

Episerver CMS Development Fundamentals Exercise Book May 2018

Product version: Update 214 Course version: 18.05



Course title: *Episerver CMS – Development Fundamentals* Course code: 170-3020 Course version: 18.05, 17th May 2018 Product Update 214, 14th May 2018

Episerver CMS Visual Studio Extension version: 11.3.0.359 Episerver CMS packages: EPiServer.CMS.Core 11.7.0, EPiServer.CMS.UI 11.4.4

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

۳

Episerver - update 214

Included packages: CMS Core 11.7.0, Commerce 12.2.0, Personalization Commerce 1.2.0, Google Analytics Commerce 2.2.0, Labs Language Manager 3.1.3, UGC 1.1.1, Lionbridge Connector 1.4.2.1100 May 14 2018

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

Copyright © 1996-2018 EPiServer AB. All rights reserved.

Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without expressed written permission of EPiServer AB. We assume no liability or responsibility for any errors or omissions in the content of this document.

EPiServer is a registered trademark of EPiServer AB.



Table of Contents

Module A – Getting Started with Episerver CMS	4
Exercise A1 – Setting up the AlloyDemo website	4
Exercise A2 – Reviewing and creating groups and users	13
Exercise A3 – Creating, editing, saving, and publishing content	24
Exercise A4 – Personalizing, approving, and A/B testing content	30
Exercise A5 – Localizing content	39
Exercise A6 – Resetting the Admin account	55
Exercise A7 – Identifying website features	57
Module B – Defining Content Types	58
Exercise B1 – Setting up the AlloyTraining website	58
Exercise B2 – Managing media assets	73
Exercise B3 – Implementing design patterns and conventions	81
Exercise B4 – Creating page types with a shared layout and navigation	94
Module C – Rendering Content Templates	106
Exercise C1 - Creating partial templates for product pages and image files for use in content areas	106
Exercise C2 – Creating a partial template for all pages	113
Exercise C3 – Enabling editors to apply tags manually using display options	118
Exercise C4 – Applying tags to content areas with code	120
Exercise C5 – Applying tags automatically using a display channel	122
Module D – Working with Blocks	127
Exercise D1 – Creating a controller-less block for editorial content	127
Exercise D2 – Creating a block with a controller for teaser content	130
Exercise D3 – Creating a preview renderer for partial pages and shared blocks	135
Exercise D4 – Moving properties to the basic info area	140
Exercise D5 – Using a block as a content property type	142
Module E – Navigating Content	144
Exercise E1 – Creating a page listing block	144
Exercise E2 – Creating a news landing page	149
Exercise E3 – Improving navigation menus	152
Exercise E4 – Creating a search page for visitors	155
Exercise E5 – Adding a search box to the top navigation menu	165
Module F – Working with Episerver Framework	167
Exercise F1 – Exporting and importing content	167
Exercise F2 – Implementing FAQs with content APIs	170
Exercise F3 – Listening for events and customizing services with initialization modules	175
Exercise F4 – Implementing scheduled jobs	179
Exercise F5 – Implementing soft and hard deletes	182
Exercise F6 – Learning from Episerver's assemblies	185
Module G – Optimizing, Securing, and Deploying	187
Exercise G1 – Controlling the caching of responses	187
Exercise G2 – Implementing logging	199
Exercise G3 – Securing an Episerver site	202
Summary of Attributes in Episerver	205



Episerver CMS - Development Fundamentals

Student prerequisites

You should have previous experience with Microsoft Visual Studio and ASP.NET MVC development.

Software requirements

To complete these exercises, you will need:

- Microsoft Visual Studio 2015 or 2017 with latest updates.
- Episerver CMS Visual Studio Extension version 11.3.0.359.
- **cmsdevfun_exercisefiles.zip** containing starter, solution, and support files for the exercises, as shown in the following screenshots:

📕 🛃 📜 🗧 cmsdevfun-e	exercisefiles			_		\times
File Home Share	View					~ ?
← → • ↑ 🖡 « 18.03	3 > cmsdevfun-exercisefi	es > v U	Search cmsde	evfun-exe	rcisefiles	Q
A Quick access	Name	Date modified	Туре	Size		
	👃 Assets	28/11/2017 16:26	File folder			
💑 Dropbox	🜛 Demos	07/12/2017 15:01	File folder			
\land OneDrive - Episen	👃 Module A	28/02/2018 16:17	File folder			
TH DC	👃 Module B	27/02/2018 14:26	File folder			
This PC	👃 Module C	21/11/2017 19:18	File folder			
🎝 3D Objects	👃 Module D	21/11/2017 19:18	File folder			
늘 Desktop	👃 Module E	08/01/2018 16:03	File folder			
📑 Documents	👃 Module F	19/01/2018 14:40	File folder			
📮 Downloads 🛛 🗸	臱 Module G	21/11/2017 19:18	File folder			
9 items						
					_	
📕 🗹 📕 🖛 B3				_		Х
File Home Share	View					~ ?
← → • ↑ 🖡 « Mod	ule B 🔉 B3 🔉	ن ~	Search B3			Q
▲ Quick access	Name	Date modified	Туре	Size		
	🜛 Business	28/02/2018 14:34	File folder			
💑 Dropbox	🕹 Controllers	21/11/2017 19:18	File folder			
\land OneDrive - Episen	🜛 Models	21/11/2017 19:18	File folder			
	👃 Resources	21/11/2017 19:18	File folder			
S This PC	👃 Static	21/11/2017 19:18	File folder			
🎝 3D Objects	👃 Views	21/11/2017 19:18	File folder			
늘 Desktop	SiteContentIcons.cs	22/06/2017 09:19	CS File		2 KB	
Documents	<i>iteTabNames.cs</i> 🔊	20/06/2017 08:53	CS File		1 KB	
📜 Downloads 🗸 🗸						
8 items						



Module A – Getting Started with Episerver CMS

Goal

The overall goal of the exercises in this module is to learn how to use the core features of Episerver CMS for editors and admins, and how to set up new Episerver websites. You will:

- 1. Set up an Alloy (MVC) sample website, update it to the latest version of Episerver CMS, and install some useful add-ons.
- 2. Review authentication and authorization good practice, create common groups and users, and set access rights.
- 3. Practice creating, editing, and publishing content.
- 4. Define content approval sequences, approve content, and perform A/B testing.
- 5. Localize content for English, Swedish, and Danish languages, localize choices in the styles drop-down menu in the TinyMCE editor, and localize names and descriptions of content types.
- 6. Implement functionality to reset the administrator account in case of a forgotten password.

Exercise A1 – Setting up the AlloyDemo website

In this exercise, you will set up an Alloy (MVC) site, update it to use the latest version of Episerver CMS, and you will install some add-ons that extend Episerver with extra features:

- Episerver Forms: http://webhelp.episerver.com/latest/addons/episerver-forms/episerver-forms.htm
- Episerver TinyMCE customization: <u>https://world.episerver.com/documentation/developer-guides/CMS/add-ons/customizing-the-tinymce-editor-v2/</u>
- Episerver A/B Testing: <u>http://webhelp.episerver.com/latest/cms-edit/ab-testing.htm</u>

If you are using an Episerver virtual machine, or you have already installed Visual Studio, Episerver CMS Visual Studio Extension, and set up the Episerver NuGet package source, then you can skip ahead to page 6, Creating an Alloy (MVC) Episerver website with sample content.

Minimum development system requirements

http://world.episerver.com/documentation/Items/System-Requirements/System-Requirements---EPiServer/

- Microsoft Windows 8, or later, or Windows Server 2012, or later.
- Microsoft Visual Studio 2015, or later.

These instructions will use our most recent recommended development stack: Microsoft Windows 10, Microsoft Visual Studio 2017 including SQL Server LocalDb, and Episerver CMS Visual Studio Extension. Older versions may look and behave slightly differently.

Installing Microsoft Visual Studio Community 2017

If you have already installed Microsoft Visual Studio 2015 or Microsoft Visual Studio 2017, or you are using a virtual machine provided by Episerver, then you can skip this section.

