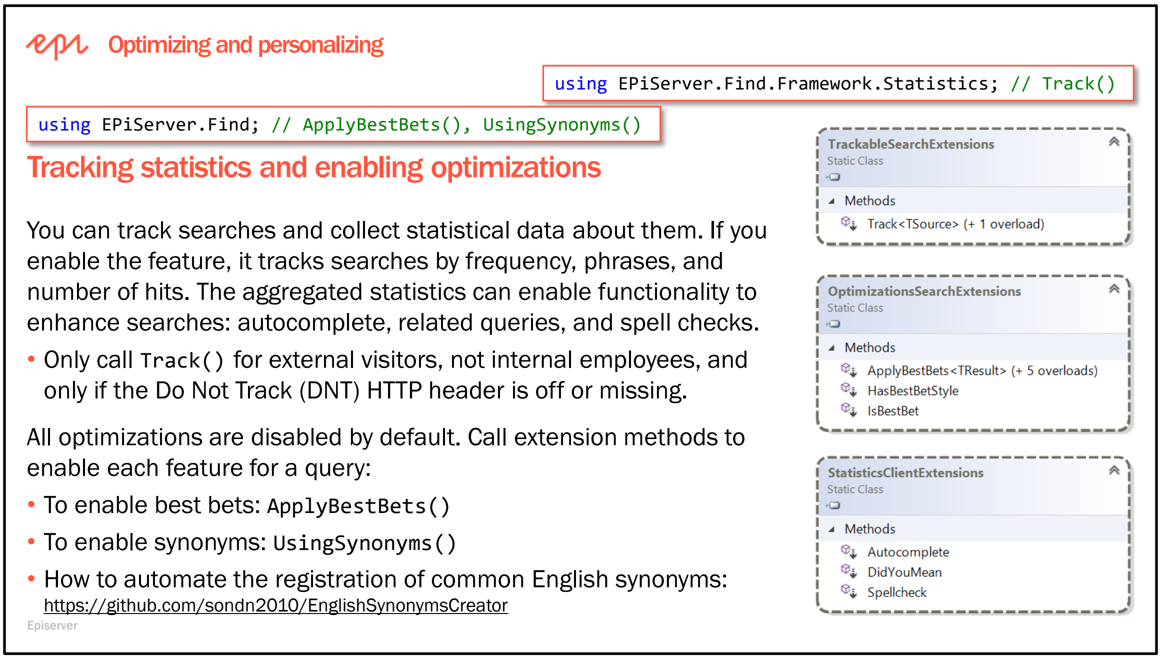


Name: Search
Visible to: Everyone Manage
Name in URL: search Change
Languages: en, no
Simple address: Change
ID, Type: 110, Search page
□ Display in navigation

Meta Title:
Meta Keywords:
Place items on separate lines
Meta Description:
☑ Disable Indexing

To disable indexing in Episerver Find, configure Find conventions in an initialization module:

```
ContentIndexer.Instance.Conventions.ForInstancesOf<SearchPage>()
  .ShouldIndex(x => false);
```



Optimizing and personalizing

```
using EPiServer.Find.Framework.Statistics; // Track()
```

```
using EPiServer.Find; // ApplyBestBets(), UsingSynonyms()
```

Tracking statistics and enabling optimizations

You can track searches and collect statistical data about them. If you enable the feature, it tracks searches by frequency, phrases, and number of hits. The aggregated statistics can enable functionality to enhance searches: autocomplete, related queries, and spell checks.

- Only call `Track()` for external visitors, not internal employees, and only if the Do Not Track (DNT) HTTP header is off or missing.

All optimizations are disabled by default. Call extension methods to enable each feature for a query:

- To enable best bets: `ApplyBestBets()`
- To enable synonyms: `UsingSynonyms()`
- How to automate the registration of common English synonyms: https://github.com/sondn2010/EnglishSynonymsCreator

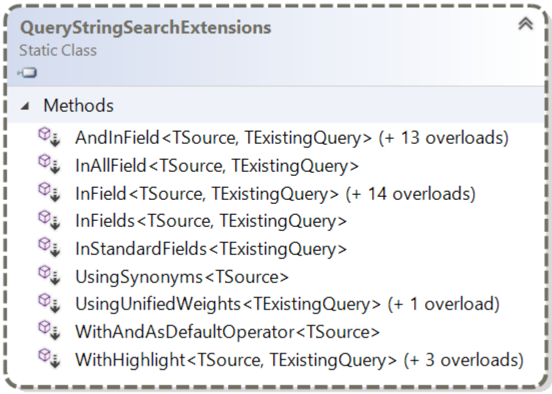
TrackableSearchExtensions
Static Class
Methods
Track<TSource> (+ 1 overload)

OptimizationsSearchExtensions
Static Class
Methods
ApplyBestBets<TResult> (+ 5 overloads)
HasBestBetStyle
IsBestBet

StatisticsClientExtensions
Static Class
Methods
Autocomplete
DidYouMean
Spellcheck

Episerver

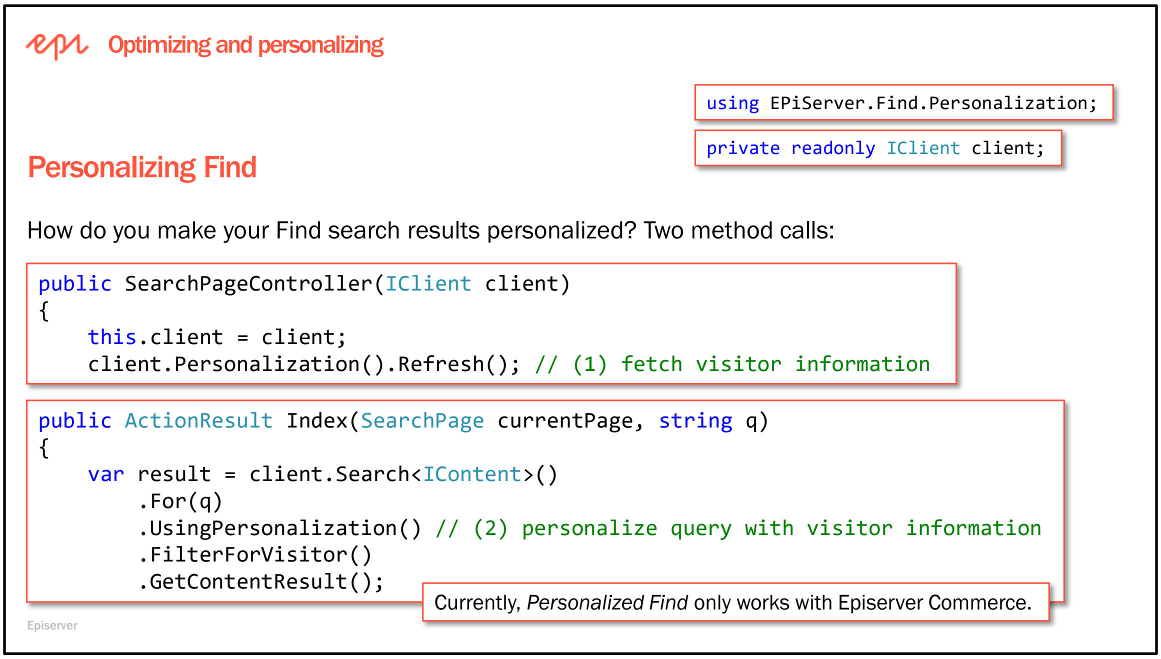
After importing `EPiServer.Find` the extension methods of `QueryStringSearchExtensions` are available to `IQueriedSearch<T>` queries, as returned by the `For()` free-text extension method:



QueryStringSearchExtensions
Static Class
Methods
AndInField<TSource, TExistingQuery> (+ 13 overloads)
InAllField<TSource, TExistingQuery>
InField<TSource, TExistingQuery> (+ 14 overloads)
InFields<TSource, TExistingQuery>
InStandardFields<TExistingQuery>
UsingSynonyms<TSource>
UsingUnifiedWeights<TExistingQuery> (+ 1 overload)
WithAndAsDefaultOperator<TSource>
WithHighlight<TSource, TExistingQuery> (+ 3 overloads)

To get the statistics client in order to get related queries and so on, import the `EPiServer.Find.Framework.Statistics` and `EPiServer.Find.Statistics` namespaces:

```
IStatisticsClient stats = find.Statistics(); // EPiServer.Find.Statistics

DidYouMeanResult relatedQueries = stats
    .DidYouMean(query: "alloy", size: 1); // EPiServer.Find.Framework.Statistics

AutocompleteResult suggestion = stats
    .Autocomplete(prefix: "alloy", size: 1);

SpellcheckResult spellings = stats
    .Spellcheck(query: "alloy", size: 3);
```



Optimizing and personalizing

```
using EPiServer.Find.Personalization;
```

```
private readonly IClient client;
```

Personalizing Find

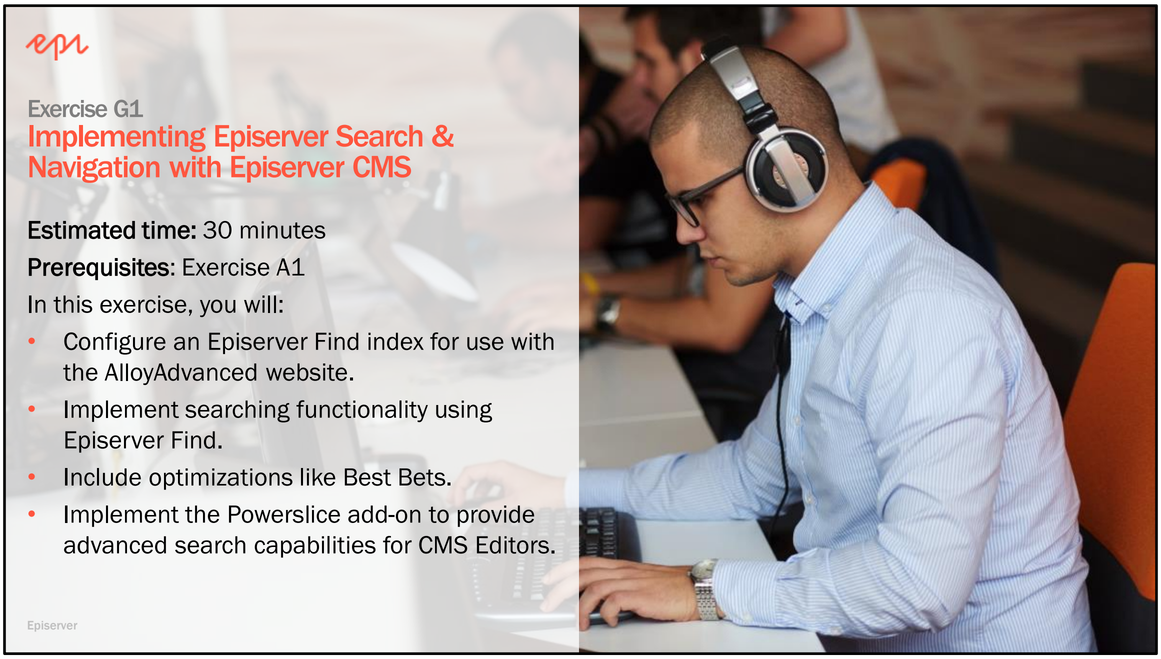
How do you make your Find search results personalized? Two method calls:

```
public SearchPageController(IClient client)
{
    this.client = client;
    client.Personalization().Refresh(); // (1) fetch visitor information
```

```
public ActionResult Index(SearchPage currentPage, string q)
{
    var result = client.Search<IContent>()
        .For(q)
        .UsingPersonalization() // (2) personalize query with visitor information
        .FilterForVisitor()
        .GetContentResult();
```

Currently, *Personalized Find* only works with Episerver Commerce.

Episerver



Exercise G1

Implementing Episerver Search & Navigation with Episerver CMS

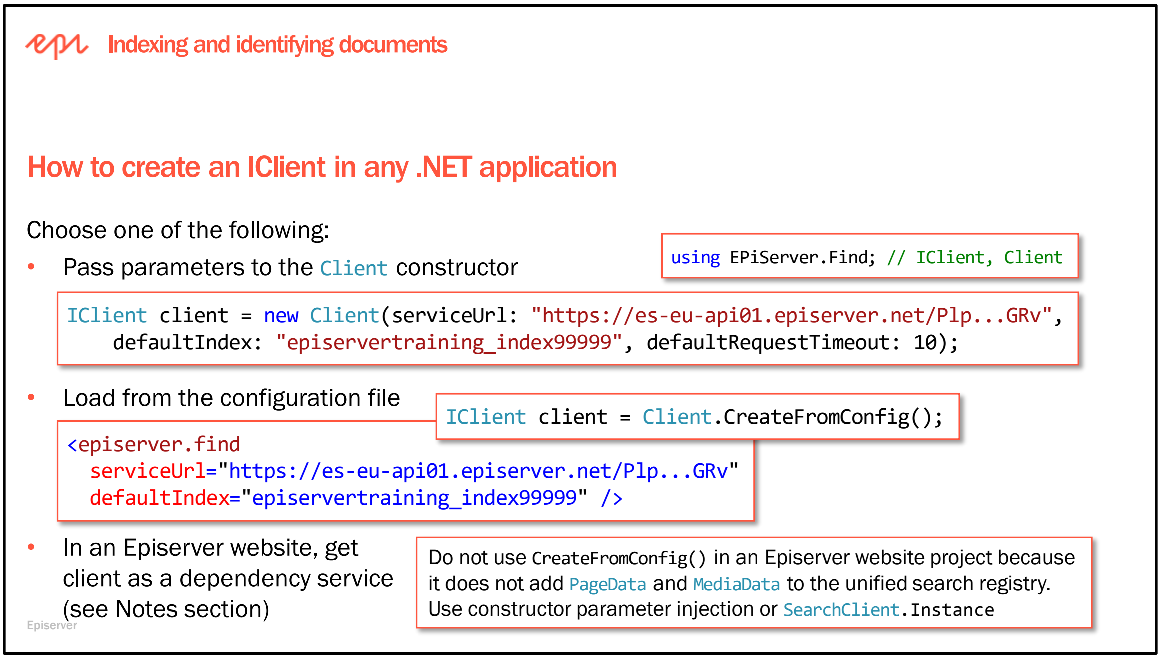
Estimated time: 30 minutes

Prerequisites: Exercise A1

In this exercise, you will:

- Configure an Episerver Find index for use with the AlloyAdvanced website.
- Implement searching functionality using Episerver Find.
- Include optimizations like Best Bets.
- Implement the Powerslice add-on to provide advanced search capabilities for CMS Editors.

Episerver



Indexing and identifying documents

How to create an IClient in any .NET application

Choose one of the following:

- Pass parameters to the Client constructor

```
using EPiServer.Find; // IClient, Client
```

```
IClient client = new Client(serviceUrl: "https://es-eu-api01.episerver.net/Plp...GRv",
    defaultIndex: "episervertraining_index99999", defaultRequestTimeout: 10);
```

- Load from the configuration file

```
IClient client = Client.CreateFromConfig();
```

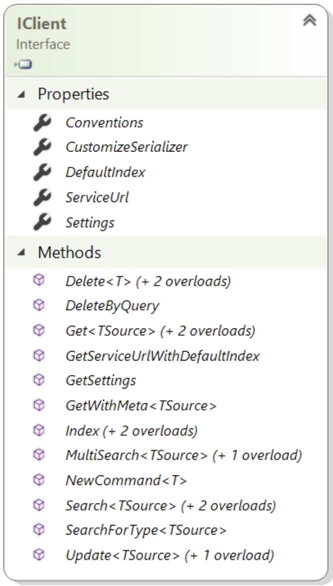
```
<episerver.find
  serviceUrl="https://es-eu-api01.episerver.net/Plp...GRv"
  defaultIndex="episervertraining_index99999" />
```

- In an Episerver website, get client as a dependency service (see Notes section)

Do not use CreateFromConfig() in an Episerver website project because it does not add PageData and MediaData to the unified search registry. Use constructor parameter injection or SearchClient.Instance

Episerver

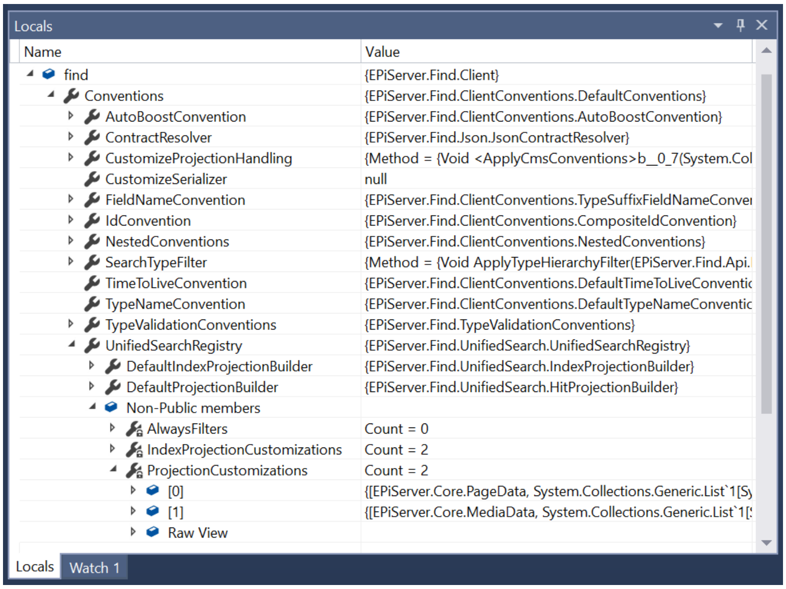
In an Episerver website project, use constructor parameter injection to get IClient so that PageData and MediaData are added to the unified search registry, as shown in the following screenshot:



IClient
Interface

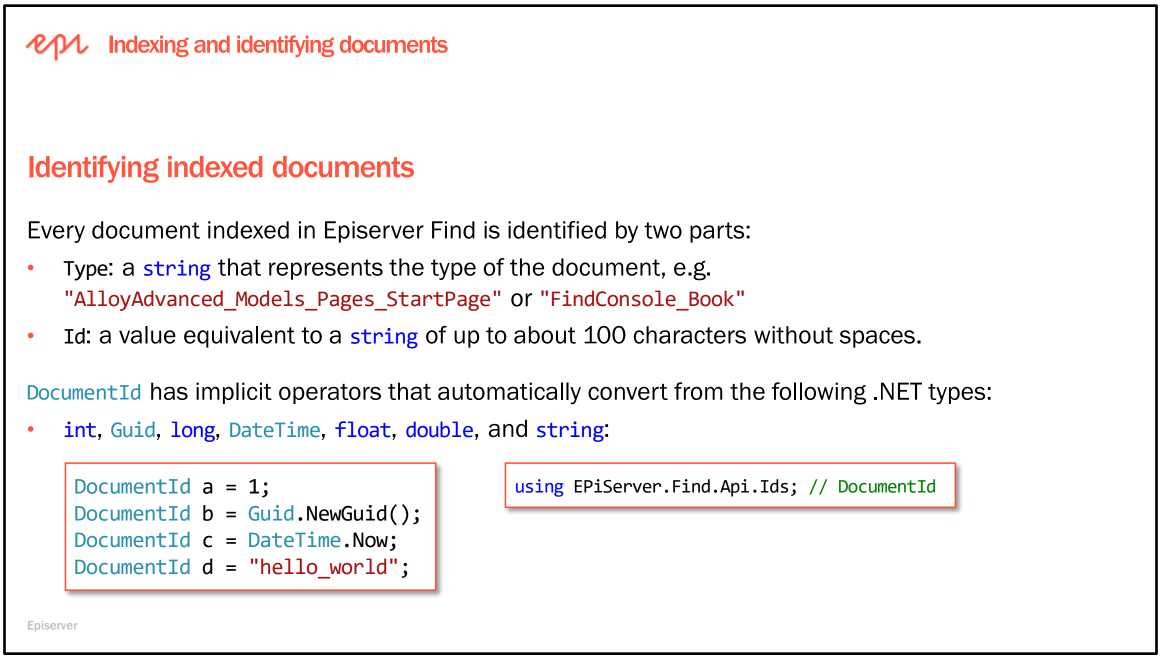
Properties
- Conventions
- CustomizeSerializer
- DefaultIndex
- ServiceUrl
- Settings

Methods
- Delete<T> (+ 2 overloads)
- DeleteByQuery
- Get<TSource> (+ 2 overloads)
- GetServiceUrlWithDefaultIndex
- GetSettings
- GetWithMeta<TSource>
- Index (+ 2 overloads)
- MultiSearch<TSource> (+ 1 overload)
- NewCommand<T>
- Search<TSource> (+ 2 overloads)
- SearchForType<TSource>
- Update<TSource> (+ 1 overload)



Locals

```
Name                                      Value
find                                      {EPiServer.Find.Client}
  Conventions                             {EPiServer.Find.ClientConventions.DefaultConventions}
    AutoBoostConvention                   {EPiServer.Find.ClientConventions.AutoBoostConvention}
    ContractResolver                      {EPiServer.Find.Json.JsonContractResolver}
    CustomizeProjectionHandling           {Method = {Void <ApplyCmsConventions>b__0_7(System.Col
    CustomizeSerializer                   null
    FieldNameConvention                   {EPiServer.Find.ClientConventions.TypeSuffixFieldNameConver
    IdConvention                          {EPiServer.Find.ClientConventions.CompositeIdConvention}
    NestedConventions                     {EPiServer.Find.ClientConventions.NestedConventions}
    SearchTypeFilter                      {Method = {Void ApplyTypeHierarchyFilter(EPiServer.Find.Api.
    TimeToLiveConvention                  {EPiServer.Find.ClientConventions.DefaultTimeToLiveConventic
    TypeNameConvention                    {EPiServer.Find.ClientConventions.DefaultTypeNameConventic
    TypeValidationConventions             {EPiServer.Find.TypeValidationConventions}
    UnifiedSearchRegistry                 {EPiServer.Find.UnifiedSearch.UnifiedSearchRegistry}
      DefaultIndexProjectionBuilder       {EPiServer.Find.UnifiedSearch.IndexProjectionBuilder}
      DefaultProjectionBuilder            {EPiServer.Find.UnifiedSearch.HitProjectionBuilder}
      Non-Public members
        AlwaysFilters                     Count = 0
        IndexProjectionCustomizations     Count = 2
        ProjectionCustomizations          Count = 2
          [0]                             {[EPiServer.Core.PageData, System.Collections.Generic.List`1[Sy
          [1]                             {[EPiServer.Core.MediaData, System.Collections.Generic.List`1[S
          Raw View
```



Indexing and identifying documents

Identifying indexed documents

Every document indexed in Episerver Find is identified by two parts:

- `Type`: a `string` that represents the type of the document, e.g. `"AlloyAdvanced_Models_Pages_StartPage"` or `"FindConsole_Book"`
- `Id`: a value equivalent to a `string` of up to about 100 characters without spaces.