 Download and install Microsoft Visual Studio Community 2017 from the following link: <u>https://www.visualstudio.com/downloads/</u>



- 2. At a minimum, choose the workloads: **ASP.NET and web development**, **Azure development**, and **.NET Core cross-platform development**.
- 3. Add the individual components: Azure Storage AzCopy, Class Designer, Git for Windows, GitHub extension for Visual Studio, and PowerShell tools.

Installing the Episerver CMS Visual Studio Extension

If you have already installed the Episerver CMS Visual Studio Extension, or you are using a virtual machine provided by Episerver, then you can skip this section.

- 1. Start Microsoft Visual Studio.
- 2. Navigate to **Tools** | **Extensions and Updates...,** and in the section on the left, click **Online**, to show the **Visual Studio Marketplace**.
- 3. Press *Ctrl* + *E*, or click in the **Search** box, and then enter **Episerver**.
- 4. Select the Episerver CMS Visual Studio Extension, click Download.
- 5. Wait for the extension to download, and then click **Close**.
- 6. Close Visual Studio and wait for the VSIX Installer to start.
- 7. Click Modify, wait for it to complete installing, and then click Close.

Episerver CMS Visual Studio Extension 11.3.0.359 was released on 30th April 2018. After installation, we recommend that you manually update this extension by clearing the Automatically update this extension checkbox. To install an older version, download from the following link: <u>https://marketplace.visualstudio.com/items?itemName=EPiServer.EpiserverCMSVisualStudioExtension</u>

Configuring the Episerver NuGets package source

If you have already configured the Episerver NuGets package source, or you are using a virtual machine provided by Episerver, then you can skip this section.

- 1. Start Microsoft Visual Studio.
- 2. Navigate to Tools | NuGet Package Manager | Package Manager Settings.
- 3. In the Options dialog, in the list on the left, click Package Sources.

Microsoft SQL Server Express LocalDb should be included with Visual Studio, but you can also install it separately from the following link: <u>https://www.microsoft.com/en-gb/download/details.aspx?id=42299</u> Click **Download**, check the box for **LocalDB 64BIT\SqlLocalDB.msi**, and click **Next**.



4. If the Episerver NuGet feed doesn't exist as an available package source, as shown in the following screenshot, then click the **green plus** button to add it. The name can be anything, although we recommend using **Episerver NuGets**, and the path must be:

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

Options		?	\times
Search Options (Ctrl+E)	٩	Available package sources:	$\mathbf{\Psi}$
▷ Azure Data Lake	^	v nuget.org	
Container Tools		https://api.nuget.org/v3/index.json	
Cookiecutter		Episerver NuGet Feed	
Cross Platform		https://nuget.episerver.com/feed/packages.svc	
Database Tools			
▷ F# Tools			
▷ GitHub for Visual Studio			
▷ Node.js Tools			
NuGet Package Manager			
General			
Package Sources		Machine-wide package sources:	
PowerShell Tools		Microsoft Visual Studio Offline Packages	
▷ Python Tools		C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\	
▷ R Tools			
▷ SQL Server Tools			
▷ Test			
▷ Text Templating		Name: Episerver NuGet Feed	
▷ Web		Source: https://nuget.episerver.com/feed/packages.svc Updat	te
SWIE D.	~		
		OK Cance	el

Creating an Alloy (MVC) Episerver website with sample content

- 1. In Visual Studio, navigate to File | New | Project..., or press Ctrl + Shift + N.
- 2. In the left section, navigate to **Installed | Templates | Visual C# | Episerver**.
- 3. In the middle section, select Episerver Web Site project, and enter the following options:
 - Name: AlloyDemo

You MUST name your project AlloyDemo. The solution classes have been written assuming that project name and so all the namespaces will use it, e.g., AlloyDemo.Features.RegisterPersonas, and so on.

- Location: C:\Episerver (or some other folder).
- Solution name: **CMSTraining** (or something else unique).
- Create directory for solution: Yes
- Framework: .NET Framework 4.6.1 (or a later compatible version).

For compatibility with Episerver CMS 11 you must choose a minimum target of .NET Framework 4.6.1 because it requires .NET Standard 2.0.

<u>N</u> ame:	AlloyDemo	
Location:	C:\Episerver	<u>B</u> rowse
Solution name:	CMSTraining	\checkmark Create <u>directory</u> for solution
<u>F</u> ramework:	.NET Framework 4.6.1 -	Add to So <u>u</u> rce Control

4. Click OK.



5. Choose the Alloy (MVC) template, and in the Configure Search section, select Episerver Search, as shown in the following screenshot:

New Episerver Web Site - ,	AlloyDemo		×
Select a template:	Aloy (MVC)	Aloy (WebForms)	A sample web site for the fictional company Aloy. The templates are based on ASP.NET NVC. The project creation process will setup a new project and instal all required MUGet packages. Addabase is created in the App. Date folder using SQL Server Express LocaIDB. The Episerver editional user interface components are installed and placed under the URL "/EPServer". See nuget episerver.
Configure Search	Search solut content mod	ion configured for extended s lel. Sign up at http://find.epis	vearch capabilities based on your custom
Episerver Search	Search solut The search :	ion configured to use the bas service will be configured to r	ic built-in search with full-text support. un inside the web site.
			OK Cancel

6. Click **OK** and wait for the project to be created.

Updating the Episerver NuGet packages

Episerver CMS Visual Studio Extension 11.3.0.359 uses NuGet packages from 30th April 2018, i.e. EPiServer.CMS.Core 11.5.4. Recommended practice is to update to the latest NuGet packages for your chosen major version number that are usually released on a weekly schedule.

- 1. In Visual Studio, navigate to Tools | NuGet Package Manager | Package Manager Console.
- 2. In Package Manager Console, set these two options, as shown in the following screenshot:
 - a) Package source: All
 - b) Default project: AlloyDemo

Package Manager	Console				•	џ >	<
Pac <u>k</u> age source:	All	Φ.	Default project:	AlloyDemo	-		

- If you cannot see All as a Package source, then close Visual Studio, use File Explorer to open %appdata%\NuGet, rename NuGet.config to NuGet.backup, and restart Visual Studio. That should recreate NuGet.config, including the All option. You can redefine the Episerver NuGets feed following the instructions above, or copy and paste previous configuration from the backup, if necessary.
 - 3. Enter the following command to update all packages referenced by the project **AlloyDemo** using restrictions defined in packages.config:

Update-Package -ProjectName AlloyDemo -ToHighestMinor

Doing updates at the command line allows Episerver packages to be safely installed because it won't upgrade to a higher major version that would have breaking changes, and non-Episerver dependencies will be updated, as well as the EPiServer ones, for example, **Microsoft.Tpl.DataFlow**, but won't accidently install newer but incompatible packages. If you don't have the latest version of NuGet, you can follow instructions here: https://docs.microsoft.com/en-us/nuget/guides/install-nuget

You might need to restart Visual Studio a couple of times to update Entity Framework, as shown in the following screenshot:

Package Manager	Console				Ŧ	P	×
Package source:	All	• \$	Default project:	AlloyDemo 🔹 📔 🖉			
The package at 'C:\	Episerver\Training-2018-03-05	\packag	es\EntityFramewor	k.6.1.0' failed to uninstall. Restart Visual Studio to finish the process.	Re	st <u>a</u> r	t

Updating the Episerver database

1. Start the site by navigating to **Debug | Start Without Debugging** or press Ctrl + F5.



2. If you see an exception message about the **EPiServerDB** database, as shown in the following screenshot, then changes need to be made to the database schema:

Server Error in '/' Application.

The database schema for 'CMS' has not been updated to version '7052.0', current database version is '7050.0'. Update the database manually by running the cmdlet 'Update-EPiDatabase' in the package manager console or set updateDatabaseSchema="true" on episerver.framework configuration element.

3. Close the browser.

You can check if an update to a NuGet package would require an update to the database schema at the following link: <u>https://nuget.episerver.com/en/Comparedatabase/</u>

There are two ways to update the Episerver CMS database schema: manually with a console command, or automatically, with a Web.config setting, as explained in the error message. Recommended practice for deployments that you have control over is to perform migrations manually, especially if you have written custom SQL scripts to manipulate the CMS database schema, for example, to improve DDS performance. If you are deploying to DXC Service, we recommend using the Web.config setting.

- 4. In Visual Studio, navigate to Tools | NuGet Package Manager | Package Manager Console.
- 5. Make sure the **Default project** is **AlloyDemo**, as shown in the following screenshot, and at the prompt enter:

Update-EPiDatabase

Package Manager	Console			-	Ψ×
Package source:	All	- 🔯 Default project:	AlloyDemo	- Meine - Mei	
PM> Update-EPi	Database				
Processing C:\	Episerver\Training\packa	ages\EPiServer.CMS.C	Core.10.9.1\tools\epiupdates\sq	1\7.8.0.sql	
Processing C:\	Episerver\Training\packa	ages\EPiServer.CMS.C	ore.10.9.1\tools\epiupdates\sq	1\7.10.0.sql	1.1
Processing C:\	Episerver\Training\packa	ages\EPiServer.CMS.C	ore.10.9.1\tools\epiupdates\sq	1\7.11.0.sql	
Processing C+V	Eniserver\Training\nacka	ages\FPiServer CMS C	ore 10 9 1\tools\enjundates\so	1\7 12 0 sol	- T
100 % 👻 🔍					

If you get a warning about a missing packages folder, exit, and restart Visual Studio, and try again.

Testing the Episerver website and registering an administrator account

- 1. Start the site by navigating to **Debug** | **Start Without Debugging** or press *Ctrl* + *F*5.
- 2. You will be prompted to **Create Administrator Account**, as shown in the following screenshot, so enter the following details:
- Username: Admin or something else but make a note of it!
- Email: admin@alloy.com or some other e-mail address (it does not need to be real).
- Password: **Pa\$\$wOrd** or something else but make a note of it!

Create Admin	strator Acc X	Mark	D X
< → C ∆	localhost:58249/Register		☆ :
Create Ad	ninistrator Account		
Admin			
Email			
admin@alloy.co	m		
Password			
•••••			
Confirm passwor	d		
••••••			
Regis	ler		



3. If you use Chrome, it will prompt to save your **Username** and **Password**, so click **Save**, as shown in the following screenshot:

Do you w for this si	vant Google Chrome to s te?	ave your pa	ssword
Username	Admin		
Password			
		Save	Never

4. You should now see the Alloy sample site's Start page, as shown in the following screenshot:



5. Click the epi menu in the top-right corner, as shown in the following screenshot:



If you can't see the epi quick access menu, scroll down to the bottom of the page, and in the footer, click Log In, and log in as Admin.

6. At the top of the page, pull down the orange **Global** menu, and optionally click the pull-down menu a second time if you would like to pin it, as shown in the following screenshot:





- 7. Navigate to CMS | Admin | Config | Tool Settings | Plug-in Manager to confirm the version of Episerver CMS you are using, as shown in the following screenshot:
 - EPiServer and EPiServer.Cms.AspNet version 11.5.2.0: these are the CMS Core APIs.
 - EPiServer User Interface and EPiServer.Cms.Shell.UI version 11.4.3.0: these are the CMS user interface including Edit and Admin views.

lug-in Manager					
e list below displays comp	onents that have been registered as plug-ins in E	PiServer C	MS.		
Plug-ins Overview					
Name	Description	Version	Company	License	More Info
EPiServer.LinkAnalyzer	Link analyzer for Episerver CMS	11.5.2.0	Episerver AB	System internal	http://www.episerver.com
EPiServer User Interface	Supporting logic for the built-in web forms and user controls	11.4.3.0	EPiServer AB	System internal	http://www.episerver.com
EPiServer	Episerver Web Content Management System	11.5.2.0	Episerver AB	System internal	http://www.episerver.com
AlloyDemo		1.0.0.0	HP Inc.	Custom license	
EPiServer.Search.Cms	Episerver Web Content Management System	9.0.1.0	Episerver AB	System internal	http://www.episerver.com
EPiServer.Cms.AspNet	Episerver Web Content Management System	11.5.2.0	Episerver AB	System internal	http://www.episerver.com
EPiServer.Cms.Shell.UI	OnLine Center support for EPiServer CMS	11.4.3.0	EPiServer AB	System	http://www.episerver.com

8. Close the browser.

Windows 7 users

If you are using Windows 7, or your operating system has web sockets disabled, then in Edit view you will see warning messages about no support for real-time communication. To disable these warning messages, add the following element to Web.config:

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

In the Episerver CMS Advanced Development training course you learn how to create a plug-in for Admin view to allow this appSetting to be managed by administrators.

Installing some useful add-ons

- 1. Open the AlloyDemo project and view Solution Explorer.
- 2. Expand ~\modules_protected folder, as shown in the screenshot:
- Episerver products and add-ons are deployed as *shell modules*. In a new Alloy (MVC) project you will have four modules already installed: CMS is Episerver CMS user interface, EPiServer.Cms.TinyMce is the default rich text editor, EPiServer.Search.Cms is the (optional) integration with Episerver Search, and Shell is the shared UI including Dashboard and Global menu. The ZIP files contain the built-in functionality of the CMS, like Admin and Edit view. You can open CMS.zip and look at the contents, but don't change anything!

🖌 🛋 modules

- _protected
 _ S
 - CMS.zip
 - web.config
 - EPiServer.Cms.TinyMce
 EPiServer.Cms.TinyMce.zip
 web.config
 - EPiServer.Search.Cms
 IndexContent.aspx

P web.config

- P module.config
 ▲ Shell
 ▲ Shell.zip
- 3. Navigate to Tools | NuGet Package Manager | Package Manager Console.
- 4. Enter the following command to install Episerver Forms:
- In Package Manager Console, you can press the up arrow ↑ to repeat a previous command. You can then edit the command before pressing ENTER. Package Manager Console NuGet commands and package names are not case-sensitive.

Install-Package -ProjectName AlloyDemo EPiServer.Forms

5. Enter the following command to install A/B testing:



Install-Package -ProjectName AlloyDemo EPiServer.Marketing.Testing

- After installing add-ons, you may have to update some packages and the database schema. A/B Testing, the EPiServer.Marketing.Testing package, is an example of an add-on that requires this.
 - 6. Enter the following command to update dependent packages:

Update-Package -ProjectName AlloyDemo -ToHighestMinor

7. Enter the following command to update the database schema, as shown in the following screenshot:

Update-EPiDatabase

Package Manager Console 🔹 🕀 🗧	×
Package source: All 🔹 🕏 Default project: AlloyDemo 🔹 🞽 🗏	
Processing C:\Episerver\Training\packages\EPiserver.CMS.Core.10.10.0\tools\epiupdates\sql\10.8.0.sql Processing C:\Episerver\Training\packages\EPiserver.CMS.core.10.10.0\tools\epiupdates\sql\10.0.0.sql Processing C:\Episerver\Training\packages\EPiserver.CMS.core.10.10.0\tools\epiupdates\sql\10.10.0.1.sql Processing C:\Episerver\Training\packages\EPiserver.Narketing.Testing.2.3.1\tools\epiupdates\sql\10.0.1.sql Processing C:\Episerver\Training\packages\EPiserver.Narketing.Testing.2.3.1\tools\epiupdates\sql\10.0.1.sql Processing C:\Episerver\Training\packages\EPiserver.Narketing.Testing.2.3.1\tools\epiupdates\sql\10.0.3.sql Processing C:\Episerver\Training\Packages\EPiserVer.Narketing.Testing.2.3.1\tools\epiupdates\sql\10.0.	•
Processing C:\Episerver\Training\packages\EPiServer.Marketing.Testing.2.3.1\tools\epiupdates\sql\1.0.0.6.sql PM>	¥
100 % · ·	

Exploring Episerver Forms, TinyMCE, and A/B Testing

- 1. Start the website by navigating to Debug | Start Without Debugging or pressing Ctrl + F5.
- 2. Scroll down the Start page, and in the **Customer Zone**, click **Log in**, as shown in the following screenshot:

Products
Alloy Plan
Alloy Track
Alloy Meet

The Company About us News & Events Management News & Events Events Press Releases Customer Zone Reseller extranet Log in

- 3. Log in as Admin and click the epi menu to switch to Edit view for the current page.
- 4. Navigate to one of the product pages, like Alloy Plan, click the Main Body property, and note the rich text editing toolbar including Fullscreen mode, as shown in the following screenshot:



5. In Edit view, click Toggle assets pane, as shown in the following screenshot:





- 6. In the **Assets** pane, note the **Forms** tab, next to **Blocks** and **Media**, and the **Form Elements** and **Achived Tests** gadgets.
- 7. At the bottom of the **Assets** pane, in **Archived Tests**, click the **Settings** button, and select **Remove Gadget**, as shown in the screenshot:
- Some add-ons, like A/B Testing, automatically add gadgets, like Archived Tests that take up valuable space. It is safe to remove them, because they can easily be added back later if the logged in user needs them. Gadgets are specific to the logged in user, so you won't affect anyone else's gadgets.
 - 8. Close the browser.
- Congratulations! You have now created an Episerver CMS website, updated it to the latest version, and installed some useful add-ons.