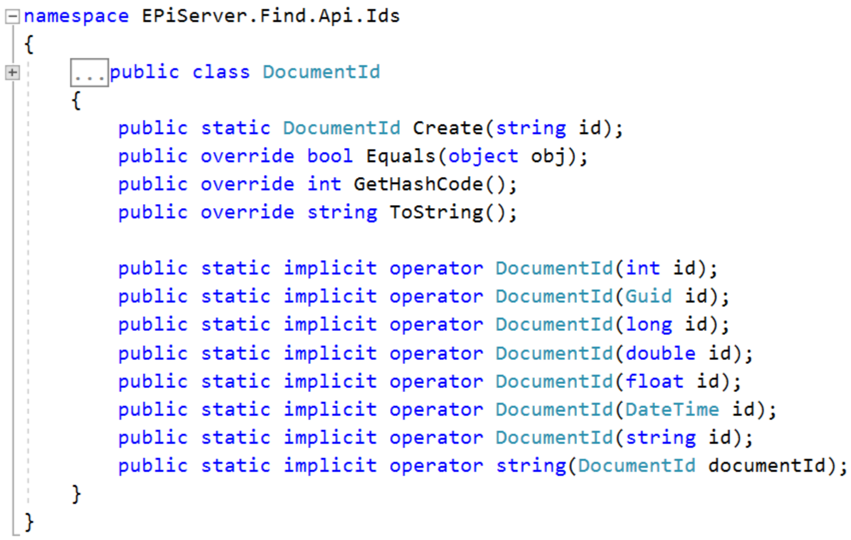
`DocumentId` has implicit operators that automatically convert from the following .NET types:

- `int`, `Guid`, `long`, `DateTime`, `float`, `double`, and `string`:

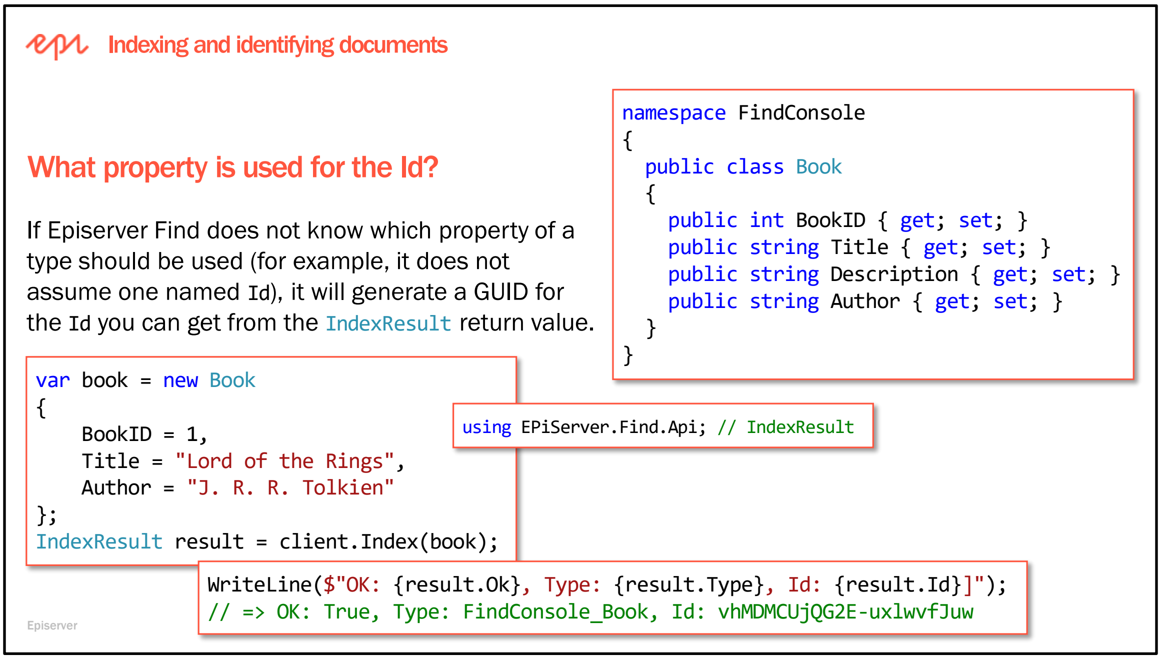
```
DocumentId a = 1;
DocumentId b = Guid.NewGuid();
DocumentId c = DateTime.Now;
DocumentId d = "hello_world";
```

```
using EPiServer.Find.Api.Ids; // DocumentId
```

Episerver



```
namespace EPiServer.Find.Api.Ids
{
    ...public class DocumentId
    {
        public static DocumentId Create(string id);
        public override bool Equals(object obj);
        public override int GetHashCode();
        public override string ToString();

        public static implicit operator DocumentId(int id);
        public static implicit operator DocumentId(Guid id);
        public static implicit operator DocumentId(long id);
        public static implicit operator DocumentId(double id);
        public static implicit operator DocumentId(float id);
        public static implicit operator DocumentId(DateTime id);
        public static implicit operator DocumentId(string id);
        public static implicit operator string(DocumentId documentId);
    }
}
```



Indexing and identifying documents

What property is used for the Id?

If Episerver Find does not know which property of a type should be used (for example, it does not assume one named `Id`), it will generate a GUID for the `Id` you can get from the `IndexResult` return value.

```
namespace FindConsole
{
  public class Book
  {
    public int BookID { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public string Author { get; set; }
  }
}
```

```
var book = new Book
{
    BookID = 1,
    Title = "Lord of the Rings",
    Author = "J. R. R. Tolkien"
};
IndexResult result = client.Index(book);
```

```
using EPiServer.Find.Api; // IndexResult
```

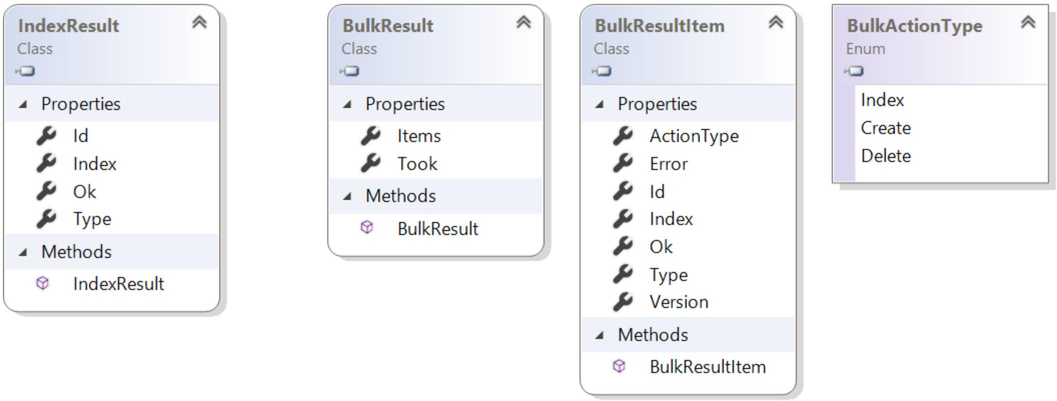
```
WriteLine($"OK: {result.Ok}, Type: {result.Type}, Id: {result.Id}]");
// => OK: True, Type: FindConsole_Book, Id: vhMDMCUjQG2E-uxlwvfJuw
```

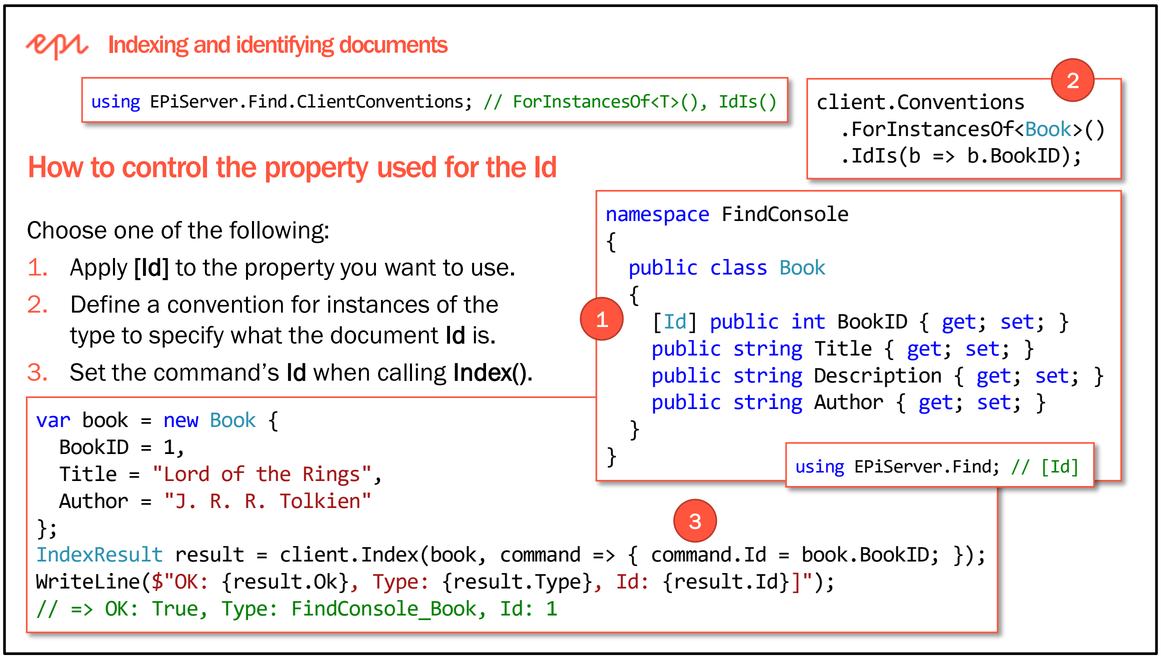
Episerver

Indexing is done using the client’s `Index()` method. Any .NET/CLR object can be indexed as long as it can be serialized to JSON.

It’s possible to index several objects at the same time using overloads of the Index method which has `IEnumerable<object>` or `params object[]` as parameters.

```
BulkResult Index(IEnumerable objectsToIndex);
IndexResult Index(object objectToIndex, Action<IndexCommand> commandAction);
IndexResult Index(object objectToIndex);
```





Indexing and identifying documents

```
using EPiServer.Find.ClientConventions; // ForInstancesOf<T>(), IdIs()
```

```
client.Conventions
  .ForInstancesOf<Book>()
  .IdIs(b => b.BookID);
```

How to control the property used for the Id

Choose one of the following:

1. Apply [Id] to the property you want to use.
2. Define a convention for instances of the type to specify what the document Id is.
3. Set the command's Id when calling Index().

```
namespace FindConsole
{
  public class Book
  {
    [Id] public int BookID { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public string Author { get; set; }
  }
}
```

```
using EPiServer.Find; // [Id]
```

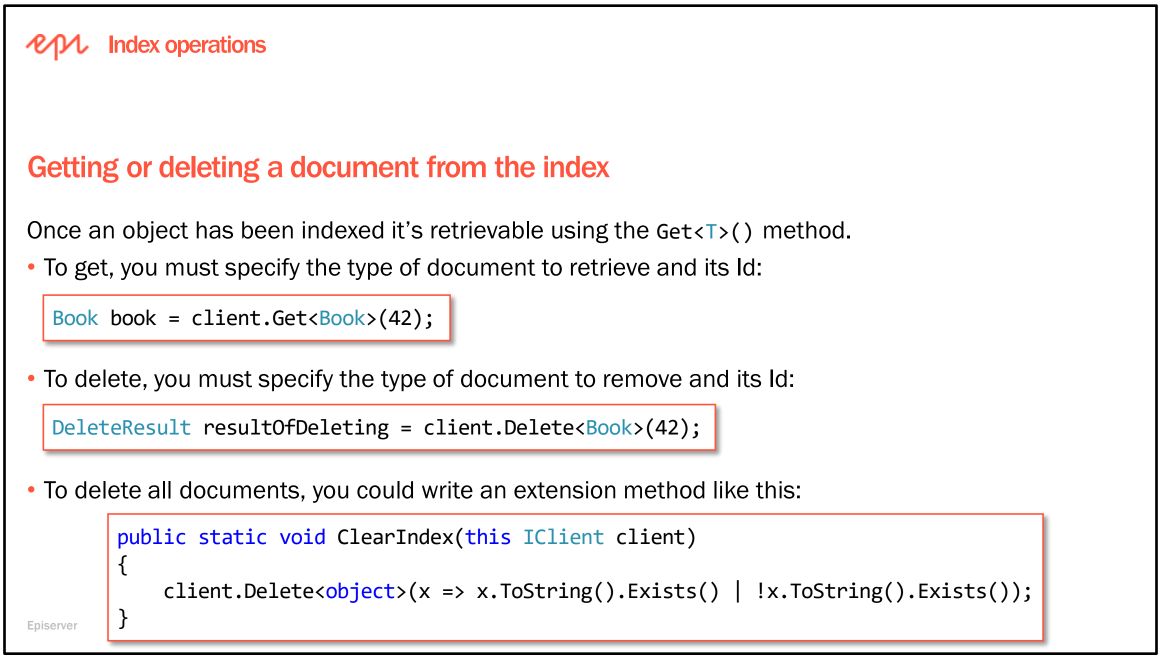
```
var book = new Book {
  BookID = 1,
  Title = "Lord of the Rings",
  Author = "J. R. R. Tolkien"
};
IndexResult result = client.Index(book, command => { command.Id = book.BookID; });
WriteLine($"OK: {result.Ok}, Type: {result.Type}, Id: {result.Id}]");
// => OK: True, Type: FindConsole_Book, Id: 1
```

Controlling the indexing operation

When calling the `Index()` method, you can pass a `command` lambda that controls:

- If the service waits for the index to refresh before returning the index result.
- How long the document will remain in the index.
- Which property is used for the `Id`.

```
IndexResult result = client.Index(book, command =>
{
    command.Refresh = true; // so it appears in results immediately
    command.TimeToLive = TimeSpan.FromMinutes(30); // auto-delete after 30 minutes
    command.Id = book.BookID; // manually set the Id for the document
});
```



Index operations

Getting or deleting a document from the index

Once an object has been indexed it’s retrievable using the `Get<T>()` method.

- To get, you must specify the type of document to retrieve and its Id:

```
Book book = client.Get<Book>(42);
```

- To delete, you must specify the type of document to remove and its Id:

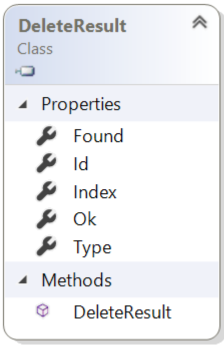
```
DeleteResult resultOfDeleting = client.Delete<Book>(42);
```

- To delete all documents, you could write an extension method like this:

```
public static void ClearIndex(this IClient client)
{
    client.Delete<object>(x => x.ToString().Exists() | !x.ToString().Exists());
}
```

Episerver

```
DeleteResult Delete(Type type, DocumentId id, Action<DeleteCommand> commandAction);
DeleteResult Delete<T>(DocumentId id);
DeleteResult Delete<T>(DocumentId id, Action<DeleteCommand> commandAction);
```

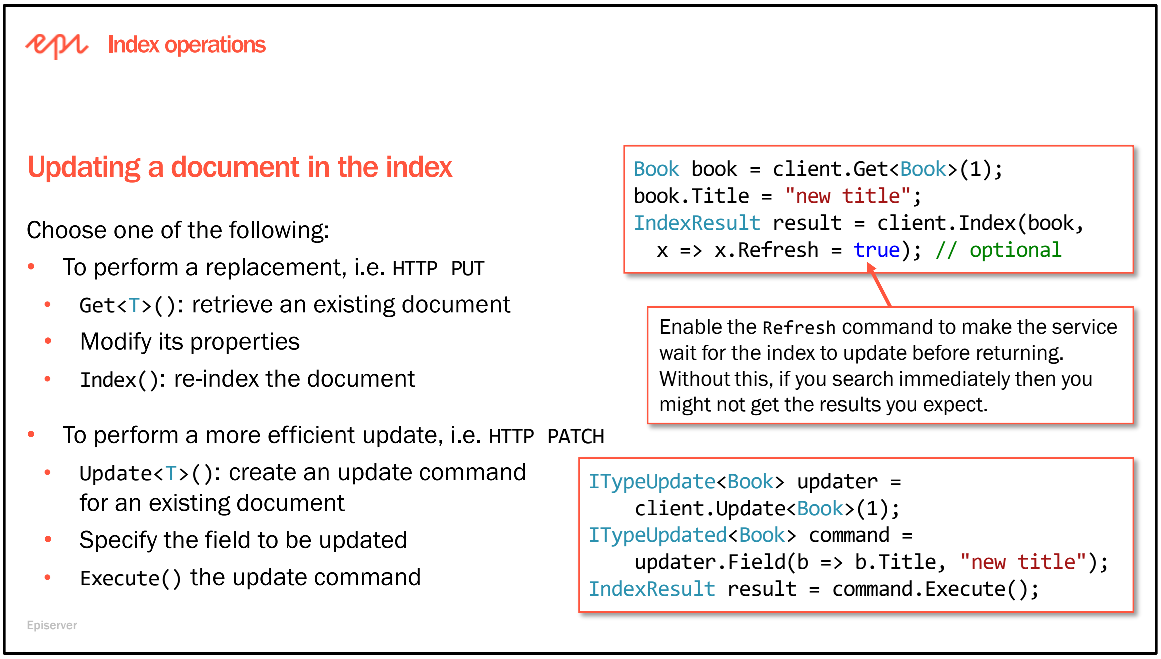


DeleteResult
Class

Properties
- Found
- Id
- Index
- Ok
- Type

Methods
- DeleteResult

```
IEnumerable<GetResult<TSource>> Get<TSource>(IEnumerable<DocumentId> ids);
TSource Get<TSource>(DocumentId id, Action<GetCommand<TSource>> commandAction);
TSource Get<TSource>(DocumentId id);
```



Index operations

Updating a document in the index

Choose one of the following:

- To perform a replacement, i.e. `HTTP PUT`
  - `Get<T>()`: retrieve an existing document
  - Modify its properties
  - `Index()`: re-index the document
- To perform a more efficient update, i.e. `HTTP PATCH`
  - `Update<T>()`: create an update command for an existing document
  - Specify the field to be updated
  - `Execute()` the update command

```
Book book = client.Get<Book>(1);
book.Title = "new title";
IndexResult result = client.Index(book,
  x => x.Refresh = true); // optional
```

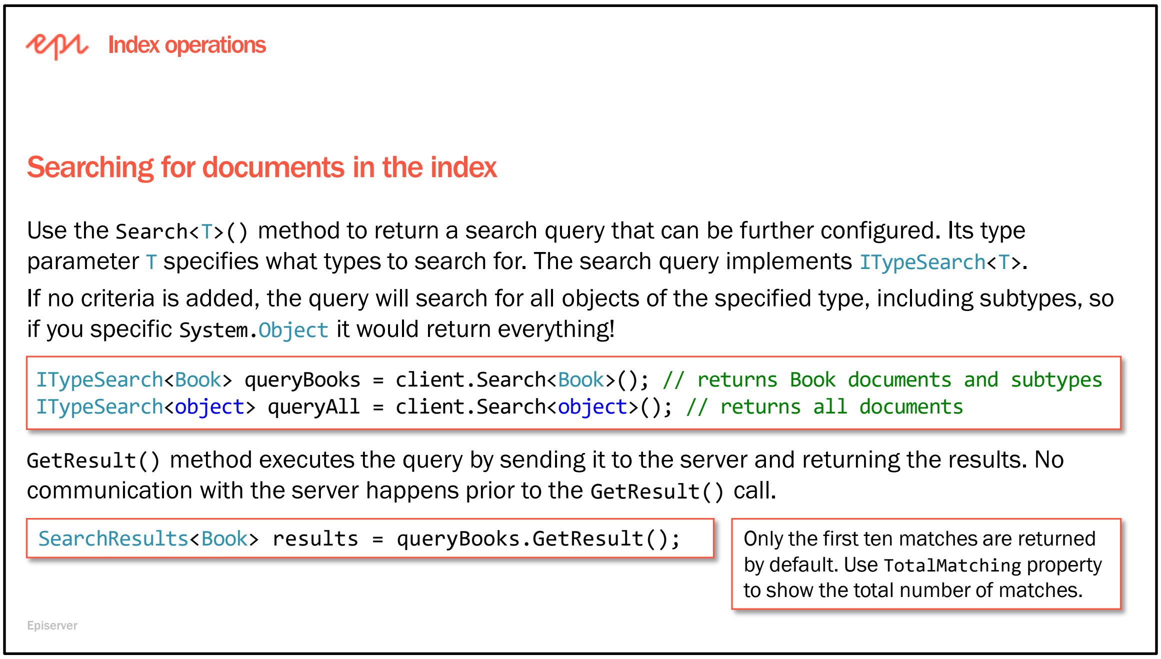
Enable the `Refresh` command to make the service wait for the index to update before returning. Without this, if you search immediately then you might not get the results you expect.

```
ITypeUpdate<Book> updater =
    client.Update<Book>(1);
ITypeUpdated<Book> command =
    updater.Field(b => b.Title, "new title");
IndexResult result = command.Execute();
```

Episerver

Objects which have been indexed can be updated by indexing them again. The index method does not differentiate between adding new objects or updating existing ones. If an document with the same ID exists in the index it will be overwritten, otherwise a new document will be added.