Ŧ	¢				
v	Bloc	ks	Media	Forms	
Q	Sea	rch			
- 6	For	- All	Sites		≡-
_	Δ	llov	Meet		
				+ New Block	
+	≣				\$
∨ F	orm	Ele	ements		
v A	Archi	vec	l Tests		
En	d			Start	Owner
					\$
					Remove Gadget



Exercise A2 – Reviewing and creating groups and users

In this exercise, you will create users and groups with different access rights on the website.

Prerequisites: complete Exercise A1.

Reviewing authentication and authorization in an Alloy (MVC) site

- 1. In the AlloyDemo project, open ~/Web.config.
- 2. Navigate to **Edit** | **Find and Replace** | **Quick Find**, or press *Ctrl* + *F*, and enter **<au** to find the **<authentication>** element, as shown in the following screenshot:

Web.config	-p ()	×							-
51		<add name="ClientReso</td><td></td><td><au</td><td></td><td>× -</td><td>-</td><th>• • ×</th><td>÷</td></tr><tr><td>52</td><td>-</td><td></outputCacheSettings></td><td>A</td><td>а 🖽</td><td>_*</td><td>Current Document</td><td></td><th>-</th><td>Â</td></tr><tr><td>54</td><td>-</td><td></caching></td><td>-</td><td>_</td><td>_</td><td></td><td>-</td><th></th><td>•</td></tr><tr><td>55 🗄</td><td>-</td><td><authentication mode=" none"=""></add>							
56		<pre></pre>	in"	logi	nUrl	="Util/login.aspx	" t	imeou	1
57									
58 E		<profile defaultprovider="Def</td><td>au</td><td>ltPro</td><td>file</td><td>Provider"></profile>							

3. Note the mode and loginUrl attributes.

Alloy (MVC) project template sets authentication mode to None in the configuration file. This does not mean the Alloy site does not authenticate! For historical reasons, you must set authentication mode to None to enable federated authentication systems like ASP.NET Identity.

- Find the <membership> and <roleManager> elements, and note they are cleared because Alloy (MVC) does not use ASP.NET Membership.
- 5. Open ~/Startup.cs. and review the Configuration method, as partially shown in the following code:

```
// Add CMS integration for ASP.NET Identity
app.AddCmsAspNetIdentity<ApplicationUser>();
```

```
// Remove to block registration of administrators
app.UseAdministratorRegistrationPage(
    () => HttpContext.Current.Request.IsLocal);
```

// Use cookie authentication

app.UseCookieAuthentication(new CookieAuthenticationOptions

Alloy (MVC) project template uses ASP.NET Identity with accounts stored in Episerver CMS database to authenticate users and authorize roles and uses cookies for re-authentication. It has a custom extension method named UseAdministratorRegistrationPage that is only displayed to the visitor if, (1) the browser is executing on the web server i.e. it is a local request, and (2) there are no users in the CMS database.

- 6. In Solution Explorer, click Show All Files.
- 7. Expand the ~\App_Data folder, and note the **blobs** and **Index** folders, and the SQL database file, as shown in the following screenshot:
- 🔺 🛋 App_Data
 - blobs
 - ▶ 🛄 Index
 - DefaultSiteContent.episerverdata
 - EPiServerDB_144b9042.mdf
 - EPiServerErrors.log
 - GeoLiteCity.dat
- 8. Double-click EPiServerDB_{GUID}.mdf to open a database connection to it.



9. In Server Explorer, expand Tables, and note the tables that begin with AspNet, as shown in the following screenshot:



10. Right-click **AspNetRoles**, and click **Show Table Data**, and note the **WebAdmins** role has been created for you, as shown in the following screenshot:



- 11. Right-click **AspNetUsers**, and click **Show Table Data**, and note the **Admin** user (or whatever you entered on the register page) has been created for you, and that it has columns for the following to implement best practice for security: **IsApproved**, **LastLoginDate**, **Email**, **PasswordHash** (the original password is not stored), **AccessFailedCount**, **UserName**.
- 12. Close both tables.
- 13. In Server Explorer, right-click EPiServerDB (AlloyDemo), and click Close Connection.

Reviewing virtual roles and changing the default mapping for CmsEditors

You will now follow good practice by reviewing and modifying the configuration of the virtual roles that give access to the working areas of Episerver CMS. Most projects should not leave the default configuration asis so you will make some changes to learn some common modifications that you might make.

- 1. In the AlloyDemo project, open ~\Web.config.
- 2. Find the <virtualRoles> element, as shown in the following configuration:

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



type="EPiServer.Security.CreatorRole, EPiServer" />

- In this default configuration, users in the Windows group named Administrators, or users in the database-stored role named WebAdmins, will become members of the virtual role named CmsAdmins, which gives access to the Global menu and Admin view, but not necessarily content. Users in the database-stored role named WebEditors, will become members of the virtual role named CmsEditors, which gives access to Edit and Reports in the Global menu.
 - 3. Find the existing entry for **CmsAdmins**, and modify it so that a stored role named **AccessToAdminView** as well as **WebAdmins** and **Administrators** maps to **CmsAdmins**, as shown <u>underlined</u> in the following markup:

```
<add name="CmsAdmins"
type="EPiServer.Security.MappedRole, EPiServer.Framework"
roles="WebAdmins, Administrators, <u>AccessToAdminView</u>" mode="Any" />
```

You will create a stored role aka group named AccessToAdminView later in this exercise.

4. Find the existing entry for **CmsEditors**, and modify it so that a stored role named **AccessToEditView** maps to **CmsEditors** instead of **WebEditors**, as shown <u>underlined</u> in the following markup:

```
<add name="CmsEditors"
type="EPiServer.Security.MappedRole, EPiServer.Framework"
roles="<u>AccessToEditView</u>" mode="Any" />
```

You will create a stored role aka group named AccessToEditView later in this exercise.

 Add an entry at the bottom of the list of virtual role providers, that maps members of a stored role named Personalizers to the virtual role named VisitorGroupAdmins, as shown in the following markup:

```
<add name="VisitorGroupAdmins"
type="EPiServer.Security.MappedRole, EPiServer.Framework"
roles="Personalizers" mode="Any" />
```

You will create a stored role aka group named **Personalizers** later in this exercise. Any member of this group will have access to the **Visitor Groups** administration user interface.

6. Add an entry at the bottom of the list of virtual role providers, that maps members of a stored role named **Developers** to the virtual role named **EPiBetaUsers**, as shown in the following markup:

```
<add name="EPiBetaUsers"
type="EPiServer.Security.MappedRole, EPiServer.Framework"
roles="Developers" mode="Any" />
```

You will create a stored role aka group named **Developers** later in this exercise. Any member of this group will have access to any beta features of the user interface.