A more efficient method to update a single property is to create an updater as shown in the second example.



Index operations

Searching for documents in the index

Use the `Search<T>()` method to return a search query that can be further configured. Its type parameter `T` specifies what types to search for. The search query implements `ITypeSearch<T>`.

If no criteria is added, the query will search for all objects of the specified type, including subtypes, so if you specific `System.Object` it would return everything!

```
ITypeSearch<Book> queryBooks = client.Search<Book>(); // returns Book documents and subtypes
ITypeSearch<object> queryAll = client.Search<object>(); // returns all documents
```

`GetResult()` method executes the query by sending it to the server and returning the results. No communication with the server happens prior to the `GetResult()` call.

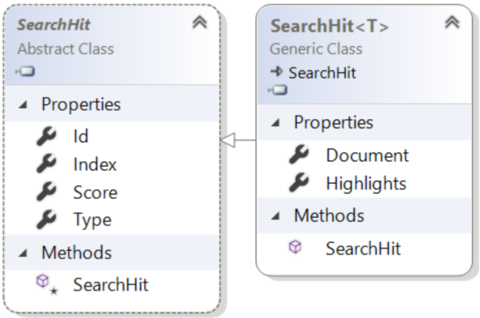
```
SearchResults<Book> results = queryBooks.GetResult();
```

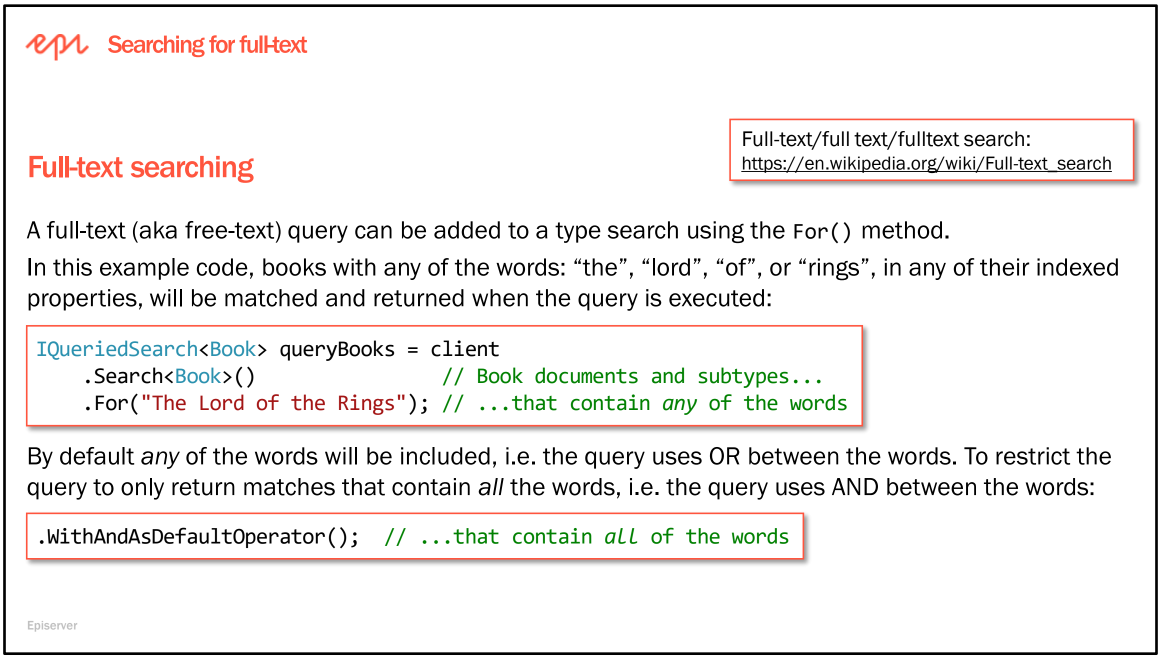
Only the first ten matches are returned by default. Use `TotalMatching` property to show the total number of matches.

Episerver

The `Hits` property on the `SearchResults<T>` contains `SearchHit` objects.

A `SearchHit` contains the `Document`, the `Score`, and `Highlights`. Results are automatically sorted with the highest scoring document first.





Searching for full-text

Full-text searching

Full-text/full text/fulltext search:
https://en.wikipedia.org/wiki/Full-text_search

A full-text (aka free-text) query can be added to a type search using the `For()` method.

In this example code, books with any of the words: “the”, “lord”, “of”, or “rings”, in any of their indexed properties, will be matched and returned when the query is executed:

```
IQueriedSearch<Book> queryBooks = client
    .Search<Book>()                  // Book documents and subtypes...
    .For("The Lord of the Rings"); // ...that contain any of the words
```

By default *any* of the words will be included, i.e. the query uses OR between the words. To restrict the query to only return matches that contain *all* the words, i.e. the query uses AND between the words:

```
.WithAndAsDefaultOperator();  // ...that contain all of the words
```

Episerver

When using the `For()` method each word in the string passed will by default be ORed. Meaning that a string with two words will be interpreted as <word1> OR <word2>. Applied to the above example this means that the query would match a book titled “Lord of the Flies” as it contains the word “lord”.

This is often the desired behavior as a book titled “The Lord of the Rings” would get a higher score and therefore be placed before “Lord of the Flies” in the results. However, in some cases we may want to limit the search results to such that match all keywords in the query.

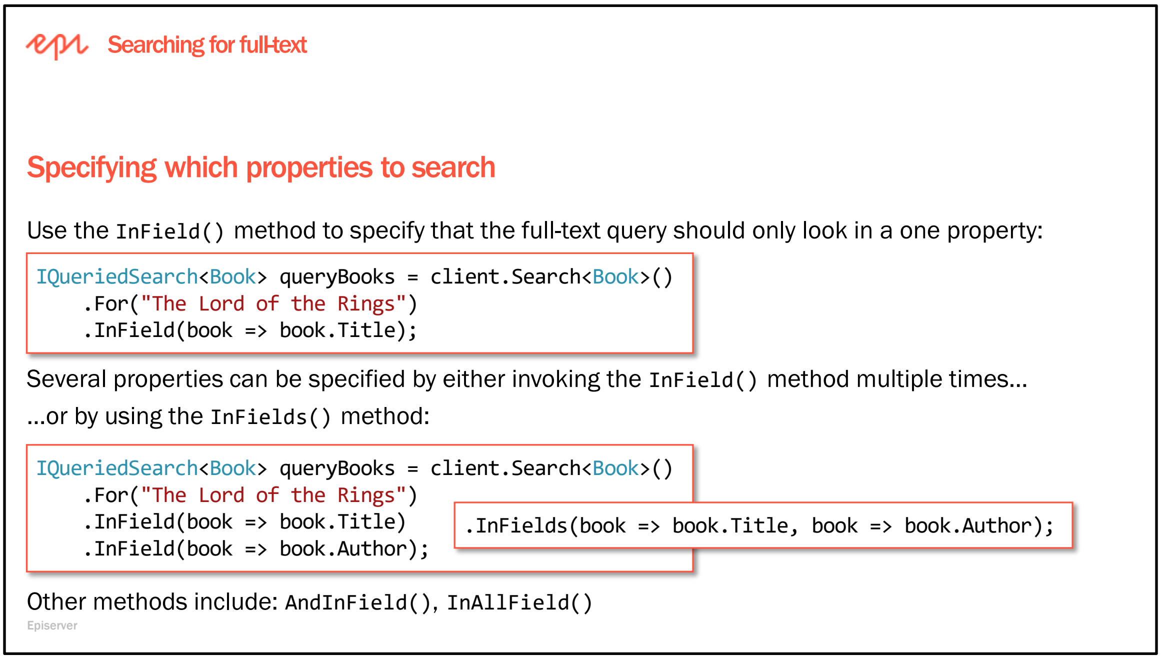
We can then use the `WithAndAsDefaultOperator()` method. For a string with two words passed as argument to the `For()` method will then be interpreted as <word1> AND <word2>.

Recently, Episerver changed the configuration of the indexes to remove all registered stop words, so “of” and “the” are treated the same as “lord” and “rings”.

If you use the `MoreLikeThis()` extension method then you can supply a `StopWords` property with a list of words, but for general queries, remove the stop words using a regular expression before running the query. But explicitly `Track()` using the original query text.

https://world.episerver.com/forum/developer-forum/EPiServer-Search/Thread-Container/2017/10/removing-extra-results-that-use-grammatical-article-words/

https://world.episerver.com/forum/developer-forum/Feature-requests/Thread-Container/2017/1/be-able-to-filter-out-stopwords-for-all-search-not-only-morelike/



Searching for full-text

Specifying which properties to search

Use the `InField()` method to specify that the full-text query should only look in a one property:

```
IQueriedSearch<Book> queryBooks = client.Search<Book>()
    .For("The Lord of the Rings")
    .InField(book => book.Title);
```

Several properties can be specified by either invoking the `InField()` method multiple times…

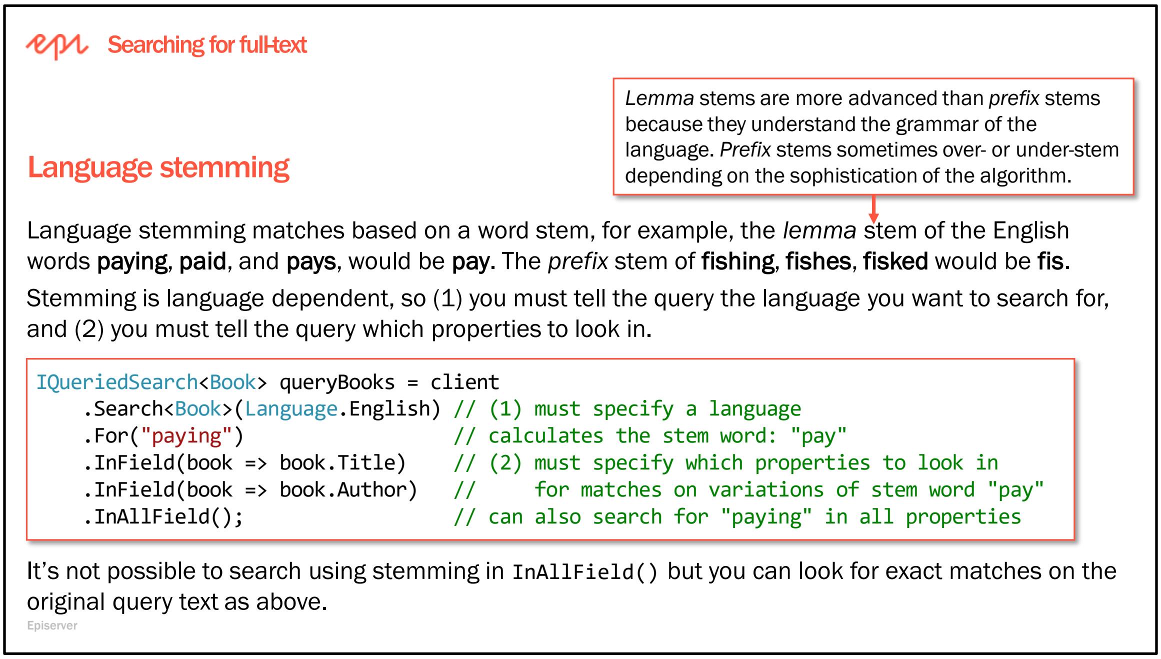
…or by using the `InFields()` method:

```
IQueriedSearch<Book> queryBooks = client.Search<Book>()
    .For("The Lord of the Rings")
    .InField(book => book.Title)
    .InField(book => book.Author);
```

```
.InFields(book => book.Title, book => book.Author);
```

Other methods include: `AndInField()`, `InAllField()`

Episerver



Searching for full-text

Lemma stems are more advanced than *prefix* stems because they understand the grammar of the language. *Prefix* stems sometimes over- or under-stem depending on the sophistication of the algorithm.

Language stemming

Language stemming matches based on a word stem, for example, the *lemma* stem of the English words **paying**, **paid**, and **pays**, would be **pay**. The *prefix* stem of **fishing**, **fishes**, **fisked** would be **fis.**

Stemming is language dependent, so (1) you must tell the query the language you want to search for, and (2) you must tell the query which properties to look in.

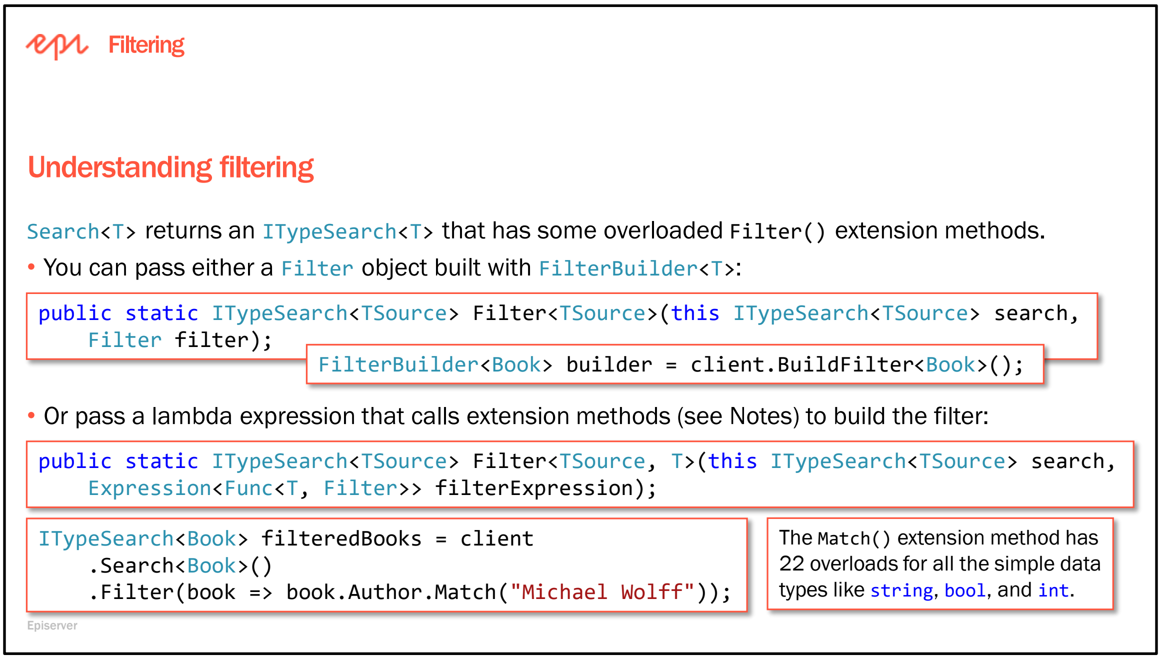
```
IQueriedSearch<Book> queryBooks = client
    .Search<Book>(Language.English) // (1) must specify a language
    .For("paying")                  // calculates the stem word: "pay"
    .InField(book => book.Title)    // (2) must specify which properties to look in
    .InField(book => book.Author)   //     for matches on variations of stem word "pay"
    .InAllField();                  // can also search for "paying" in all properties
```

It's not possible to search using stemming in `InAllField()` but you can look for exact matches on the original query text as above.

Episerver

Understanding stemming
http://www.elastic.co/guide/en/elasticsearch/guide/current/stemming.html

Lemmatization
A lemma is the canonical, or dictionary, form of a set of related words—the lemma of paying, paid, and pays is pay. Usually the lemma resembles the words it is related to but sometimes it doesn't – the lemma of is, was, am, and being is be.



Filtering

Understanding filtering

`Search<T>` returns an `ITypeSearch<T>` that has some overloaded `Filter()` extension methods.

- You can pass either a `Filter` object built with `FilterBuilder<T>`:

```
public static ITypeSearch<TSource> Filter<TSource>(this ITypeSearch<TSource> search,
    Filter filter);
```

```
FilterBuilder<Book> builder = client.BuildFilter<Book>();
```

- Or pass a lambda expression that calls extension methods (see Notes) to build the filter:

```
public static ITypeSearch<TSource> Filter<TSource, T>(this ITypeSearch<TSource> search,
    Expression<Func<T, Filter>> filterExpression);
```

```
ITypeSearch<Book> filteredBooks = client
    .Search<Book>()
    .Filter(book => book.Author.Match("Michael Wolff"));
```

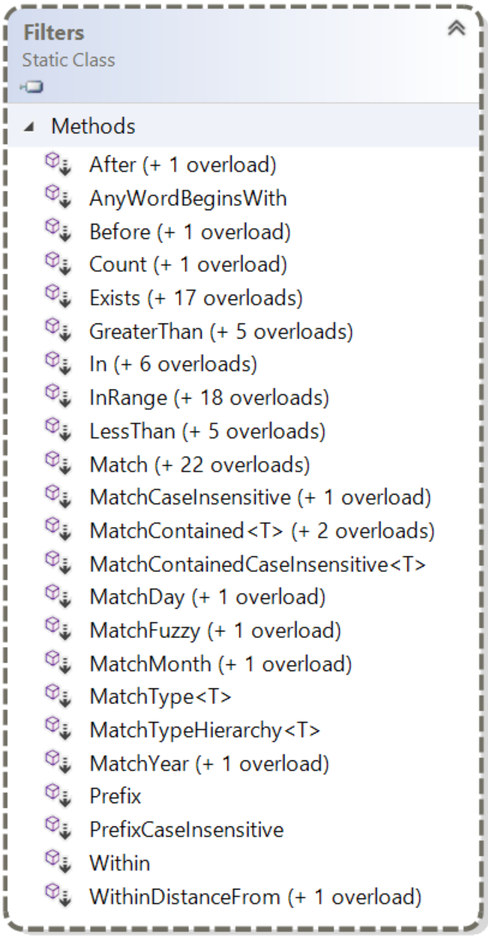
The `Match()` extension method has 22 overloads for all the simple data types like `string`, `bool`, and `int`.

Episerver

When we want to find only documents that matches a specific condition we can use filters. As opposed to full-text queries, filters either match completely or not at all. That is, while full-text queries (and other types of queries) rank documents by score where one document can match the query a lot and another just a little and both are returned, filter does not produce or affect scoring.

The Filter method is quite similar to the Where method in LINQ. It does however have a slightly different syntax as it requires an expression that returns a Filter object instead of a Boolean value. When using the Filter method we typically use the Match method in the filter expression to match a value exactly, or for lists of objects implementing IEnumerable, to match require one of the objects in the list to match a value.