7. Find the <location path="EPiServer"> element, and note the <authorization> element allows members of WebEditors, WebAdmins, or Adminstrators to access the EPiServer path, as shown in the following markup:

```
<le><location path="EPiServer">
    <system.web>
    ...
    <authorization>
        <allow roles="WebEditors, WebAdmins, Administrators" />
        <deny users="*" />
        </authorization>
```



8. Replace the three <allow> roles with **CmsEditors** and **CmsAdmins**, as shown in the following markup:

```
<allow roles="CmsEditors, CmsAdmins" />
```

 Find the <location path="EPiServer/CMS/admin"> element, and note the <authorization> element allows members of WebAdmins or Adminstrators to access the EPiServer/CMS/admin path, as shown in the following markup:

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

10. Replace the two <allow> roles with CmsAdmins, as shown in the following markup:

```
<allow roles="CmsAdmins" />
```

11. Find the <location path="Views/Plugins"> element, and note the <authorization> element allows members of WebEditors, WebAdmins, or Adminstrators to access the Views/Plugins path, as shown in the following markup:

```
<lr><location path="Views/Plugins">
<system.web>
<authorization>
<allow roles="WebAdmins, WebEditors, Administrators" />
<deny users="*" />
</authorization>
```

12. Replace the three entries with CmsEditors and CmsAdmins, as shown in the following markup:

```
<allow roles="CmsEditors, CmsAdmins" />
```

You have now removed any dependency on the existance of stored roles or groups named WebAdmins, Administrators, and WebEditors. Instead, any member of the stored role or group named AccessToEditView will have access to the Edit view, and any member of a stored role or group that maps to CmsAdmins will have access to Admin view.

Reviewing the current groups and users

- 1. Start the AlloyDemo website, and log in as Admin.
- 2. Navigate to CMS | Admin | Admin | Access Rights | Administer Groups, as shown in the following screenshot:

Dashboard CMS Add-ons	epr 🖁)? 💄 Admin 🔍
Edit Admin Reports Visitor Groups		
Admin Config Content Type	Administer Groups	?
▼ Access Rights		
Set Access Rights	Delate an edd arrange was die ander ander allekte	
Administer Groups	Delete of add groups used to assign access rights.	
Search User/Group	🖨 Add	
Create User		
	Group Provider	Delete
Scheduled Jobs	2 WebAdmins EPi_AspNetIdentityRoleProvider	34
Marketing Test Monitor	-	<u>~</u>
Link Validation	1	



3. Click WebAdmins, to show the members of that group, as shown in the following screenshot:

elete o	r add groups used to assign access rights.		
			🔀 Car
	Group	Provider	Delete
-	WebAdmins	EPi_AspNetIdentityRoleProvider	*
1			
sers of	group "WebAdmins"	Devide	Delete
	wame	Provider	Delete

- In a newly created Alloy (MVC) website, **WebAdmins** has a single member, that is the user that you registered when the site is first started.
- 4. Navigate to CMS | Admin | Admin | Access Rights | Set Access Rights, and note that by default the Start page (and all other pages) inherit their access rights from their parent item, as shown in the following screenshot:

Set Access	Righ	ts for	"Start'						?
Restore access righ all rights on all item	nts in EP s.	iServer C	MS for iter	ms that yo	ou have, f	or example, o	completely rem	oved access to	o. You can chang
Root Start Alloy Plar Alloy Trat Alloy Trat Alloy Trat Alloy Mee Campaig Search Recycle Bin For Al Sites Customer Z	n sk uy ns s								
🍇 Add Users/Gro	oups								
Administrators	Read @	Create 🕜	Change 🕜	Delete 🕜	Publish @	Administer			
Everyone	4								
3 WebAdmins	4	4	1	4	4	1			
Inherit settings fr	om pare	ent item							

5. Click **Root**, and note that it has entries for **Administrators** (usually the name of a Windows-stored group), **Everyone** (a virtual role) and **WebAdmins** (usually the name of an SQL-stored group in the CMS database), as shown in the following screenshot:

ights on all items.	na m EP	IServer C	MS for iter	ns that yo	ou have, to	or example, c	ompletely remo	ived access to	o. You can change a
Root									
Recycle Bin									
For All Sites									
Customer 2	one								
an Add Osers/Gh	Jups								
		0	Change	Delete	Publish	Administer			
	Read	Create	_						
强 Administrators	Read	Create	-	\$	1	1			
Administrators	Read				Image: A start of the start				

Everyone represents anonymous visitors, so by default they only have Read access rights to all content.



Creating some new groups and a user using Admin view

- 1. Navigate to CMS | Admin | Admin | Access Rights | Administer Groups.
- 2. Click + Add.
- 3. Enter the name AccessToEditView and click Save.

Remember that AccessToEditView is mapped to CmsEditors which gives access to the Edit view and Reports view working areas.

- 4. Add a few more groups:
 - a. AccessToAdminView: this is mapped to CmsAdmins which gives access to all working areas including Admin view.
 - b. **ContentCreators**: members of this group will be able to create, change, and publish content throughout the site, except in the **News & Events** section.
 - c. **NewsEditors:** members of this group will be the only editors who can create, change, and publish pages in the **News & Events** section.
 - d. Marketers: members of this group will be the only editors who can create Product pages.
 - e. Developers: members of this group can access beta features.
- 5. Navigate to CMS | Admin | Admin | Access Rights | Create User, and fill in the following information, as shown in the following screenshot:
 - a. Username: Alice
 - b. Password: Pa\$\$wOrd
 - c. E-mail address: alice@alloy.com
 - d. Active: selected
 - e. Member of: AccessToAdminView

New User		(?)
User Information	Display Options	
Username Password Confirm password E-mail address	Alice alice@alloy.com ✓ Active → Active	
Provider Created date Last login date Description		
Not member of AccessToEditView CLevelExecs ContentCreators Developers Lawyers Marketers	AccessToAdminView	

6. Click Save.

Assigning access rights using Admin view

- 1. Navigate to CMS | Admin | Admin | Access Rights | Set Access Rights.
- 2. In the content tree, select Root.



3. Click Add Users/Groups, as shown in the following screenshot:



4. Search for **Groups**, add **ContentCreators**, **Marketers**, and **CmsAdmins**, and click **OK**, as shown in the following screenshot:



It is good practice to assign access rights to **CmsAdmins** instead of **WebAdmins** and **Administrators** because it uses the mapping in configuration and therefore provides flexibility.

5. Modify the following roles and their access rights, as shown in the following screenshot:

Set Access F	Rights	s for "l	Root"										?
Restore access rights items.	in EPiS	erver CMS	6 for items	that you	have, for	example, con	npletely re	emoved	access t	o. You ci	an chang	e all right	s on all
Start Recycle Bin For All Sites Customer Zor	ne												
ୡ Add Users/Grou	ips												
Administrators Everyone WebAdmins ContentCreators Amrketers CrnsAdmins Inherit settings fror Apply settings for	Read	Create	Change	Delete	Publish	Administer							
													📙 Save

- a. Administrators: clear all access rights.
- b. WebAdmins: clear all access rights.
- c. CmsAdmins: set all access rights.



- d. ContentCreators: set Read, Create, Change, Delete access rights.
- e. Marketers: set Create and Publish access rights.

U Clearing all access rights will remove the access control entry from the list when you click Save.

Five of the six access rights (all except Create), apply to the content item that is selected, for example, Root in the previous screenshot. Applying Create access right to Root does not mean you can create the root. It means you can create children underneath Root.

- 6. Click Save.
- 7. In the Set Access Rights content tree, select Recycle Bin.

U By default, only members of the Windows group **Administrators** can work with the Recycle Bin!

- 8. Click Add Users/Groups.
- 9. Search for Groups, add the CmsAdmins group, and click OK.
- 10. Clear all access rights from **Administrators**, and assign all access rights to **CmsAdmins**, as shown in the following screenshot:

Set Access	Righ	ts for	"Recy	cle Bi	n"		?
Restore access righ items.	nts in EP	iServer Cl	MS for iten	ns that yo	u have, fo	example, completely removed access to. You	can change all rights on all
 Root Start Recycle Bir For All Sites Customer Z 	s Cone						
ୡ Add Users/Gr	oups						
Administrators	Read	Create	Change	Delete	Publish	Administer	
Everyone	v						
CmsAdmins	•	∢	•	•	•		
 Inherit settings fr Apply settings fo 	rom pare or all subi	nt item items					
							🛃 Save

11. Click Save.

Now, any user who is a member of the virtual role CmsAdmins can manage the Recycle Bin. Currently, that means any user who is a member of the stored roles named AccessToAdminView, Administrators, or WebAdmins.

- 12. In the Set Access Rights content tree, expand Start, expand About us, and select News & Events.
- 13. Clear the Inherit settings from parent item check box.
- 14. Click Add Users/Groups.
- 15. Search for Groups, add the NewsEditors group, and click OK.



16. Clear all access rights from **ContentCreators** and **Marketers**, and assign all access rights to **NewsEditors**, as shown in the following screenshot:

Add Users/Groups Rad Creat Change Delete Publish Administer Conscientings Image: Construction of the state of the s	About us Conset Conset Conset Contact Con	Events Releases ment us a reseller e									
CrosAdmins ContentCreators Co	🖧 Add Users/Grou	Pead	Create	Change	Delete	Publich	Administer				
ContentCreators Image: ContentCreators Everyone Image: ContentCreators Marketers Image: ContentCreators NewsEditors Image: ContentCreators Inherit settings from parent item Apply settings for all subitems	CmsAdmins	 Intelligence 					∠ Contractor				
Everyone Imarketers Marketers Imarketers NewsEditors Imarket Inherit settings from parent item Apply settings for all subitems	ContentCreators										
Marketers Image: Constraint of the second secon	Everyone	1									
NewsEditors III I I I I I I I I I I I I I I I I I	Marketers										
Inherit settings from parent item Apply settings for all subitems	NewsEditors		1	1	1	1					
] Inherit settings from] Apply settings for a	n parent all subiter	item ms								

17. Click Save.

U If you were to open EPiServerDB and the table named tblContentAccess, you would see the following:

- 🖒 🔽 To	Max Rows:	1000	- 11
fkContentID	Name	IsRole	AccessMask
1	Administrators	1	0
1	CmsAdmins	1	63
1	ContentCreators	1	15
1	Everyone	1	1
1	Marketers	1	18
2	Administrators	1	0
2	CmsAdmins	1	63
2	Everyone	1	1
11	CmsAdmins	1	63
11	Everyone	1	1
11	Marketers	1	0
11	NewsEditors	1	63

fkContentID:	1 = RootPag	ge, 2 = Rec	ycle Bin,	11 =	News &	Events
--------------	-------------	-------------	-----------	------	--------	--------

18. In Admin view, click **Content Type**, click **[Products] Product**, and then click **Settings**, as shown in the following screenshot:





19. At the bottom of the **Information** tab, click **Add Users/Groups**, search and add the **Marketers** group, clear the check box for **Everyone**, and click **Save**, as shown in the following screenshot:

Access level			
	Create		
Everyone			
A Marketers			
🖧 Add Users	Groups		
		📙 Save 样 Revert to Default	🔀 Cancel

20. In Admin view, click **Config**, click **Permissions for Functions**, and then next to **Detailed error messages for troubleshooting**, click **Edit**, as shown in the following screenshot:

Admin Config Content Type	Permissions for Eunctions
System Configuration	
System Settings	Give users/groups access to specific functions in EPiServer
Manage Websites	
Manage Website Languages	Functions in CMS
Edit Categories	Allow users to move data/pages between page providers 🥜 Ed
Edit Frames	Detailed error messages for troubleshooting
Edit Tabs	2 CO
Property Configuration	Allow the user to act as a web service user 27 Ed
Edit Custom Property Types	

21. Click Add Users/Groups, search and add the CmsAdmins and Developers groups, clear the check box for Administrators, and click Save, as shown in the following screenshot:

Permissions for Functions				
Detailed error messages for troubleshooting				
3 Developers	2			
3 CmsAdmins	2			
& Add Users/Groups	Save 🕄 Cancel			

22. Close the browser.

Automating the management of groups, users, and access rights

Now that you have seen how to manage authorization by manually using Admin view, you will write some code to automate the process to create some more groups and users and set their access rights.

- 1. If you haven't done so already, extract the folders and files in cmsdevfun_exercisefiles.zip.
- Drag and drop the \cmsdevfun-exercisefiles\Module A\A2\Features\ folder into the AlloyDemo project.
- 3. Expand the **Features** folder and review the files included, as shown in the screenshot:
 - a. **RegisterPersonas.cs**: a static class to enable the feature.
 - b. **RegisterPersonas.cshtml**: a Razor file for the user interface of the feature.
 - c. **RegisterPersonasController.cs**: a controller that performs the work of the feature.
- Controllers
 Features
 RegisterPersonas
 C* RegisterPersonas.cstml
 C* RegisterPersonasController.cs
 Startup.cs-changes.txt
 Web.config
 Helpers
- 4. In **AlloyDemo**, in the root of the website, open **Startup.cs**, and import the AlloyDemo.Features.RegisterPersonas namespace.
- 5. Add a statement to the Configuration method to call the UseRegisterPersonas extension method that enables registering of personas, as shown in the following code:

app.UseRegisterPersonas(() => HttpContext.Current.Request.IsLocal);



- 6. Start the website, and note that a page appears offering to register some personas for you, as shown in the screenshot:
- If all the users already exist, then this page will not appear. If necessary, you can use Admin view to delete one of the users to force it to appear.
 - 7. Click **Yes, do it!**, and note the success message.
 - Log in as Admin or Alice and navigate to CMS | Admin | Admin | Search User/Group.

Register Personas

User	Groups
Alice	AccessToAdminView
Dana	AccessToAdminView Developers
Eve	AccessToEditView ContentCreators
Nick	AccessToEditView NewsEditors
Michelle	AccessToEditView Marketers Personalizers
Carlos	AccessToEditView CLevelExecs
Larry	AccessToEditView
Yes, do it!	

that you want to create the following users make them members of these groups, and assign access

Log me in to Episerver CMS.

9. Click Search to list all the users, as shown in the following screenshot:

Sea	rch Use	ers/Group	DS					?
Туре			Users	•				
Name								
E-mail	address							
Numbe	r of hits per p	page	20					
			🙀 Search					
	Name	Provider		E-mail addre	ISS	Description	Active	Locked out
2	Admin	EPi_AspNe	tldentityUserProvider	admin@alloy	/.com		×	
2	Alice	EPi_AspNe	tldentityUserProvider	alice@alloy.c	com		×	
2	Carlos	EPi_AspNe	tldentityUserProvider	carlos@alloy	.com		×	
2	Dana	EPi_AspNe	tldentityUserProvider	dana@alloy.	com		×	
2	Eve	EPi_AspNe	tldentityUserProvider	eve@alloy.ce	om		~	
2	Larry	EPi_AspNe	tldentityUserProvider	larry@alloy.c	com		×	
2	Michelle	EPi_AspNe	tldentityUserProvider	michelle@al	oy.com		×	
2	Nick	EPi_AspNe	tldentityUserProvider	nick@alloy.c	om		×	
1								

- 10. Click each new user to see which group(s) they have been made a member of and note that three new groups that were created: **CLevelExecs**, **Lawyers**, and **Personalizers**.
- 11. Open **Startup.cs**, comment out the statement that calls the **UseRegisterPersonas** extension method, and save changes.

Optional: Review alternative authentication providers

Review the following topics in the documentation and popular blog articles:

- Federated security: <u>http://world.episerver.com/documentation/developer-guides/CMS/security/federated-security/</u>
- OWIN authentication: <u>http://world.episerver.com/documentation/developer-guides/CMS/security/owin-authentication/</u>
- EPiServer CMS UI AspNetIdentity OWIN authentication: <u>http://world.episerver.com/documentation/developer-guides/CMS/security/episerver-aspnetidentity/</u>
- Configuring mixed-mode OWIN authentication: <u>http://world.episerver.com/documentation/developer-guides/CMS/security/configuring-mixed-mode-owin-authentication/</u>
- Configuring Active Directory membership provider: <u>http://world.episerver.com/documentation/developer-guides/CMS/security/Configuring-Active-Directory-membership-provider/</u>
- Integrate Azure AD using OpenID Connect: <u>http://world.episerver.com/documentation/developer-guides/CMS/security/integrate-azure-ad-using-openid-connect/</u>



Exercise A3 – Creating, editing, saving, and publishing content

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

Prerequisites: complete Exercises A1 and A2.