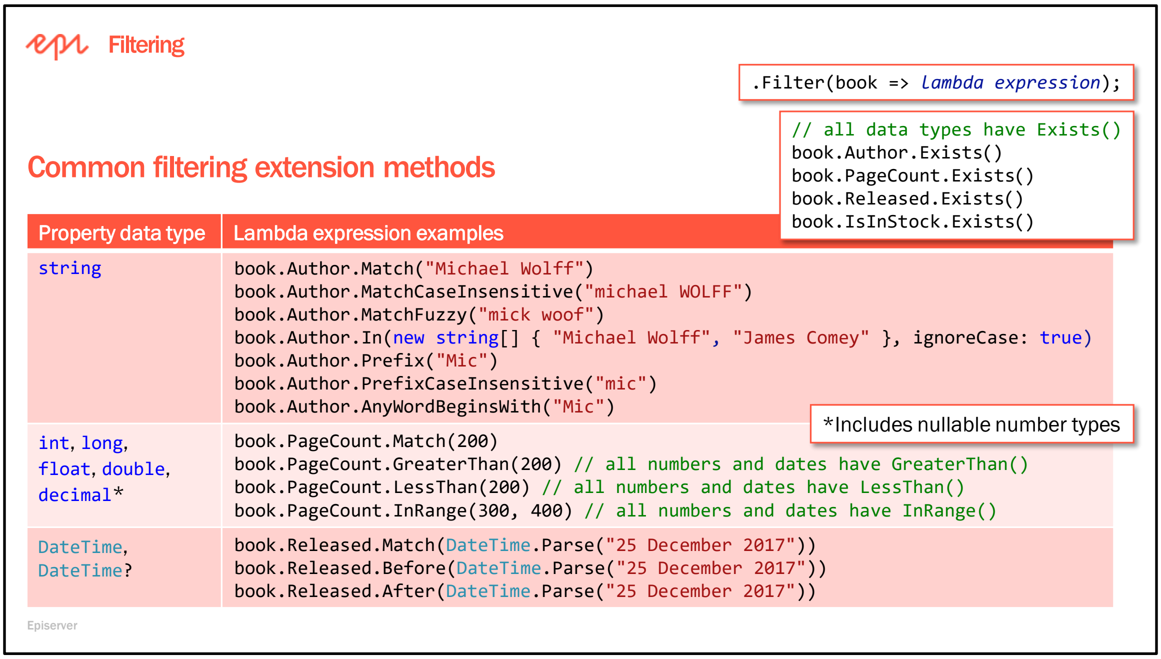
As filter expressions are not executed “as-is” but parsed and sent over to the search engine we generally don’t have to do null-checks like we would with in-memory LINQ queries. For instance, with an expression such as x => x.Author.Prefix(“A”) it doesn’t matter if the Author property has a value or not.
It’s possible to extend Finds filtering API by creating custom filter methods. For instance, if we often use x => x.Author.Prefix(“A”) we could create a method that allows us to instead write x => x.AuthorNameStartsWithA().



Filters
Static Class

Methods

- After (+ 1 overload)
- AnyWordBeginsWith
- Before (+ 1 overload)
- Count (+ 1 overload)
- Exists (+ 17 overloads)
- GreaterThan (+ 5 overloads)
- In (+ 6 overloads)
- InRange (+ 18 overloads)
- LessThan (+ 5 overloads)
- Match (+ 22 overloads)
- MatchCaseInsensitive (+ 1 overload)
- MatchContained<T> (+ 2 overloads)
- MatchContainedCaseInsensitive<T>
- MatchDay (+ 1 overload)
- MatchFuzzy (+ 1 overload)
- MatchMonth (+ 1 overload)
- MatchType<T>
- MatchTypeHierarchy<T>
- MatchYear (+ 1 overload)
- Prefix
- PrefixCaseInsensitive
- Within
- WithinDistanceFrom (+ 1 overload)



Filtering

```
.Filter(book => lambda expression);
```

```
// all data types have Exists()
book.Author.Exists()
book.PageCount.Exists()
book.Released.Exists()
book.IsInStock.Exists()
```

Common filtering extension methods

Property data type	Lambda expression examples

string	book.Author.Match("Michael Wolff")
	book.Author.MatchCaseInsensitive("michael WOLFF")
	book.Author.MatchFuzzy("mick woof")
	book.Author.In(new string[] { "Michael Wolff", "James Comey" }, ignoreCase: true)
	book.Author.Prefix("Mic")
	book.Author.PrefixCaseInsensitive("mic")
	book.Author.AnyWordBeginsWith("Mic")

int, long,
float, double,
decimal*
	book.PageCount.Match(200)
	book.PageCount.GreaterThan(200) // all numbers and dates have GreaterThan()
	book.PageCount.LessThan(200) // all numbers and dates have LessThan()
	book.PageCount.InRange(300, 400) // all numbers and dates have InRange()

DateTime,
DateTime?
	book.Released.Match(DateTime.Parse("25 December 2017"))
	book.Released.Before(DateTime.Parse("25 December 2017"))
	book.Released.After(DateTime.Parse("25 December 2017"))

*Includes nullable number types

Episerver

Filtering string properties

String properties can be filtered in a number of ways. For exact matching we can use the Match method and for the equivalent of String.StartsWith we can use the Prefix method. Both methods are case sensitive but have corresponding methods for case insensitive filtering.
The Exists method matches properties which have any value.

NOTE: The AnyWordBeginsWith method while powerful isn't optimal in terms of performance when used for large strings. It's therefore best to limit its usage to short string fields such as titles, names, tags and the like.

Filtering numbers and date/time values

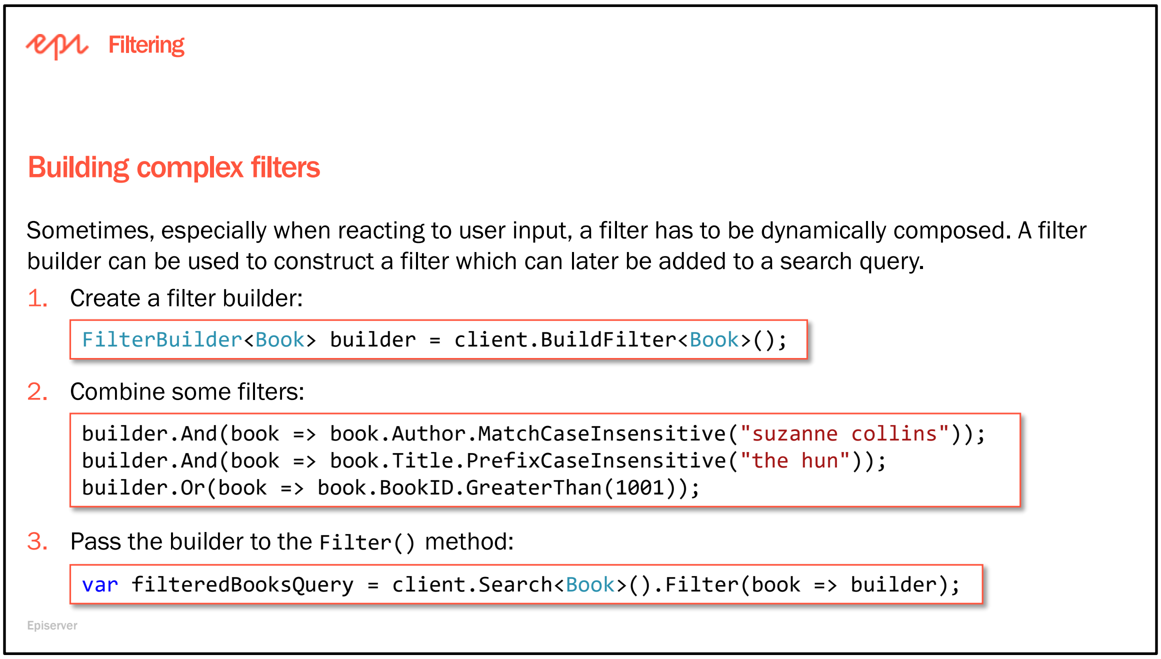
Numerical values such as integers, doubles, floats, longs, and DateTime values as well as their nullable equivalents can be matched by equality using the Match method and for existence using the Exists method. It's also possible to require that a value is within a certain range using the InRange method.

Filtering other data types

- Booleans
- Enum
- Type
- Nested fields
- Collections
- Complex objects

Property data type	Lambda expression for filter examples
bool	book.IsInStock.Match(true)

IEnumerable<string>
IEnumerable<int>
	book.Authors.Count(2)
	book.Authors.In("Michael Wolff")



Filtering

Building complex filters

Sometimes, especially when reacting to user input, a filter has to be dynamically composed. A filter builder can be used to construct a filter which can later be added to a search query.

1. Create a filter builder:

```
FilterBuilder<Book> builder = client.BuildFilter<Book>();
```

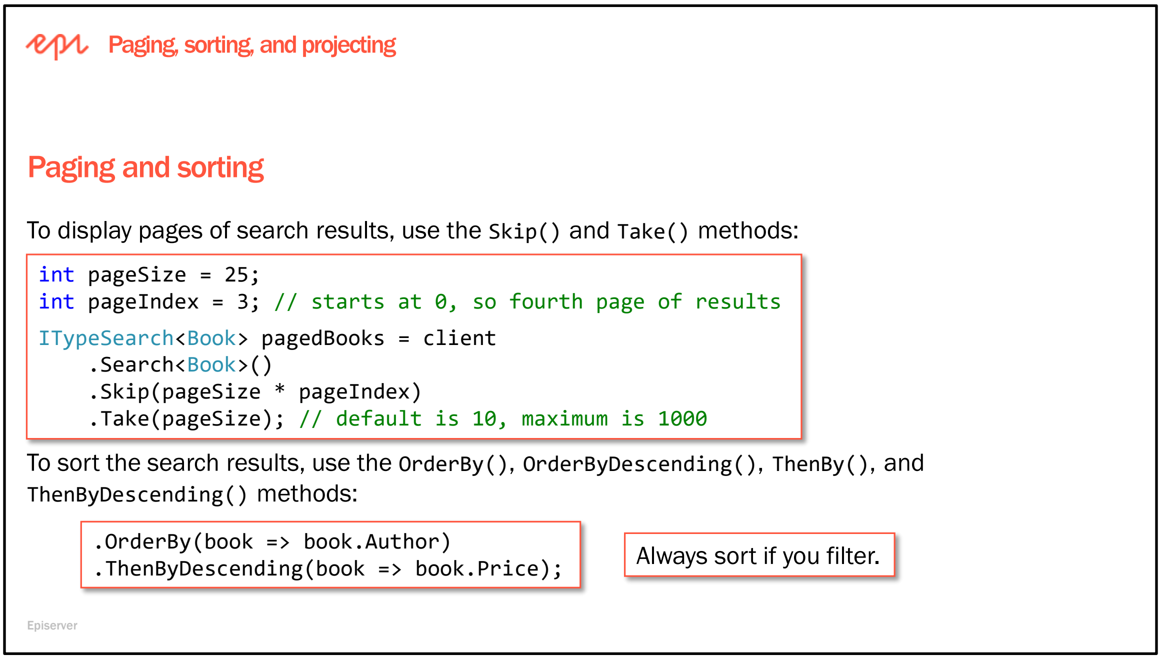
2. Combine some filters:

```
builder.And(book => book.Author.MatchCaseInsensitive("suzanne collins"));
builder.And(book => book.Title.PrefixCaseInsensitive("the hun"));
builder.Or(book => book.BookID.GreaterThan(1001));
```

3. Pass the builder to the `Filter()` method:

```
var filteredBooksQuery = client.Search<Book>().Filter(book => builder);
```

Episerver



Paging, sorting, and projecting

Paging and sorting

To display pages of search results, use the `Skip()` and `Take()` methods:

```
int pageSize = 25;
int pageIndex = 3; // starts at 0, so fourth page of results
ITypeSearch<Book> pagedBooks = client
    .Search<Book>()
    .Skip(pageSize * pageIndex)
    .Take(pageSize); // default is 10, maximum is 1000
```

To sort the search results, use the `OrderBy()`, `OrderByDescending()`, `ThenBy()`, and `ThenByDescending()` methods:

```
.OrderBy(book => book.Author)
.ThenByDescending(book => book.Price);
```

Always sort if you filter.

Episerver

Paging

The `Skip()` method bypasses the first n hits that match a search query while the `Take()` method instructs the search engine to return n number of hits. They are used together when presenting search results and listings with paging. `Take()` is also often used alone when we're only interested in a limited number of hits.

As opposed to LINQ and most database querying solutions, Find defaults to the equivalent of `Take(10)`. If you don't specify the number of hits to return using Take() then you only get the first 10 hits. Also note that `Take()` will throw an exception if we pass it a value larger than 1000. To get all results you must use paging.

Sorting

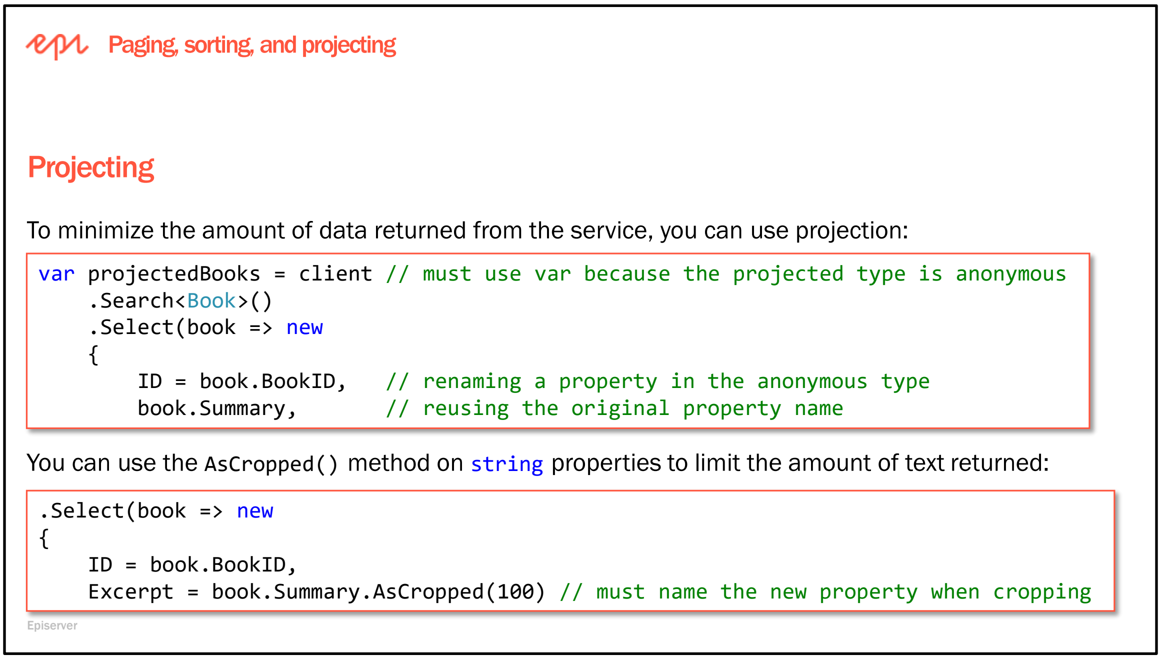
For sorting, use `OrderBy()` and `OrderByDescending()`. There are also `ThenBy()` and `ThenByDescending()` methods which are simply aliases for the two former methods and are only used to make the code more easily readable.

Sorting null values

`OrderBy()` orders null values last while `OrderByDescending()` orders them first. This default behavior can be changed by supplying a second argument of type `SortMissing`.

```
.OrderBy(book => book.Author, SortMissing.First)
```

As the sorting is done on the server it's safe to sort on fields that could potentially be null. For instance `.OrderBy(x => x.A.B.C)` won't cause an exception if either A, B or C are null. Note however that sorting on fields that have never been created can raise exceptions from the search engine. That is, while A may be null in the example, at least one object should have had a non-null value for A.



Paging, sorting, and projecting

Projecting

To minimize the amount of data returned from the service, you can use projection:

```
var projectedBooks = client // must use var because the projected type is anonymous
    .Search<Book>()
    .Select(book => new
    {
        ID = book.BookID,    // renaming a property in the anonymous type
        book.Summary,        // reusing the original property name
```

You can use the `AsCropped()` method on `string` properties to limit the amount of text returned:

```
.Select(book => new
{
    ID = book.BookID,
    Excerpt = book.Summary.AsCropped(100) // must name the new property when cropping
```

Episerver

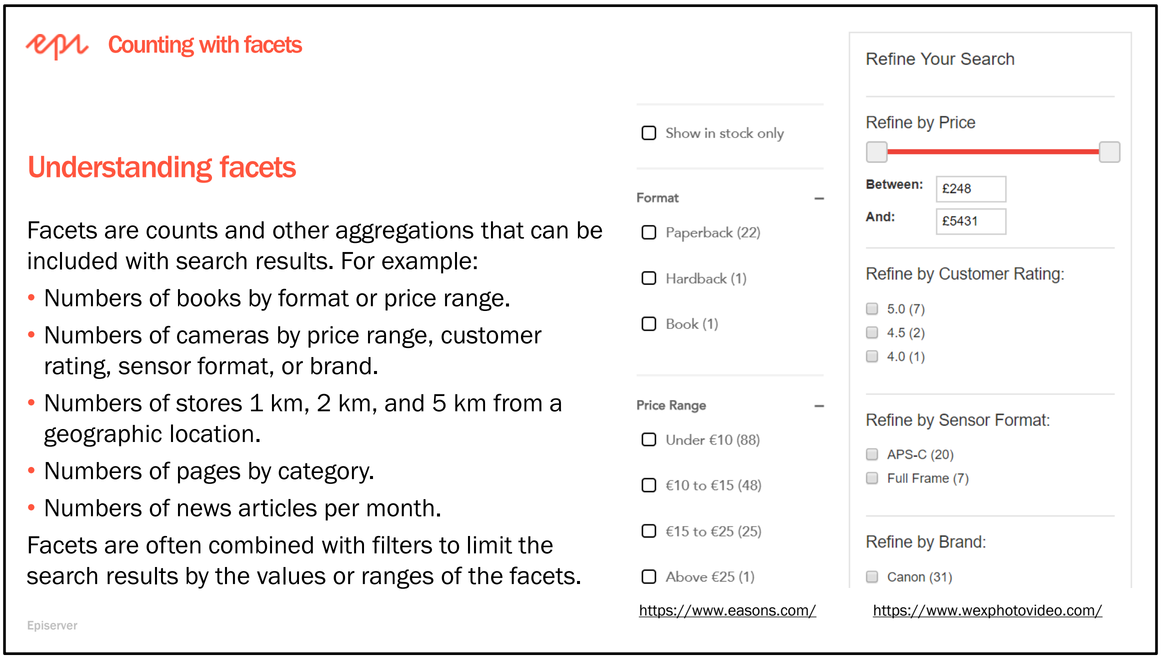
There are three reasons why we'd use projections:

1. Only the required fields need to be transferred from the search engine server resulting in a smaller response.

2. We can make the result object contain a list of objects tailored for our needs, such as data needed for presentation in a search results listing.

3. Some types may be hard to deserialize from JSON and by using a projection we can work around that. For instance, while Find's Episerver CMS integration enables indexing PageData objects it does not allow deserializing them.

It's possible to use a couple of special methods in projection expressions. One such is the AsCropped method which is an extension method for strings. When using this method only the first n characters of the string will be returned from the search engine. The search engine will do it's best to crop at the end of a word.



Counting with facets

Understanding facets

Facets are counts and other aggregations that can be included with search results. For example:

- Numbers of books by format or price range.
- Numbers of cameras by price range, customer rating, sensor format, or brand.
- Numbers of stores 1 km, 2 km, and 5 km from a geographic location.
- Numbers of pages by category.
- Numbers of news articles per month.

Facets are often combined with filters to limit the search results by the values or ranges of the facets.

Episerver

□ Show in stock only

Format –

□ Paperback (22)

□ Hardback (1)

□ Book (1)

Price Range –

□ Under €10 (88)

□ €10 to €15 (48)

□ €15 to €25 (25)

□ Above €25 (1)

https://www.easons.com/

Refine Your Search

Refine by Price

Between: £248

And: £5431

Refine by Customer Rating:

□ 5.0 (7)

□ 4.5 (2)

□ 4.0 (1)

Refine by Sensor Format:

□ APS-C (20)

□ Full Frame (7)

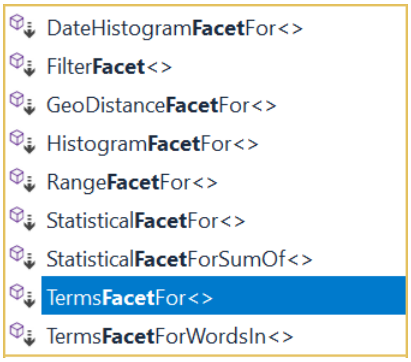
Refine by Brand:

□ Canon (31)

https://www.wexphotovideo.com/

Episerver Find features several facets ranging from simple to advanced, such as Terms, Range, Statistical and Geo Distance.

https://world.episerver.com/documentation/developer-guides/find/NET-Client-API/searching/Facets/



DateHistogram**Facet**For<>

Filter**Facet**<>

GeoDistance**Facet**For<>

Histogram**Facet**For<>

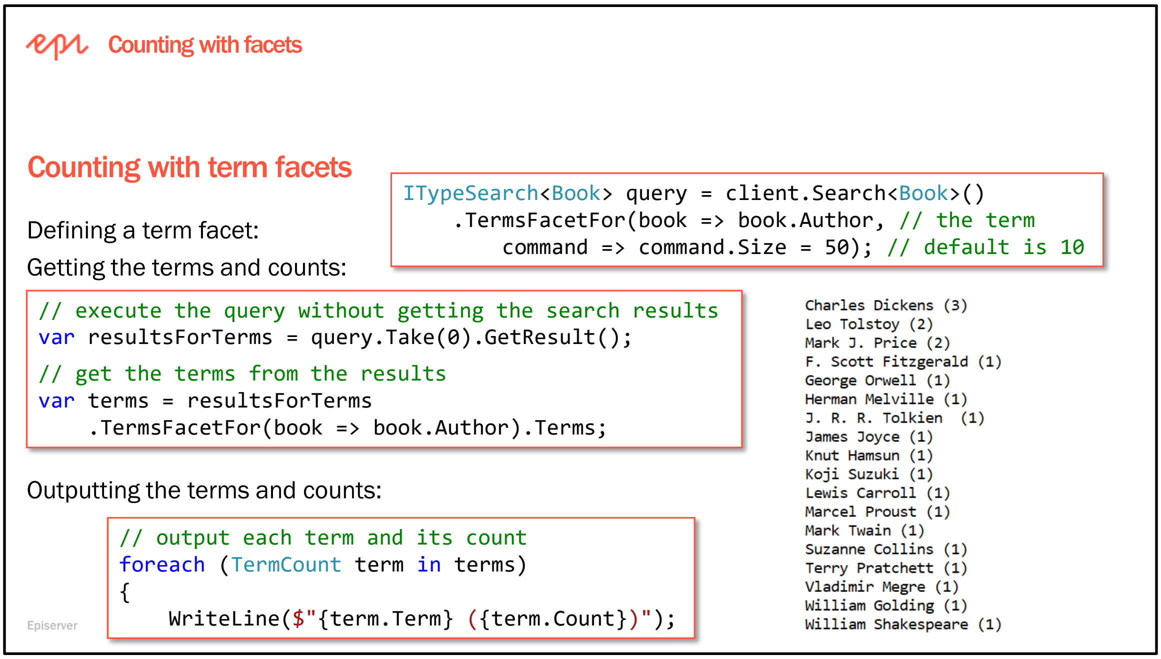
Range**Facet**For<>

Statistical**Facet**For<>

Statistical**Facet**ForSumOf<>

Terms**Facet**For<>

Terms**Facet**ForWordsIn<>



Counting with facets

Counting with term facets

Defining a term facet:

```
ITypeSearch<Book> query = client.Search<Book>()
    .TermsFacetFor(book => book.Author, // the term
        command => command.Size = 50); // default is 10
```

Getting the terms and counts:

```
// execute the query without getting the search results
var resultsForTerms = query.Take(0).GetResult();
// get the terms from the results
var terms = resultsForTerms
    .TermsFacetFor(book => book.Author).Terms;
```

Outputting the terms and counts:

```
// output each term and its count
foreach (TermCount term in terms)
{
    WriteLine($"{term.Term} ({term.Count})");
```

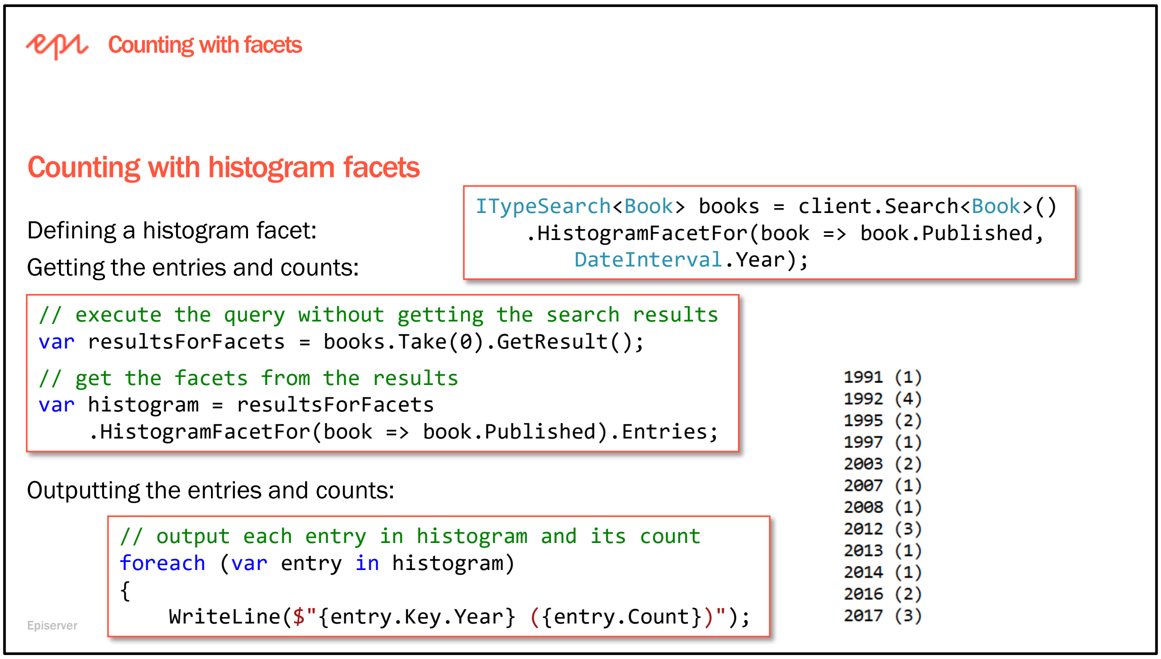
```
Charles Dickens (3)
Leo Tolstoy (2)
Mark J. Price (2)
F. Scott Fitzgerald (1)
George Orwell (1)
Herman Melville (1)
J. R. R. Tolkien  (1)
James Joyce (1)
Knut Hamsun (1)
Koji Suzuki (1)
Lewis Carroll (1)
Marcel Proust (1)
Mark Twain (1)
Suzanne Collins (1)
Terry Pratchett (1)
Vladimir Megre (1)
William Golding (1)
William Shakespeare (1)
```

Episerver

Perhaps the most common type of facets is terms facets. Terms facets provide a grouping of a specific field within the documents that match a search request. This is typically used to display a list of categories, tags, department names etc. We pass TermsFacetFor an expression to specify what field we want a facet for. The search result will contain a terms facet in addition to the regular search hits. It's possible to customize to request for the facet by passing a second argument to the TermsFacetFor method. By doing so we can specify that the facet should contain more than 10 items: .TermsFacetFor(x => x.Author, x => x.Size = 50)

As it's possible to request multiple terms facets within the same search request we must again pass an expression specifying what field the facet is for. The returned object from TermsFacetFor implements IEnumerable<TermCount>. TermCount objects have a Term property, containing the value in the field, and a Count property, containing the number of documents that has that specific value.

https://world.episerver.com/documentation/developer-guides/find/NET-Client-API/searching/Facets/Terms-facets/



Counting with facets

Counting with histogram facets

Defining a histogram facet:

```
ITypeSearch<Book> books = client.Search<Book>()
    .HistogramFacetFor(book => book.Published,
        DateInterval.Year);
```

Getting the entries and counts:

```
// execute the query without getting the search results
var resultsForFacets = books.Take(0).GetResult();

// get the facets from the results
var histogram = resultsForFacets
    .HistogramFacetFor(book => book.Published).Entries;
```

Outputting the entries and counts:

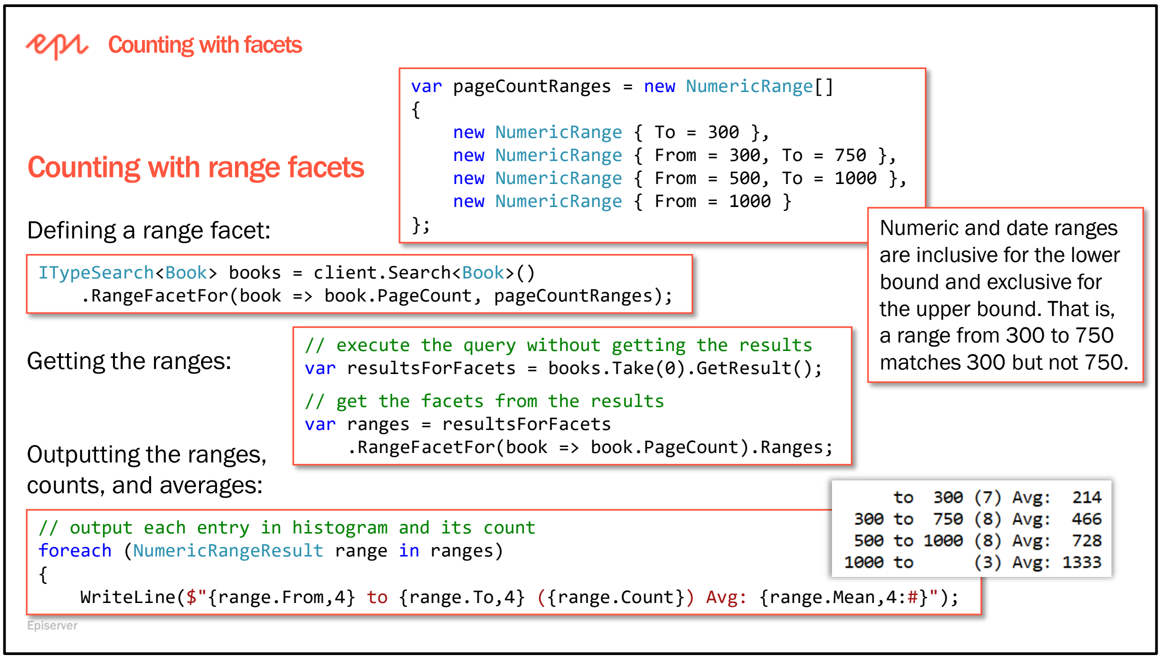
```
// output each entry in histogram and its count
foreach (var entry in histogram)
{
    WriteLine($"{entry.Key.Year} ({entry.Count})");
```

```
1991 (1)
1992 (4)
1995 (2)
1997 (1)
2003 (2)
2007 (1)
2008 (1)
2012 (3)
2013 (1)
2014 (1)
2016 (2)
2017 (3)
```

Episerver

Use histogram facets with numerical and date fields to retrieve the number of documents whose field value falls within an interval. For example, in a search of products, use a histogram facet to retrieve the number of products whose price ranges from 0 to 100, 101 to 200, and so on.

https://world.episerver.com/documentation/developer-guides/find/NET-Client-API/searching/Facets/Histogram-facets/



Counting with facets

Counting with range facets

```
var pageCountRanges = new NumericRange[]
{
    new NumericRange { To = 300 },
    new NumericRange { From = 300, To = 750 },
    new NumericRange { From = 500, To = 1000 },
    new NumericRange { From = 1000 }
};
```

Defining a range facet:

```
ITypeSearch<Book> books = client.Search<Book>()
    .RangeFacetFor(book => book.PageCount, pageCountRanges);
```

Numeric and date ranges are inclusive for the lower bound and exclusive for the upper bound. That is, a range from 300 to 750 matches 300 but not 750.

Getting the ranges:

```
// execute the query without getting the results
var resultsForFacets = books.Take(0).GetResult();

// get the facets from the results
var ranges = resultsForFacets
    .RangeFacetFor(book => book.PageCount).Ranges;
```

Outputting the ranges, counts, and averages:

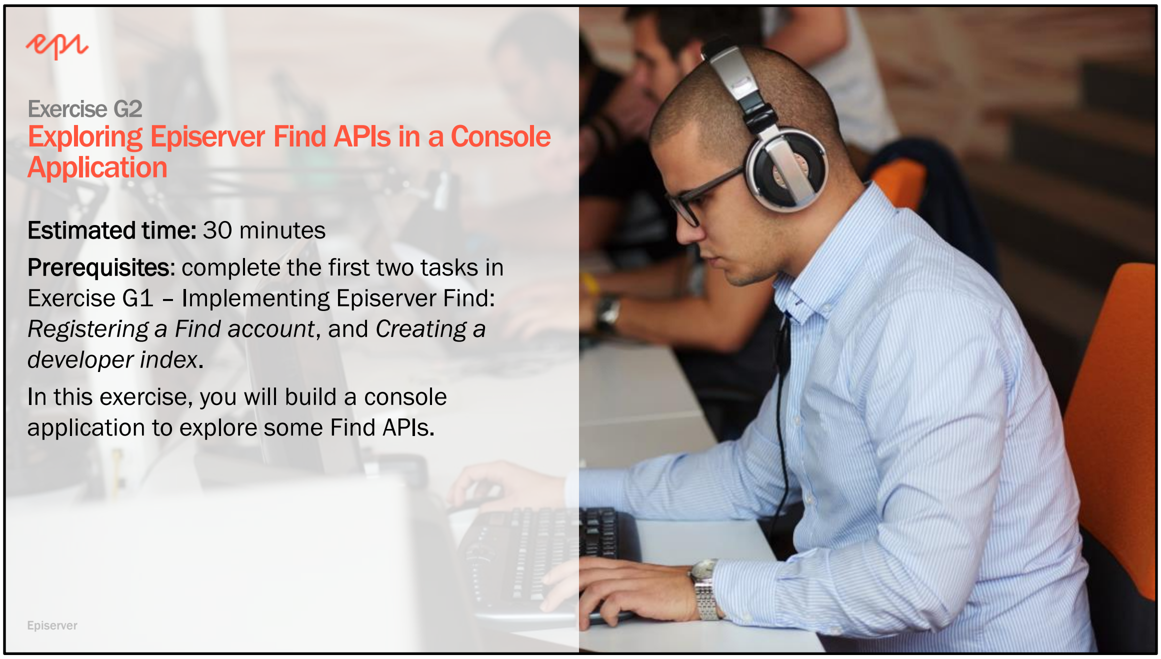
```
// output each entry in histogram and its count
foreach (NumericRangeResult range in ranges)
{
    WriteLine($"{range.From,4} to {range.To,4} ({range.Count}) Avg: {range.Mean,4:#}");
```

```
     to  300 (7) Avg:  214
 300 to  750 (8) Avg:  466
 500 to 1000 (8) Avg:  728
1000 to      (3) Avg: 1333
```

Episerver

Range facets group documents based on ranges into which a numeric or DateTime field falls. Unlike histogram facets, the ranges need not be an interval, such as 0-10, 10-20. Instead, they can be different sizes and overlap each other, such as 0-10, 5-20.

https://world.episerver.com/documentation/developer-guides/find/NET-Client-API/searching/Facets/Range-facets/



Exercise G2

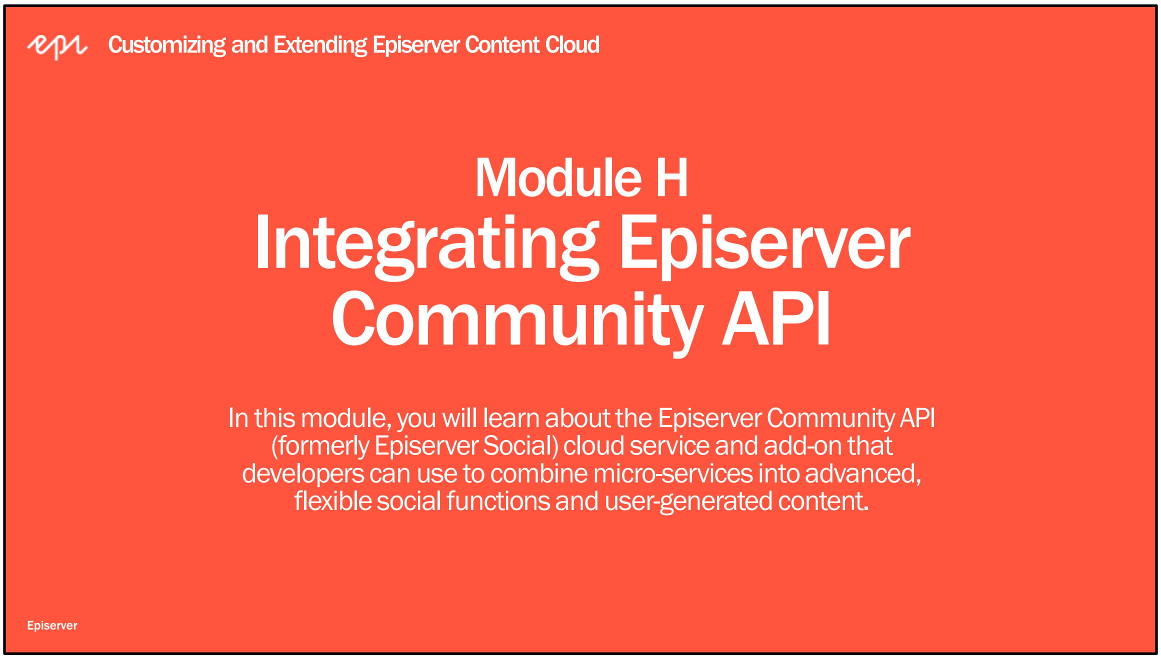
Exploring Episerver Find APIs in a Console Application

Estimated time: 30 minutes

Prerequisites: complete the first two tasks in Exercise G1 – Implementing Episerver Find: *Registering a Find account*, and *Creating a developer index*.

In this exercise, you will build a console application to explore some Find APIs.

Episerver

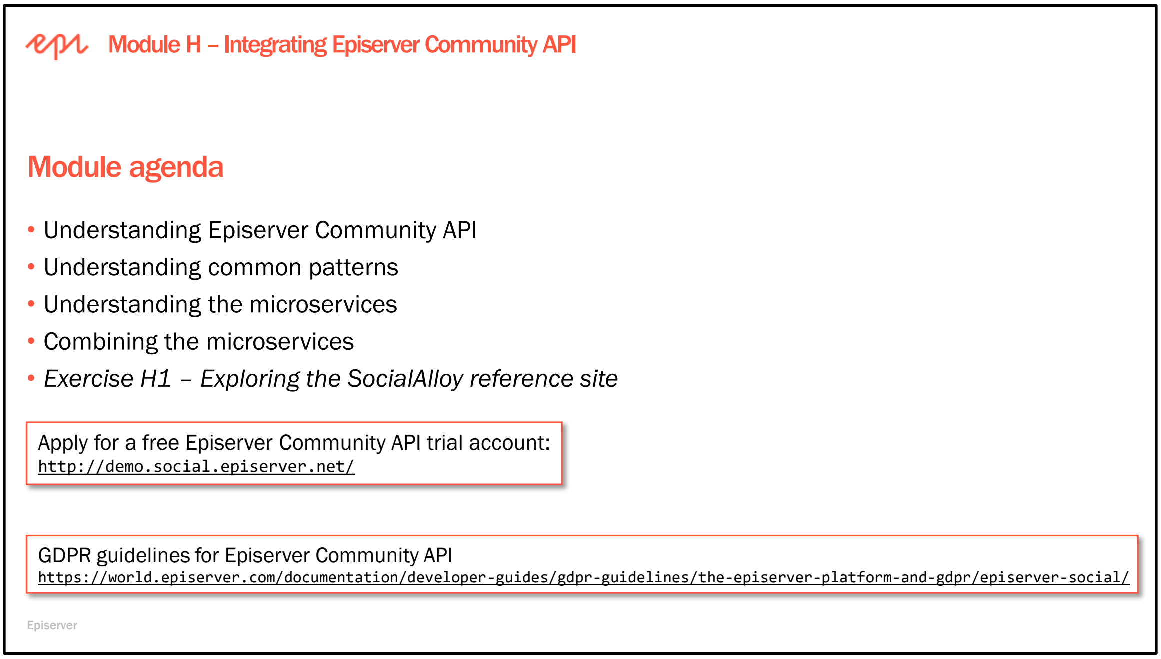


Customizing and Extending Episerver Content Cloud

Module H
Integrating Episerver Community API

In this module, you will learn about the Episerver Community API (formerly Episerver Social) cloud service and add-on that developers can use to combine micro-services into advanced, flexible social functions and user-generated content.