While working through these exercises, note the groups, users, access rights, and resources that were defined in Exercise A2, as summarized in the following table:

Groups	Users	Access rights	Resources
AccessToAdminView	Admin, Alice	All	All
AccessToAdminView, Developer	Dana	All	All
ContentCreators	Eve	Read, Create, Change, Delete	All content, except News & Events and Product pages
NewsEditors	Nick	Full	News & Events
Marketers	Mishalla	Create, Publish	All content, except News & Events
	witchelle	Create type	Product pages

Adding and publishing a new product page

1. Start the AlloyDemo website, log in as Eve, and note that she only has access to CMS | Edit and CMS | Reports, as shown in the following screenshot:

Dasł	board CMS	repr (D	?	L Eve	Q
Edit	Reports					Ŧ

2. Under the **Start** page, add a new page, as shown in the following screenshot:

Dashboard CMS	
Edit Reports	
\$	#
✓ Pages Sites Tasks Project Items	
Q Search	
B 🗐 Root	
🗖 🏫 Start	-
🖬 🗋 Alloy Plan	New Page
🗉 🕒 Alloy Track	% Cut
🖬 🕒 Alloy Meet	Сору
🖬 🕒 About us	Paste
🖬 🚍 How to buy	Move to Trash
🗉 🚍 Campaigns	•, View Approval Sequence
🗋 Search	
🖬 🛑 Customer Zone	3

3. In the list of page types, note that Product is not available for Eve, because only members of Marketers have access rights to create product pages.



4. Log out Eve, and log in as Michelle, and note that she has access to CMS | Edit, CMS | Reports, and CMS | Visitor Groups, as shown in the following screenshot:

Dash	nboard	CMS	repr	8	?	L Michelle	Q
Edit	Reports	Visitor Groups					Ŧ

- 5. Add a new page under **Start**.
- 6. Click **Product**, enter the name **Alloy Go**, and note that you must enter some USPs, as shown in the following screenshot:

New Page: Prod	uct	
Start		1 Create Cancel
Name Alloy Go		
Required proper	ties	
Unique selling points	Manage tickets Track expenses Store maps	The Unique selling points field is required.

Alloy Go will be a travel management software app that manages employee expenses, tickets, and maps.

- 7. Enter some USPs and click Create.
- 8. Note that Michelle can create a product page, and she can publish it, but she cannot edit it, as shown in the following screenshot:

			A 🗖
Start >			X Not published yet Publish? V
Name	X Alloy Go	Visible to	Last changed by you , 41 seconds ago.
Name in URL	🗶 alloy-go	Languages	Publish
Simple address	X	ID, Type	Not published yet
SEO Cont	ent		A/B Test Changes Schedule for Publish
Title	×		Crevert to Published
Keywords	×		

- 9. Publish Alloy Go.
- 10. Log out as Michelle.

Editing the product page

- 1. Log in as Eve.
- 2. Edit Alloy Go.



3. Enter a page description, as shown in the following screenshot:



Not published yet Options V	1
Last changed by you , 3 minutes ago.	
Ready to Publish	l
Not published yet	
 A/B Test Changes Revert to Published 	

- 7. In **Assets** pane, select **Media** tab, and under the **For All Sites** folder, create a new folder named **Alloy Go**.
- 8. Use your favourite search engine to find some suitable images of travel related items, and upload them to the **Alloy Go** folder.
- 9. Drag and drop the image(s) into the main body, and explore the image editor feature.
- 10. Create a Teaser block in the Alloy Go folder.

You will only be able to set the teaser to be Ready to Publish at this point.



11. Open the Alloy Go page. You previously set Alloy Go page to Ready to Publish, so it is locked. Click Options, and select Withdraw and Edit, as shown in the following screenshot:



- 12. Drag and drop the teaser block you created into the MainContentArea on Alloy Go page.
- 13. Click Options, and select Ready to Publish.
- 14. In **Navigation** pane, try to add a new page under **About us** | **News & Events**, and note that section and its children are locked for Eve, as shown in the following screenshot:



15. Log out as Eve.

Adding and publishing a new article page

1. Log in as Nick.



2. In **Navigation** pane, try to add a new page under **About us** | **News & Events** | **Events**, and note that section and its children are available for Nick, but all other pages are locked, as shown in the following screenshot:

✓ Pages Sites Tasks Project Items	
Q Search	Start > About us > News & Even
Beneficial Root	A
🖬 🏫 Start	A
🖬 🕒 Alloy Plan	
🖬 🕒 Alloy Track	
🖬 🕒 Alloy Meet	
About us	
News & Events	
🗉 📄 Events	1
🖬 🗋 Press Releases	New Page
🗉 📄 Contacts	A Cut
🗋 Management	Copy
🗉 🗋 Contact us	A Paste
🗉 🗋 Become a reseller	
🖬 🚞 How to buy	

- 3. Create an Article page under About us | News & Events | Events, named Panda is cute!
- 4. **Publish** the new page, as shown in the following screenshot:

tart > About us > News & Events > Events >	Not published yet Publish?
	Last changed by you, 5 seconds ago.
Start Alloy Plan Alloy Track Alloy Meet About u	IS Publish
AIIOY	Not published yet
	II A/B Test Changes
Start / About us / News & Events / Events /	✓ Ready to Publish
	Schedule for Publish
News & Events * Danda	Revert to Published
Events	13 Cutt.

- 5. Log out as Nick.
- 6. Log in as Larry, and note that all content is locked for Larry.
- 7. Close the browser.

Reviewing the online user guides

1. Start the AlloyDemo website, and log in as anyone.



2. In the Global menu, click Help | User Guide, as shown in the following screenshot:



3. Navigate to **CMS** | **CMS for editors** | **Creating content**, and review some of the editing features like using the rich-text editor, adding and editing images, or Working with versions, as shown in the following screenshot:

epr	☐ CMS •	🏋 Commerce 🗸	😹 Marketing 🗸	ñ Find ▼
	CMS for editors »	Getting started		
21.527 (21.91)	CMS for administrators >	Assets		
Enicon	or Onlin	Creating content		Editing content using the rich-text editor
срізег		Managing content	t »	Spellchecker for TinyMCE (Add-on)
The user quide to Epise	rver CMS. Com	Working with vers	ions	Adding links
J AND		Controlling the pu	Iblishing process >	Adding and editing images
		Search		Adding embedded media
Soarch for topics Eiltor to parrow so	parch results	CMS Editor User (Guide (PDF)	Adding dynamic content
Search for topics, Friter to Harrow Se				Working with web forms



Exercise A4 – Personalizing, approving, and A/B testing content

In this exercise, you will get an understanding of how personalization with visitor groups works, and how content approvals and A/B testing for marketing works. You will create a new page as one user, and then approve it as a sequence of other users. You will also perform A/B testing on some content.

Prerequisites: complete Exercises A1 and A2.

While working through these exercises, remember the groups, users, access rights, and resources that you defined in Exercise A2, as summarized in the following table:

Groups	Users	Access rights	Resources
ContentCreators	Eve	Full, except Administer	All content, except News & Events
NewsEditors	Nick	Full	News & Events
Marketers	Michelle	Create, Publish	All content, except News & Events
		Create type	Product pages
AccessToAdminView	Alice	All	All
Lawyers	Larry	n/a	n/a
CLevelExecs	Carlos	n/a	n/a

Define some visitor groups for personalization

- 1. Open the **Training** solution with the **AlloyDemo** project.
- 2. Start the AlloyDemo site, and log in as Alice.
- 3. Navigate to CMS | Visitor Groups, and click + Create.
- 4. Enter the following, as shown in the following screenshot:
 - a. Name: Swedish Weekenders
 - b. Notes: Visitors from Sweden at the weekend.
 - c. Security role: selected
 - d. Statistics: selected
- 5. Drag and drop from the Time and Place Criteria section, as shown in the following screenshot:
 - a. Geographic Location: Europe, Sweden
 - b. Time of Day: Day of week: Saturday and Sunday

- Criteria		Site Criteria
	Match All Y	Time and Place Criteria
	Drop new criterion here	Geographic Coordinate
		Geographic Location
	From: • To: • Day of week •	Time of Day
Geograp	hic Location Continent Europe V Country Sweden V Region Any V	
- Other Information		
Name	Swedish Weekenders	
Notes	Visitors from Sweden at the weekend.	
		URL Criteria
		Visitor Groups
Security role	↓ A whet this visitor group available when setting access rights for pages and files	
Statistics	✓ Enable statistics for this visitor group	
	🛃 Save 💽 Cancel	



- 6. Click Save.
- 7. Click + Create.
- 8. Enter the following, as shown in the following screenshot:
 - a. Name: Alloy Meet Promotion
 - b. Notes: Visitors who have expressed an interest in Alloy Meet.
 - c. Security role: selected
 - d. Statistics: selected
 - e. Match: Points
- 9. Drag and drop from the Site Criteria section, as shown in the following screenshot:
 - a. Visited Category: Alloy Meet, at least 2 pages; 2 points
 - b. User Profile: Title contains Manager; 1 point
 - c. Number of Visits: More than 3 within 60 days; 1 point
 - d. Visited Page: Alloy Meet; 3 points

	Site Criteria
Match Points T	Number of Visits
Drop new criterion here	User Profile
	Visited Category
Visited Category > Category: Alloy Meet viewed at least 2 p Required	Visited Page
User Profile > Title Contains Vanager	
Number of Visits More than Image: Second se	
Visited Page Alloy Meet [13]	Time and Place Criteria
	URL Criteria
Threshold 3/7	Visitor Groups

- 10. Set Threshold to 3 out of 7 points.
- 11. Cick Save.