Episerver



Module H – Integrating Episerver Community API

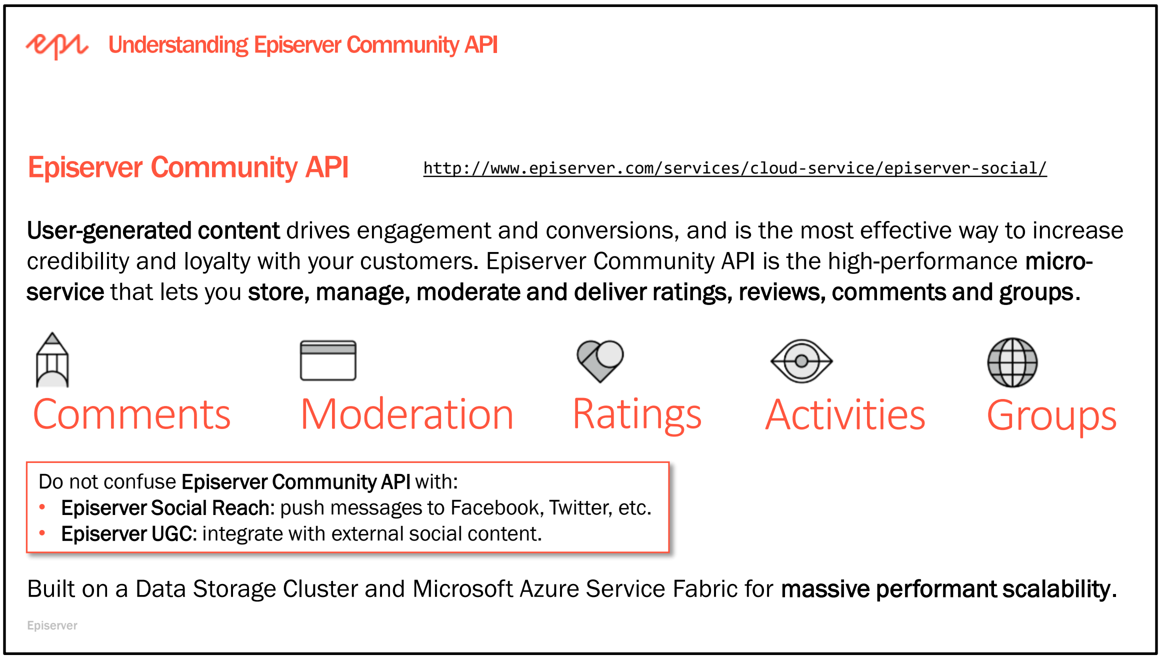
Module agenda

- Understanding Episerver Community API
- Understanding common patterns
- Understanding the microservices
- Combining the microservices
- *Exercise H1 – Exploring the SocialAlloy reference site*

Apply for a free Episerver Community API trial account:
http://demo.social.episerver.net/

GDPR guidelines for Episerver Community API
https://world.episerver.com/documentation/developer-guides/gdpr-guidelines/the-episerver-platform-and-gdpr/episerver-social/

Episerver



Understanding Episerver Community API

Episerver Community API http://www.episerver.com/services/cloud-service/episerver-social/

User-generated content drives engagement and conversions, and is the most effective way to increase credibility and loyalty with your customers. Episerver Community API is the high-performance **micro-service** that lets you **store, manage, moderate and deliver ratings, reviews, comments and groups.**

Comments Moderation Ratings Activities Groups

Do not confuse **Episerver Community API** with:
- **Episerver Social Reach:** push messages to Facebook, Twitter, etc.
- **Episerver UGC**: integrate with external social content.

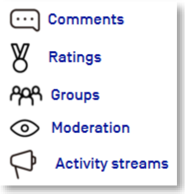
Built on a Data Storage Cluster and Microsoft Azure Service Fabric for **massive performant scalability.**

Episerver

Episerver Community API PaaS for developers

Episerver **Community API** platform is a collection of extensible micro-services for defining and collecting user community generated content.

- Comments - manage and deliver hierarchical, user-generated content
- Ratings - allow users to quantify the value of your content and products
- Groups - aggregate users and content to build digital communities
- Moderation - review and control user contributions
- Activity Streams - broadcast your audience's engagement with your application

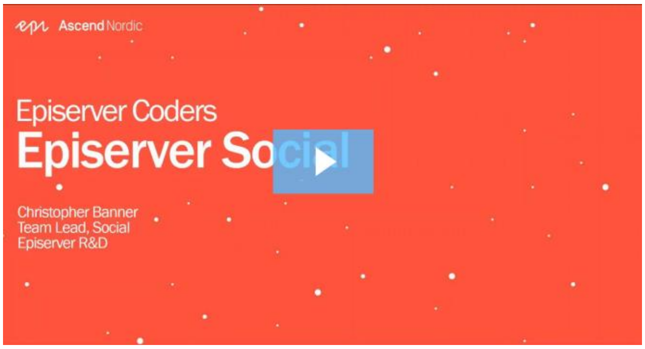


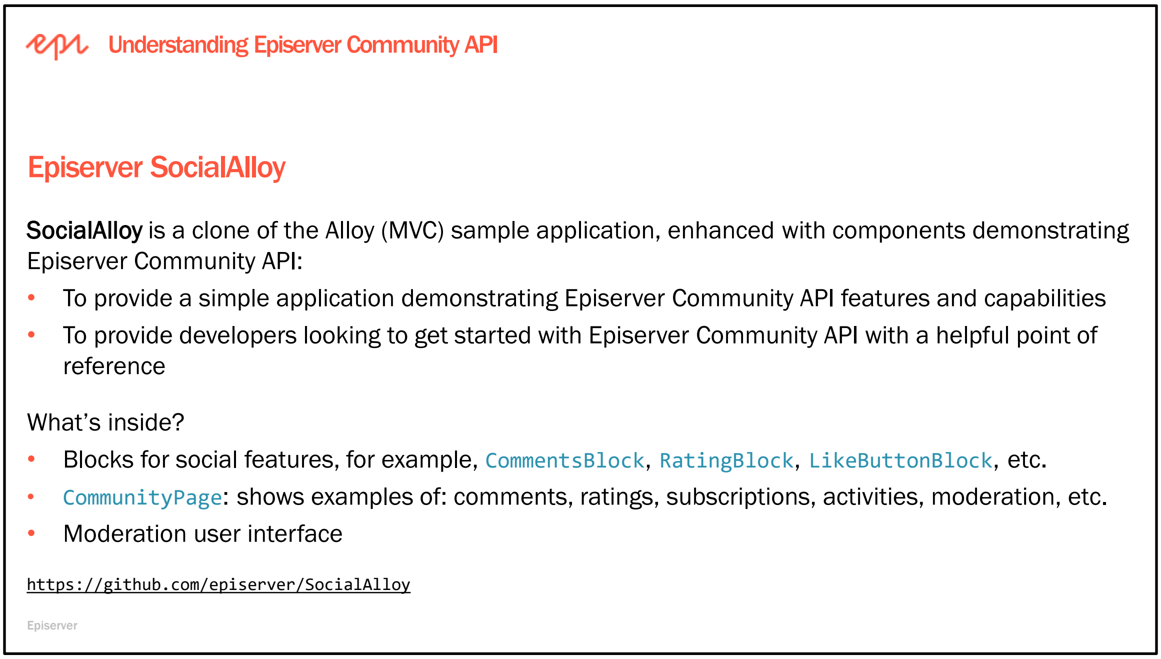
Comments
Ratings
Groups
Moderation
Activity streams

Episerver Community API Developer Guide

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

Video (64 minutes): http://fast.wistia.net/embed/iframe/b7x5k8odd4?videoFoam=true





Understanding Episerver Community API

Episerver SocialAlloy

SocialAlloy is a clone of the Alloy (MVC) sample application, enhanced with components demonstrating Episerver Community API:

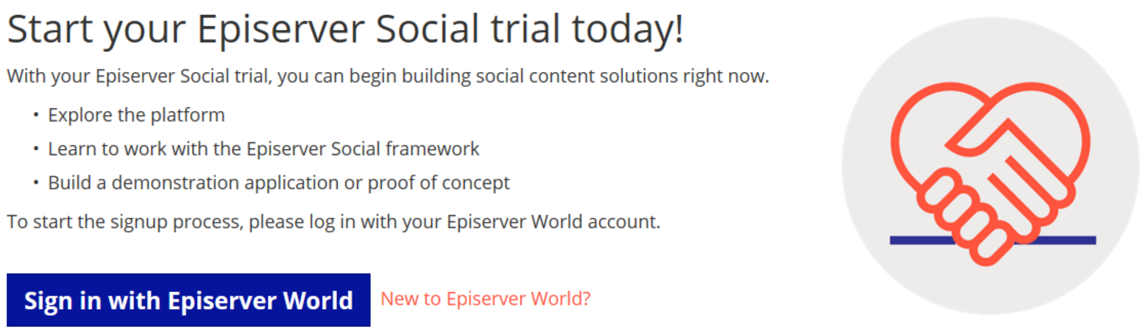
- To provide a simple application demonstrating Episerver Community API features and capabilities
- To provide developers looking to get started with Episerver Community API with a helpful point of reference

What's inside?

- Blocks for social features, for example, `CommentsBlock`, `RatingBlock`, `LikeButtonBlock`, etc.
- `CommunityPage`: shows examples of: comments, ratings, subscriptions, activities, moderation, etc.
- Moderation user interface

https://github.com/episerver/SocialAlloy

Episerver

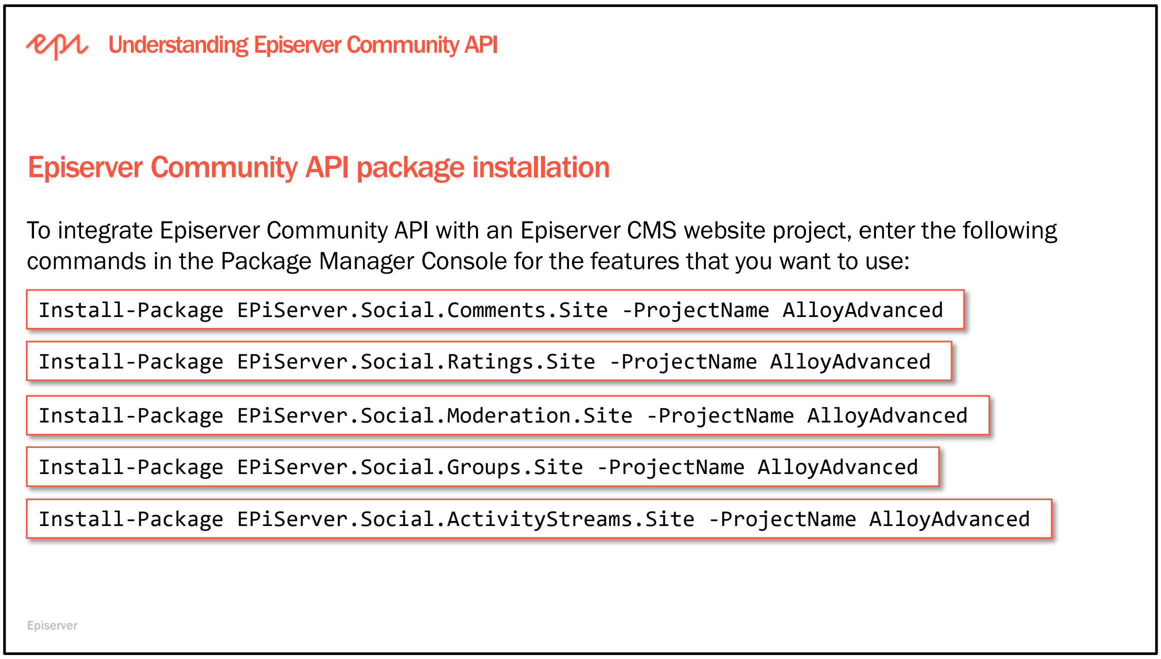


Start your Episerver Social trial today!

With your Episerver Social trial, you can begin building social content solutions right now.

- Explore the platform
- Learn to work with the Episerver Social framework
- Build a demonstration application or proof of concept

To start the signup process, please log in with your Episerver World account.



Understanding Episerver Community API

Episerver Community API package installation

To integrate Episerver Community API with an Episerver CMS website project, enter the following commands in the Package Manager Console for the features that you want to use:

```
Install-Package EPiServer.Social.Comments.Site -ProjectName AlloyAdvanced
```

```
Install-Package EPiServer.Social.Ratings.Site -ProjectName AlloyAdvanced
```

```
Install-Package EPiServer.Social.Moderation.Site -ProjectName AlloyAdvanced
```

```
Install-Package EPiServer.Social.Groups.Site -ProjectName AlloyAdvanced
```

```
Install-Package EPiServer.Social.ActivityStreams.Site -ProjectName AlloyAdvanced
```

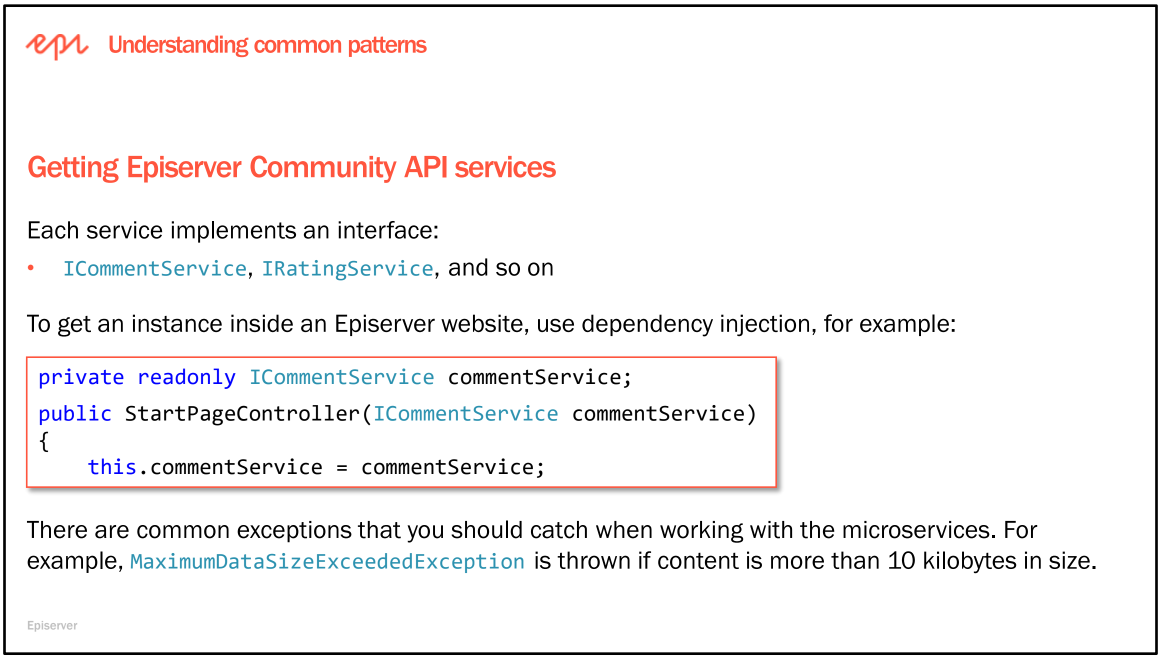
Episerver

To configure Episerver Community API, copy and paste from the email you were sent for your account:

```
<episerver.social>
  <settings timeout="100000"/>
  <authentication appId="your-application-id" secret="your-application-secret"/>
  <endpoints>
    <add name="Comments" value="https://..." />
    <add name="Ratings" value="https://..." />
    <add name="Moderation" value="https://..." />
    <add name="ActivityStreams" value="https://..." />
    <add name="Groups" value="https://..." />
  </endpoints>
</episerver.social>
```

Your application's **appId** and **secret** are private to your application. This information should not be committed to a source control repository or otherwise publicly exposed.

It is essential that the server hosting your application maintains accurate time. When the server time is inaccurate, requests are created with inaccurate timestamps. As a result, these requests may be rejected as unauthentic.



Understanding common patterns

Getting Episerver Community API services

Each service implements an interface:

- ICommentService, IRatingService, and so on

To get an instance inside an Episerver website, use dependency injection, for example:

```
private readonly ICommentService commentService;
public StartPageController(ICommentService commentService)
{
    this.commentService = commentService;
```

There are common exceptions that you should catch when working with the microservices. For example, MaximumDataSizeExceededException is thrown if content is more than 10 kilobytes in size.

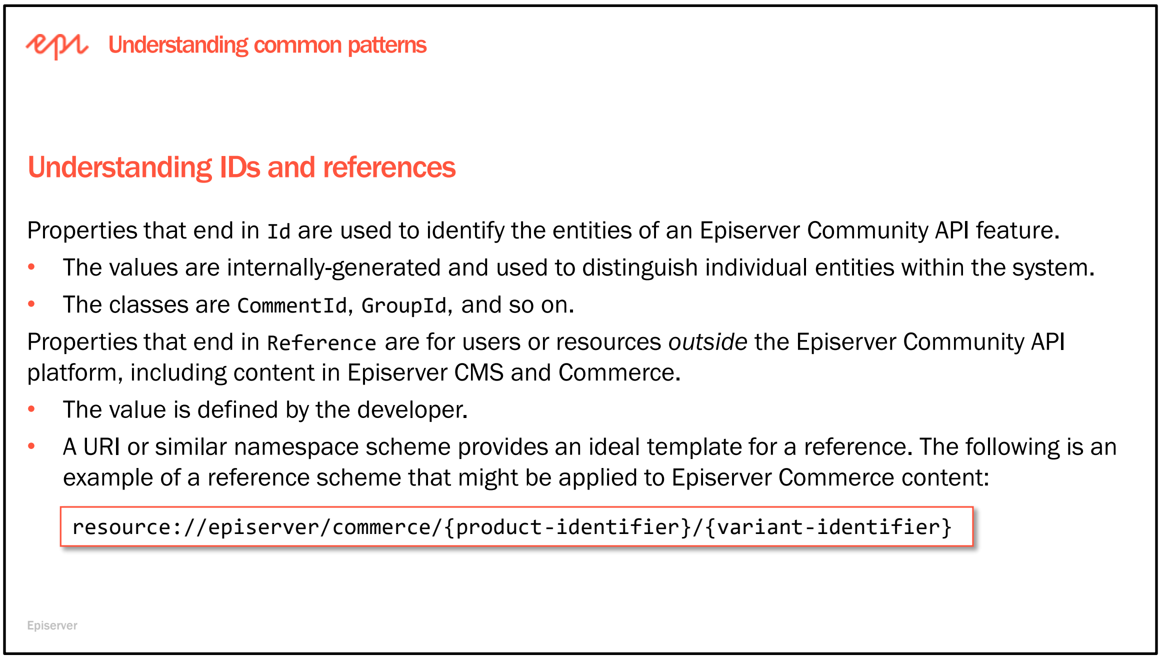
Episerver

Episerver Community API exceptions

Common exceptions thrown include:

- SocialAuthenticationException: misconfiguration, server time out-of-sync, and so on.
- MaximumDataSizeExceededException: if social content is more than 10 kilobytes in size.
- RateLimitExceededException: if you issue too many requests over a short period of time.
- SocialCommunicationException: if an application cannot connect or communicate with Episerver **Community API** platform cloud services.
- SocialException: unexpected errors.

The individual services may also throw exceptions that are unique to the feature that they implement.



Understanding common patterns

Understanding IDs and references

Properties that end in `Id` are used to identify the entities of an Episerver Community API feature.

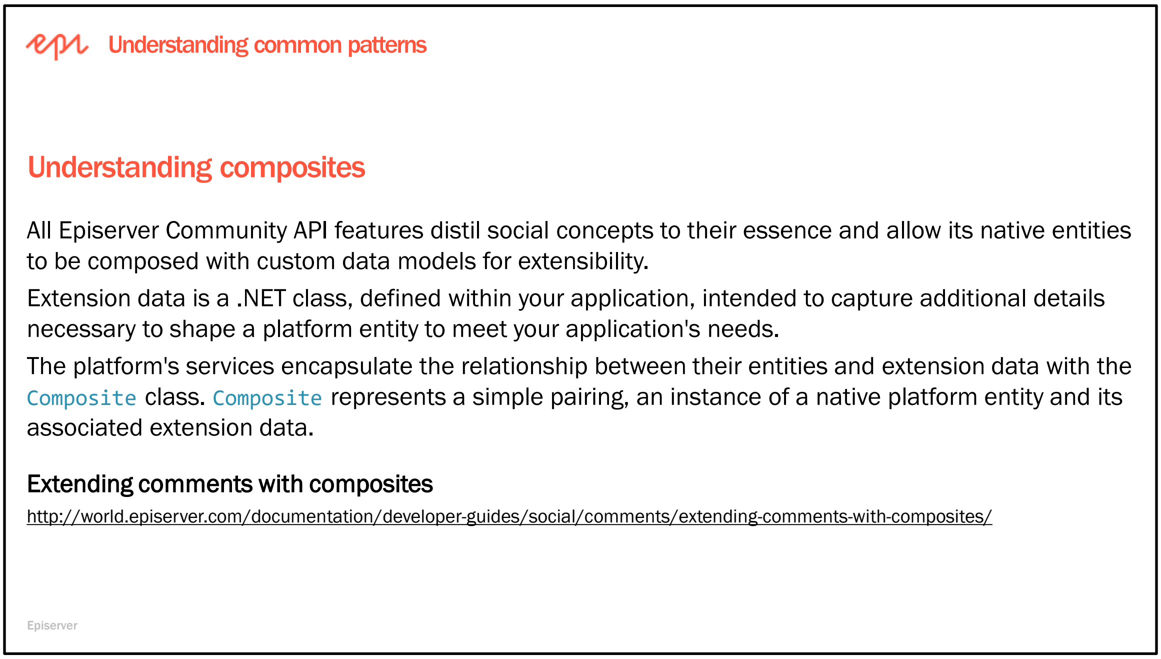
- The values are internally-generated and used to distinguish individual entities within the system.
- The classes are `CommentId`, `GroupId`, and so on.

Properties that end in `Reference` are for users or resources *outside* the Episerver Community API platform, including content in Episerver CMS and Commerce.

- The value is defined by the developer.
- A URI or similar namespace scheme provides an ideal template for a reference. The following is an example of a reference scheme that might be applied to Episerver Commerce content:

```
resource://episerver/commerce/{product-identifier}/{variant-identifier}
```

Episerver



Understanding common patterns

Understanding composites

All Episerver Community API features distil social concepts to their essence and allow its native entities to be composed with custom data models for extensibility.

Extension data is a .NET class, defined within your application, intended to capture additional details necessary to shape a platform entity to meet your application's needs.

The platform's services encapsulate the relationship between their entities and extension data with the `Composite` class. `Composite` represents a simple pairing, an instance of a native platform entity and its associated extension data.

Extending comments with composites

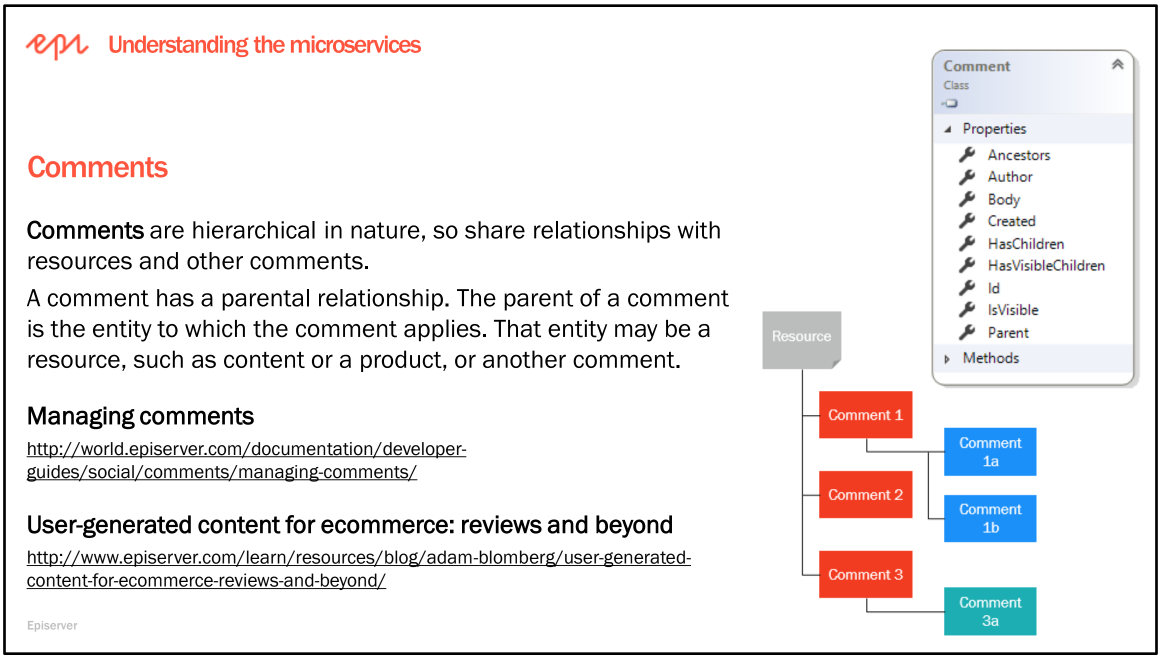
http://world.episerver.com/documentation/developer-guides/social/comments/extending-comments-with-composites/

Episerver

Episerver Community API criteria for retrieving result sets

These services accept criteria that dictate how to retrieve a result set. A class named Criteria<TFilter> encapsulates the specifications necessary to retrieve a collection of results from one of the platform services.

Criteria: http://world.episerver.com/documentation/developer-guides/social/social_platform-overview/discovering-the-platform/#criteria



Understanding the microservices

Comments

Comments are hierarchical in nature, so share relationships with resources and other comments.

A comment has a parental relationship. The parent of a comment is the entity to which the comment applies. That entity may be a resource, such as content or a product, or another comment.

Managing comments

http://world.episerver.com/documentation/developer-guides/social/comments/managing-comments/

User-generated content for ecommerce: reviews and beyond

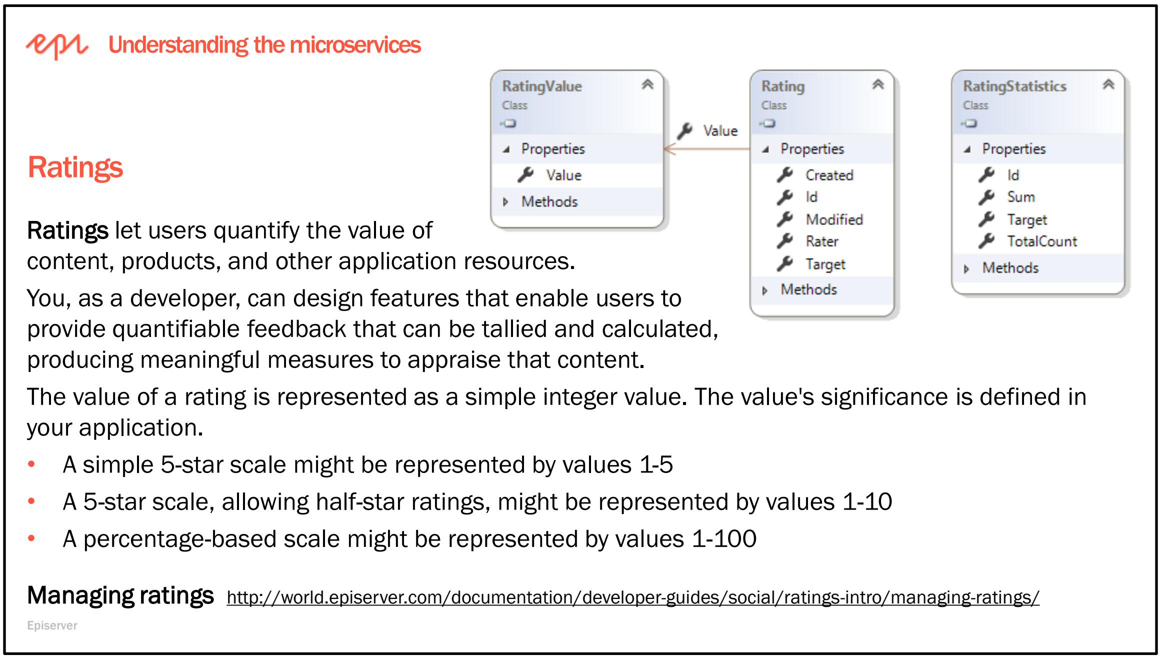
http://www.episerver.com/learn/resources/blog/adam-blomberg/user-generated-content-for-ecommerce-reviews-and-beyond/

Episerver

Comment
Class

Properties
- Ancestors
- Author
- Body
- Created
- HasChildren
- HasVisibleChildren
- Id
- IsVisible
- Parent

Methods



Understanding the microservices

Ratings

Ratings let users quantify the value of content, products, and other application resources.

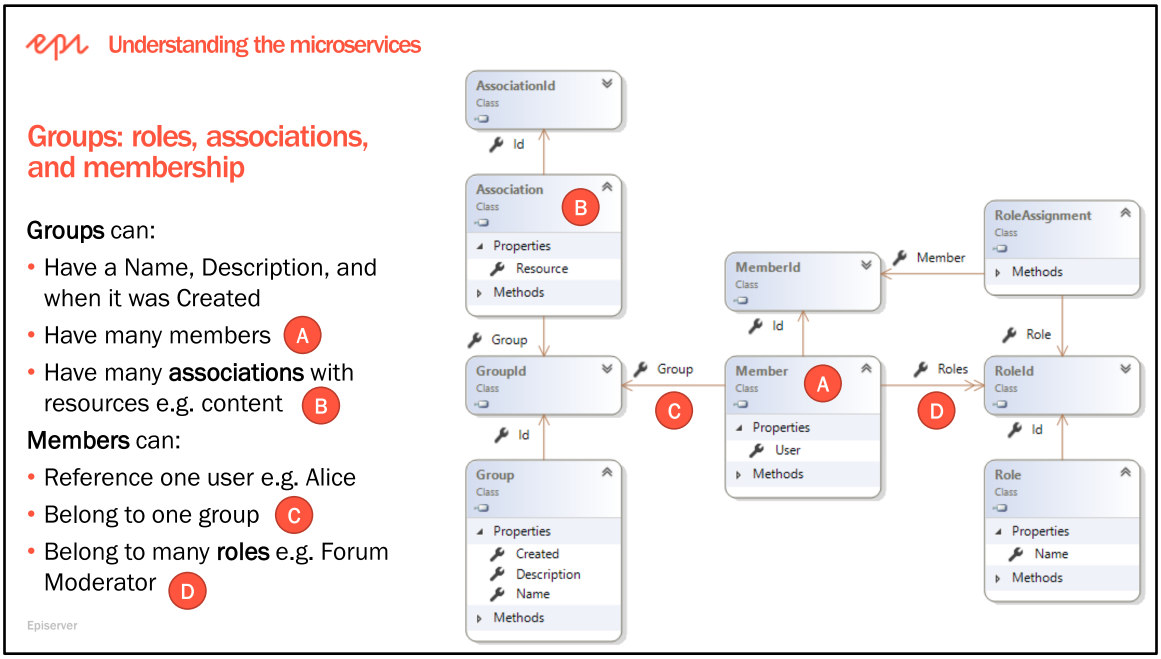
You, as a developer, can design features that enable users to provide quantifiable feedback that can be tallied and calculated, producing meaningful measures to appraise that content.

The value of a rating is represented as a simple integer value. The value's significance is defined in your application.

- A simple 5-star scale might be represented by values 1-5
- A 5-star scale, allowing half-star ratings, might be represented by values 1-10
- A percentage-based scale might be represented by values 1-100

Managing ratings http://world.episerver.com/documentation/developer-guides/social/ratings-intro/managing-ratings/

Episerver



Understanding the microservices

Groups: roles, associations, and membership

Groups can:

- Have a Name, Description, and when it was Created
- Have many members A
- Have many **associations** with resources e.g. content B

Members can:

- Reference one user e.g. Alice
- Belong to one group C
- Belong to many **roles** e.g. Forum Moderator D

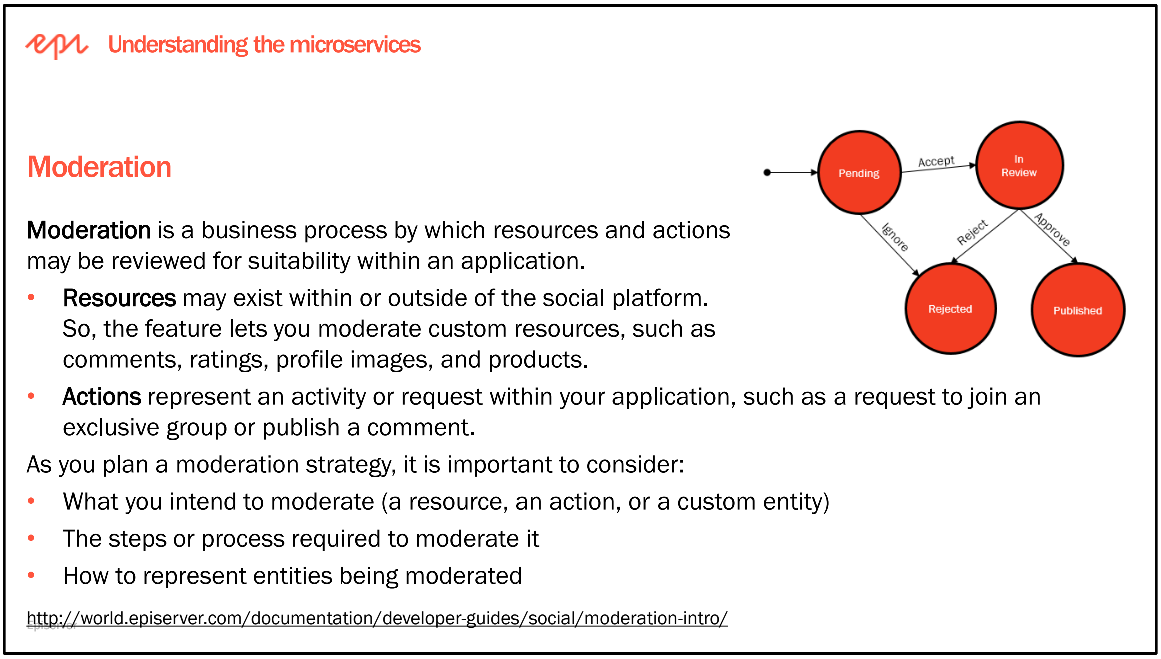
Episerver

Groups allow you to combine users and content to create digital communities.

- **Roles** provide a means of labelling or categorizing members within your digital community. They are defined, within your application, as you see fit. They may be assigned to members of a group or span multiple groups. Roles do not bestow any particular permission, status, or responsibility. This leaves your application free to apply meaning to roles as appropriate.
- You **associate** resources with a group by adding them as an association.
- Users are associated with a group by adding them as a member.

Managing groups, roles, associations, and membership

http://world.episerver.com/documentation/developer-guides/social/groups/managing-roles/
http://world.episerver.com/documentation/developer-guides/social/groups/groups-content-associations/
http://world.episerver.com/documentation/developer-guides/social/groups/groups-membership/



Understanding the microservices

Moderation

Pending
Accept
In Review
Ignore
Reject
Approve
Rejected
Published

Moderation is a business process by which resources and actions may be reviewed for suitability within an application.

- **Resources** may exist within or outside of the social platform. So, the feature lets you moderate custom resources, such as comments, ratings, profile images, and products.
- **Actions** represent an activity or request within your application, such as a request to join an exclusive group or publish a comment.

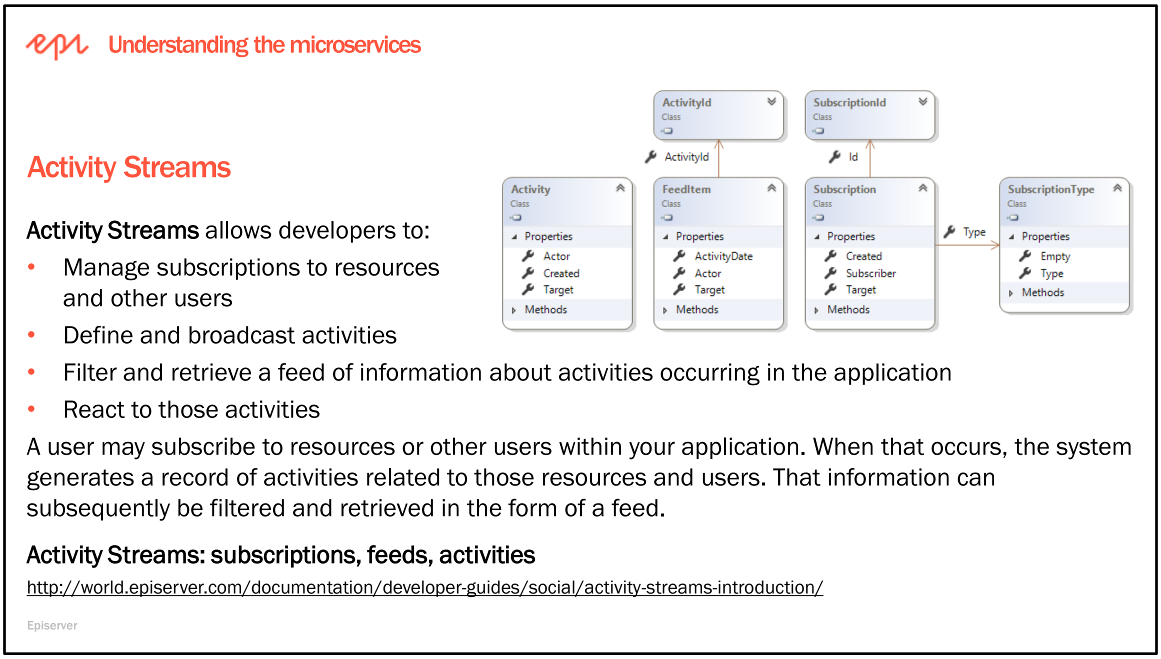
As you plan a moderation strategy, it is important to consider:

- What you intend to moderate (a resource, an action, or a custom entity)
- The steps or process required to moderate it
- How to represent entities being moderated

http://world.episerver.com/documentation/developer-guides/social/moderation-intro/

A workflow is comprised of:

- A set of states. For example: "Pending", "In Review", "Rejected", "Published".
- Actions. For example: "Accept", "Ignore", "Reject", "Publish".
- Transitions, the combination of two states (origin and destination) and an action, which causes the transition to occur. For example, an item's state is "Pending" (origin state), a reviewer accepts the request (action), changing its state to "In Review" (destination state).



Understanding the microservices

Activity Streams

Activity Streams allows developers to:

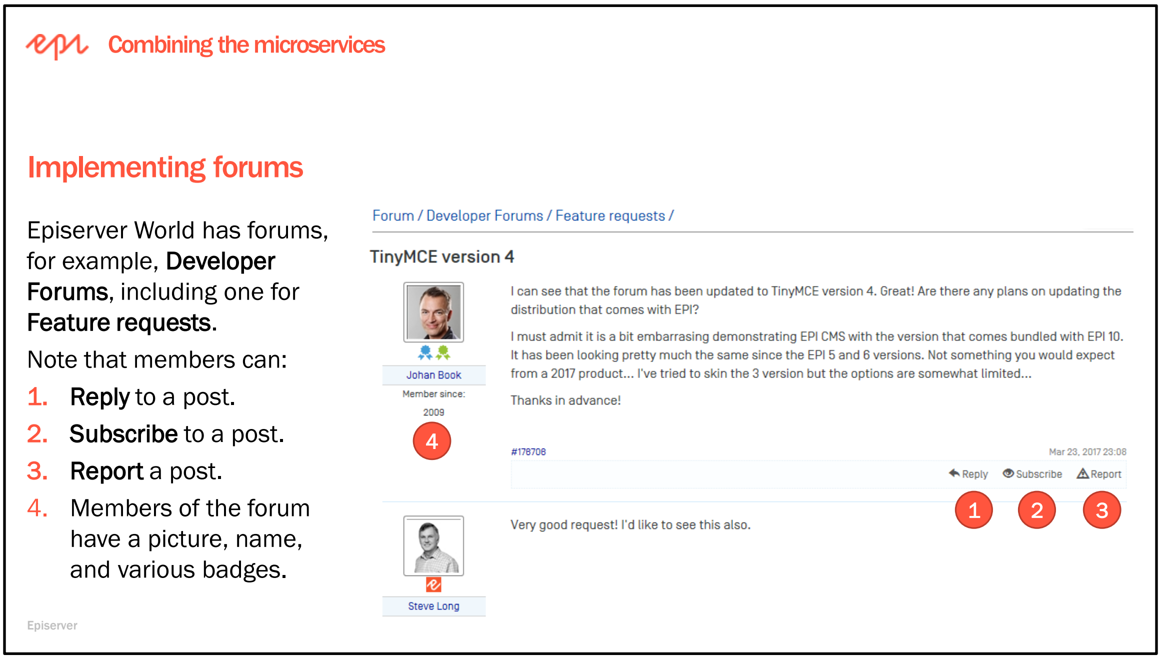
- Manage subscriptions to resources and other users
- Define and broadcast activities
- Filter and retrieve a feed of information about activities occurring in the application
- React to those activities

A user may subscribe to resources or other users within your application. When that occurs, the system generates a record of activities related to those resources and users. That information can subsequently be filtered and retrieved in the form of a feed.

Activity Streams: subscriptions, feeds, activities

http://world.episerver.com/documentation/developer-guides/social/activity-streams-introduction/

Episerver



Combining the microservices

Implementing forums

Episerver World has forums, for example, **Developer Forums**, including one for **Feature requests.**

Note that members can:

1. **Reply** to a post.
2. **Subscribe** to a post.
3. **Report** a post.
4. Members of the forum have a picture, name, and various badges.

Forum / Developer Forums / Feature requests /

TinyMCE version 4

I can see that the forum has been updated to TinyMCE version 4. Great! Are there any plans on updating the distribution that comes with EPI?

I must admit it is a bit embarrasing demonstrating EPI CMS with the version that comes bundled with EPI 10. It has been looking pretty much the same since the EPI 5 and 6 versions. Not something you would expect from a 2017 product... I've tried to skin the 3 version but the options are somewhat limited...

Thanks in advance!

Very good request! I'd like to see this also.

Episerver

To implement forum functionality on your website similar to Episerver's, you could combine all Episerver Community API's micro-services:

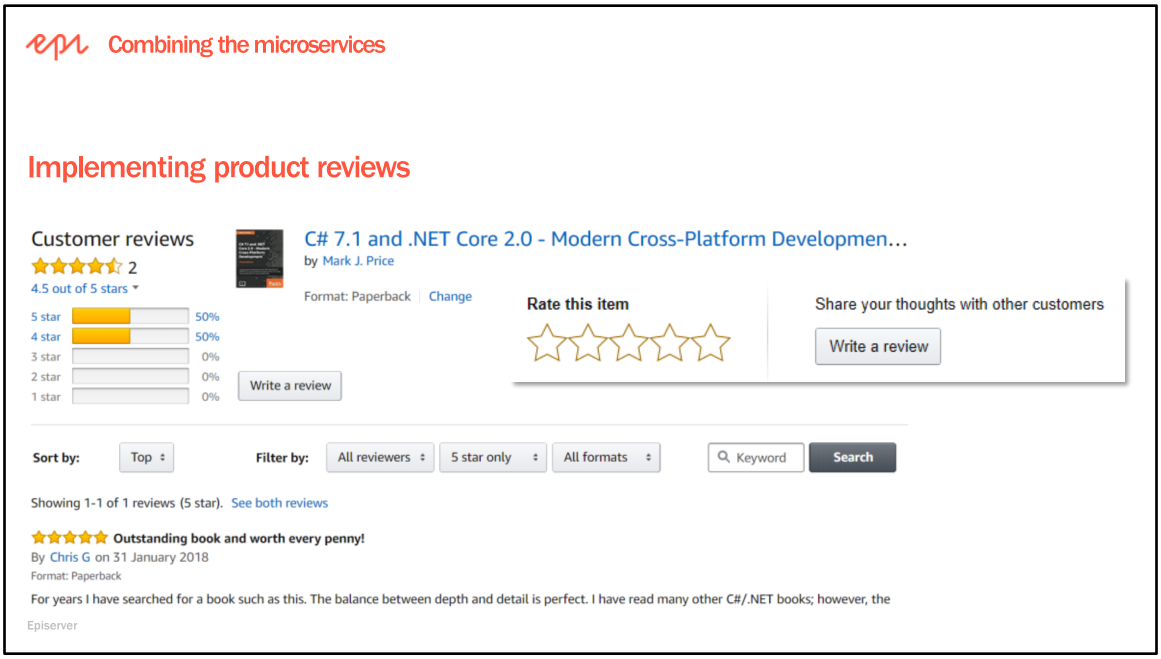
Comments: hierarchy of posts and replies.

Ratings: combine with post or reply to create a "report". If more than one member reports a post, perhaps it is temporarily hidden and flagged for forum administrator review.

Groups: use groups for Members and Moderators. Members could have extended data like badges for Episerver Certified Developers, and Episerver employees using Episerver Community API composites.

Moderation: use to determine membership of forums, and special badges to show.

Activity Streams: allow members to see posting activity so they can get answers to their questions ASAP.



Combining the microservices

Implementing product reviews

Customer reviews
★★★★☆ 2
4.5 out of 5 stars

5 star		50%
4 star		50%
3 star		0%
2 star		0%
1 star		0%

C# 7.1 and .NET Core 2.0 - Modern Cross-Platform Developmen…
by Mark J. Price
Format: Paperback | Change

Write a review

Rate this item ☆☆☆☆☆

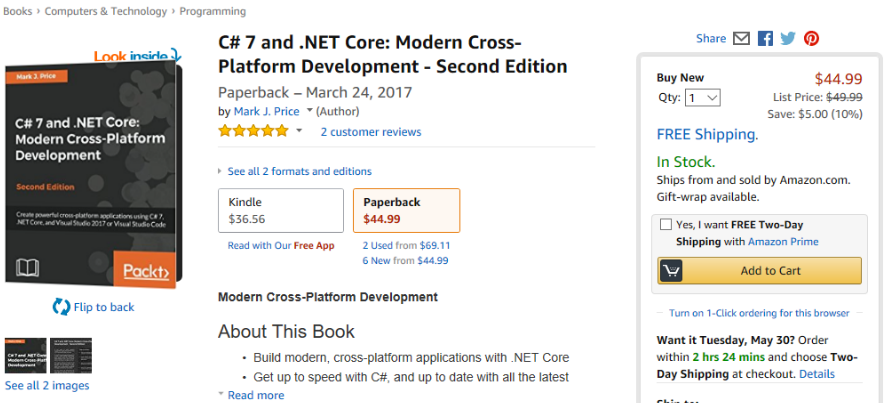
Share your thoughts with other customers
Write a review

Sort by: Top
Filter by: All reviewers | 5 star only | All formats | Keyword | Search

Showing 1-1 of 1 reviews (5 star). See both reviews

★★★★★ Outstanding book and worth every penny!
By Chris G on 31 January 2018
Format: Paperback
For years I have searched for a book such as this. The balance between depth and detail is perfect. I have read many other C#/.NET books; however, the

Episerver



Books › Computers & Technology › Programming

C# 7 and .NET Core: Modern Cross-Platform Development - Second Edition
Paperback – March 24, 2017
by Mark J. Price (Author)
★★★★★ 2 customer reviews

› See all 2 formats and editions

Kindle	Paperback
$36.56	$44.99
Read with Our Free App	2 Used from $69.11 6 New from $44.99

Modern Cross-Platform Development

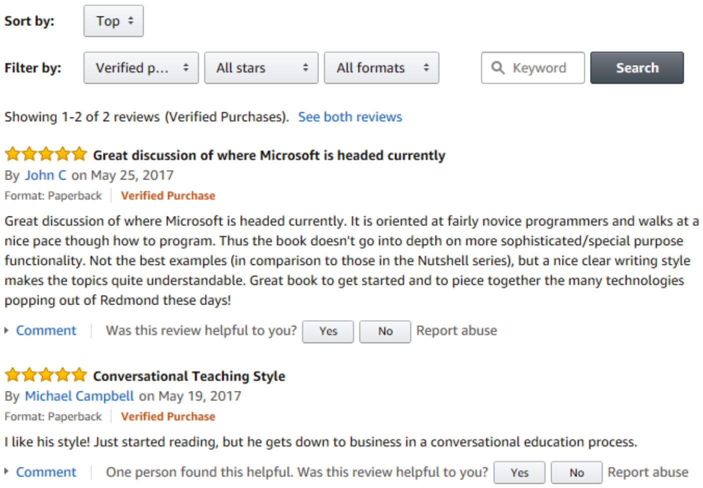
About This Book
- Build modern, cross-platform applications with .NET Core
- Get up to speed with C#, and up to date with all the latest

Read more

Look inside
Flip to back
See all 2 images

Share

Buy New $44.99
Qty: 1
List Price: $49.99
Save: $5.00 (10%)
FREE Shipping.
In Stock.
Ships from and sold by Amazon.com. Gift-wrap available.
□ Yes, I want FREE Two-Day Shipping with Amazon Prime
Add to Cart
Turn on 1-Click ordering for this browser
Want it Tuesday, May 30? Order within 2 hrs 24 mins and choose Two-Day Shipping at checkout. Details



Sort by: Top
Filter by: Verified p… | All stars | All formats | Keyword | Search

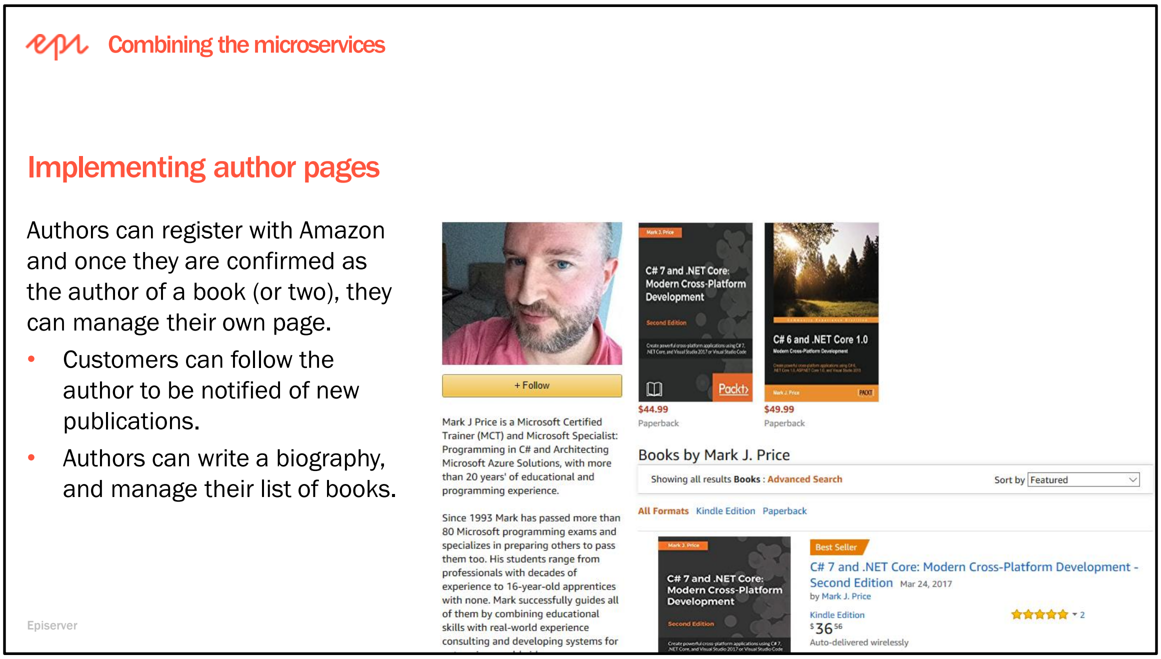
Showing 1-2 of 2 reviews (Verified Purchases). See both reviews

★★★★★ Great discussion of where Microsoft is headed currently
By John C on May 25, 2017
Format: Paperback Verified Purchase
Great discussion of where Microsoft is headed currently. It is oriented at fairly novice programmers and walks at a nice pace though how to program. Thus the book doesn't go into depth on more sophisticated/special purpose functionality. Not the best examples (in comparison to those in the Nutshell series), but a nice clear writing style makes the topics quite understandable. Great book to get started and to piece together the many technologies popping out of Redmond these days!
› Comment | Was this review helpful to you? Yes No Report abuse

★★★★★ Conversational Teaching Style
By Michael Campbell on May 19, 2017
Format: Paperback Verified Purchase
I like his style! Just started reading, but he gets down to business in a conversational education process.
› Comment | One person found this helpful. Was this review helpful to you? Yes No Report abuse

The review list is:
- Sortable, Filterable, Searchable

Each review has:
- Rating and Title
- Author and Date
- Verified purchase
- Comment (and ability to respond)
- Ability to rate the review as helpful and report abuse



Combining the microservices

Implementing author pages

Authors can register with Amazon and once they are confirmed as the author of a book (or two), they can manage their own page.

- Customers can follow the author to be notified of new publications.
- Authors can write a biography, and manage their list of books.

+ Follow

Mark J Price is a Microsoft Certified Trainer (MCT) and Microsoft Specialist: Programming in C# and Architecting Microsoft Azure Solutions, with more than 20 years' of educational and programming experience.

Since 1993 Mark has passed more than 80 Microsoft programming exams and specializes in preparing others to pass them too. His students range from professionals with decades of experience to 16-year-old apprentices with none. Mark successfully guides all of them by combining educational skills with real-world experience consulting and developing systems for

Mark J. Price
C# 7 and .NET Core: Modern Cross-Platform Development
Second Edition
Create powerful cross-platform applications using C# 7, .NET Core, and Visual Studio 2017 or Visual Studio Code
Packt>
C# 6 and .NET Core 1.0
Modern Cross-Platform Development
Mark J. Price
PACKT

$44.99
Paperback

$49.99
Paperback

Books by Mark J. Price

Showing all results **Books** : Advanced Search

Sort by Featured

All Formats Kindle Edition Paperback

Best Seller

C# 7 and .NET Core: Modern Cross-Platform Development - Second Edition Mar 24, 2017

by Mark J. Price

Kindle Edition

$36.56

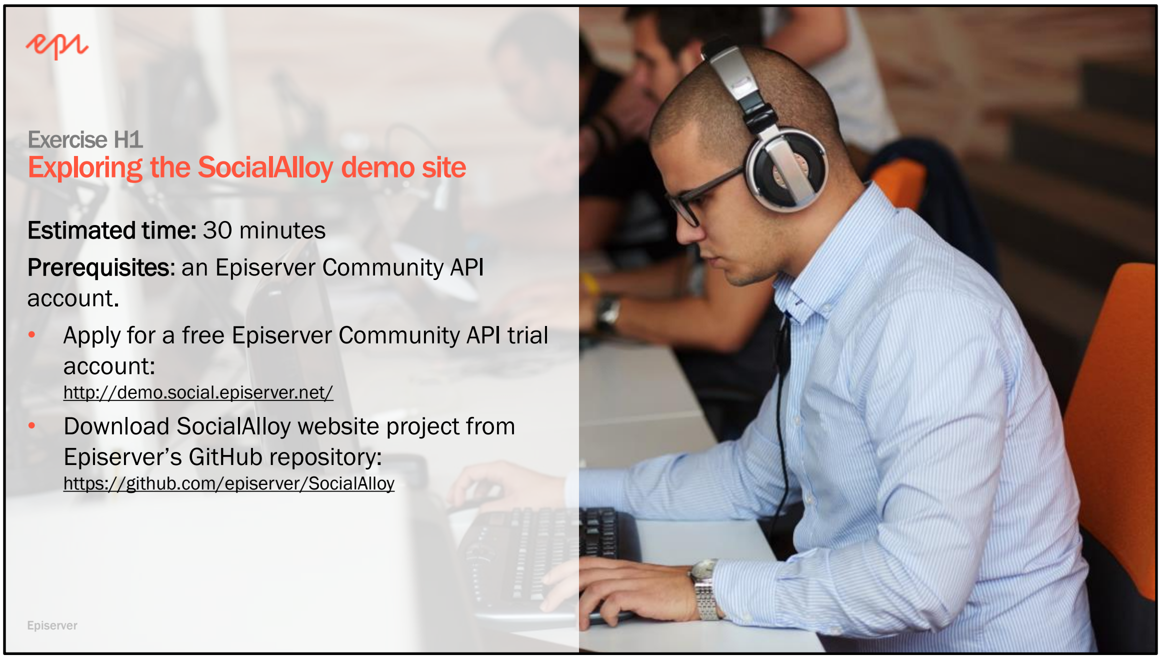
Auto-delivered wirelessly

★★★★★ 2

Episerver

To implement functionality on your website similar to Amazon's, you could combine all Episerver Community API's micro-services:

- **Comments: hierarchy of reviews and responses.**
- **Ratings: combine with comment to create a "review", or individual rating without review text; use built-in aggregation feature to show summary.**
- **Groups: use groups for VerifiedPurchaser and Author. Authors can have extended data like Biography using Episerver Community API composites.**
- **Moderation: use workflow to determine membership of VerifiedPurchaser and Author groups.**
- **Activity Streams: allow authors to see reviewing activity so they can respond ASAP, and allow customers to follow the author.**



Exercise H1

Exploring the SocialAlloy demo site

Estimated time: 30 minutes

Prerequisites: an Episerver Community API account.

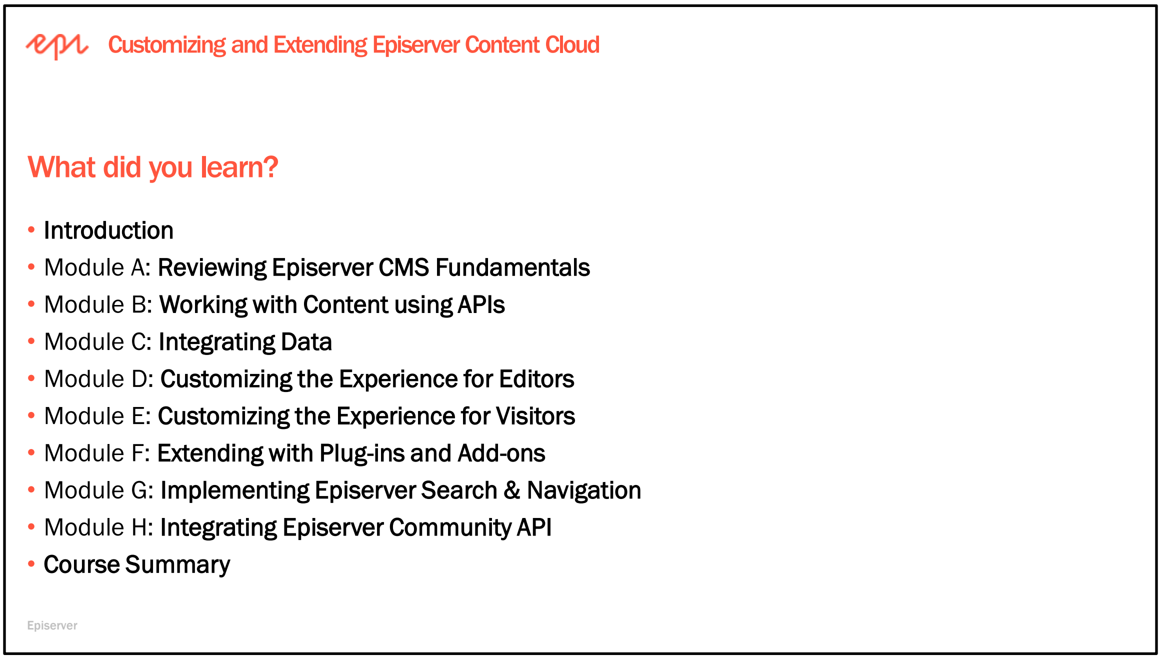
- Apply for a free Episerver Community API trial account: http://demo.social.episerver.net/
- Download SocialAlloy website project from Episerver's GitHub repository: https://github.com/episerver/SocialAlloy

Episerver

Customizing and Extending Episerver Content Cloud

Course Summary

Episerver



Customizing and Extending Episerver Content Cloud

What did you learn?

- **Introduction**
- Module A: **Reviewing Episerver CMS Fundamentals**
- Module B: **Working with Content using APIs**
- Module C: **Integrating Data**
- Module D: **Customizing the Experience for Editors**
- Module E: **Customizing the Experience for Visitors**
- Module F: **Extending with Plug-ins and Add-ons**
- Module G: **Implementing Episerver Search & Navigation**
- Module H: **Integrating Episerver Community API**
- **Course Summary**

Episerver

Module A: Reviewing Episerver CMS Fundamentals
In this module, you will review topics you should already know.

Module B: Working with Content using APIs
In this module, you will learn about some advanced APIs including working with Content Approvals, Projects, and Notifications.

Module C: Integrating Data
In this module, you will learn about various technologies and techniques for integrating non-content data, including gathering visitor data with Forms and integrating external data systems with partial routers and Service API.

Module D: Customizing the Experience for Editors
In this module, you will learn how to customize the editors experience when setting content properties.

Module E: Customizing the Experience for Visitors
In this module, you will learn how to take control of the visitors experience with custom rendering, personalization with visitor groups, and advanced customization of Episerver Search,.

Module F: Extending with Plug-ins and Add-ons
In this module, you will learn how to extend Episerver with custom plug-ins, gadgets, and add-ons.

Module G: Implementing Episerver Search & Navigation
In this module, you will learn how to integrate Episerver CMS with Episerver Find to implement advanced search capabilities.

Module H: Integrating Episerver Community API
In this module, you will learn how to integrate Episerver CMS with Episerver **Community API** to implement advanced features like comments, ratings, and managing groups.

Thank you!

Please complete the optional evaluation at the end of each eLearning course. We do read all comments and use them to improve our courses.

Episerver