Install some extra visitor groups criteria

The default 11 visitor groups criteria are shown in the following screenshots:

Site Criteria	Site Criteria	Site Criteria	Site Criteria
Number of Visits	Time and Place Criteria	Time and Place Criteria	Time and Place Criteria
User Profile	Geographic Coordinate	URL Criteria	URL Criteria
Visited Category	Geographic Location	Landing URL	Visitor Groups
II Visited Page	Time of Day	Referrer	Visitor Group Membership
		Search Keyword	
Time and Place Criteria			
URL Criteria	URL Criteria		
Visitor Groups	Visitor Groups	Visitor Groups	



After installing the Visitor Groups Criteria Pack add-on, you will have 11 more.

- 1. In Visual Studio, navigate to Tools | NuGet Package Manager | Package Manager Console.
- 2. Enter the following command:

Install-Package -ProjectName AlloyDemo EPiServer.VisitorGroupsCriteriaPack

- 3. Start the website and log in as Admin.
- 4. Navigate to CMS | Visitor Groups.

Dashboard Edit Admin	CMS Reports	Social Reach Visitor Groups	Find	Add-ons 🕫	п 🤻	?	1 Admin	Q
Visitor Gro	ups							?
Visitor groups a	re used to a	lapt the content on y	our websit	te to a specific target group.				
Name			1	Notes			Action	
Alloy Track for	free		۱ t	/isitor who have visited pages that are categorized as Alloy Traci imes.	k more th	an 2	🖉 🖹 🔥 🖊	

5. Click **Create**, and note the extra criteria, such as **Selected Language** and **Submitted Form**, as shown in the following screenshot:

dant content on v	our website by first creating visitor groups and then using the groups to target the content on pa	ner (
apt content on y	our website by first creating visitor groups and then using the groups to target the content on pa	yes.
Criteria		Site Criteria
	Match All 🔻	Number of Visits
	Drop new criterion here	Selected Language
		Submitted Form
		User Profile
Other Information	on	Visited Category
Name		Visited Page
Notes Security role	Make this visitor group available when setting access rights for pages and files	
Statistics	Enable statistics for this visitor group	Technical Criteria
		Time and Place Criteria
		URL Criteria
	🛃 Save 🔀 Cancel	Visitor Croups



The Visitor Groups Criteria Pack adds 11 more criteria, as shown in the following screenshots:

Site Criteria	Site Criteria	Site Criteria
Technical Criteria	Technical Criteria	Technical Criteria
Display channel	Time and Place Criteria	Time and Place Criteria
IP Range	Event	URL Criteria
OS & Browser	Geographic Coordinate	Downloaded file
Role	Geographic Location	# Landing URL
	Time of Day	Query String
	Time on Site	Referrer
	Time Period	E Search Keyword
Time and Place Criteria		
URL Criteria	URL Criteria	
Visitor Groups	Visitor Groups	Visitor Groups

Create some personalized content

1. In Edit view, open Assets panes, on the Blocks tab, in the Alloy Meet folder, create a Teaser block named Alloy Meet Promotion, as shown in the following screenshot:



- 2. Publish the block.
- 3. Edit the Start page, and drag and drop Alloy Meet Promotion to the top of its main content area.
- 4. In the block's context menu, click Personalize, as shown in the following screenshot:



5. Select Alloy Meet Promotion, as shown in the following screenshot:



Large content area ×
▼ 🌆 Personalized Group 🛛 🚍
Drop more content here
Alloy Meet Promotion sees
Alloy Meet Promotion
Alloy Meet jumbotron
P Contact Us
E Alloy Plan teaser
Alloy Track teaser
Alloy Meet teaser
You can drop content here, or <u>create a new block</u>

6. To preview the page as if you are a member of the visitor group, in the toolbar, toggle view settings, click **Alloy Meet Promotion**, as shown in the following sreeenshot:



Explore content approvals

You can read the Episerver CMS Editor User Guide to learn how to use this feature.

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

Episerver CMS 10.10 and later adds support for groups/roles in approval sequence steps, as well as users.

- 1. Open the Training solution with the AlloyDemo project.
- 2. Start the AlloyDemo site, and log in as Alice.
- 3. Navigate to CMS | Edit, and view the Pages in the Navigation pane.
- 4. Expand About us and News & Events.



5. Click the content menu for **Press Releases**, and choose **Edit Approval Sequence**, as shown in the following screenshot:



6. Set the approval sequence for the **Press Releases** page to **Enabled**.


7. Set Require comment on Decline.

- 8. Add three steps, and assign the groups that you created in Exercise A2, as shown in the following screenshot, and as listed in the following lettered bullets:
 - a. Legal review by a lawyer
 - b. Strategic review by a C-Level executive
 - c. Review of Swedish content by Alice, who speaks Swedish fluently, and a review of other languages by anyone with access to Edit view

Te l		
Start > About us > News & Events > Press Releases		Save
E Legal review		
🏝 Lawyers (1) 💨 👻 📶		
1)	
	0	
II Strategic review		
All		
*)	
	0	
Language review		
💄 Alice 🜒 👻 💷		
💄 AccessToEditView (5) 🔳 🔹	en da	
1]	
	0	

- It is good practice to use groups with at least two members as reviewers in each step. Assigning Alice to the Language review step is bad practice, because she might not be available to approve or decline.
 - 9. Save the sequence.

Enabling group members to approve

The members of the **Lawyers** and **CLevelExecs** groups have access to Edit view, but they have no other access rights. They must have at least **Change** access right to **Press Releases** to be able to approve content.

- 1. In Admin view, navigate to Admin | Access Rights | Set Access Rights.
- 2. In the content tree, expand About us, expand News & Events, and select Press Releases.



- 3. Clear the Inherit settings from parent item check box.
- 4. Click Add/Users Groups.
- 5. Search for, and add, Lawyers and CLevelExecs.
- 6. Give Change access rights to Lawyers and CLevelExecs, as shown in the following screenshot:

	Read	Create	Change	Delete	Publish	Administer
CLevelExecs	1		-			
CmsAdmins	1	1	-	1	1	1
Everyone	-					
awyers 🔐	1		-			
NewsEditors	1	1	•	•	\$	
Inherit settings	from par	ent item				
Apply settings f	for all sub	oitems				

7. Click Save.

Test the approval sequence

- 1. Log in as Nick.
- 2. Add a new Article to the Press Releases named Batman saves Panda!
- 3. Mark the page as **Ready for Review**, and note that the page is currently in review in step 1 of 3, as shown in the following screenshot:

t + • •	+				A 🗖
Start > About us > News & Ev	vents $ angle$ Press Releases $ angle$			•• Currently in review Options 🗸	
AIIOY	Start Alloy Plan	Alloy Track Alloy Meet	About us	In review for: 35 seconds Currently in step 1 out of 3 Requested by You, Today, 1:43 PM X Cancel Review Request and Edit	
Start / About us / N News & Events Events Press Releases	lews & Events / Press ♥	Batr Pand	nar la!	Revert to Published	J

- 4. Log out, and log in again as Larry.
- 5. Click the notification bell, click the notification, and **Approve** the article.
- 6. Log out, and log in again as Carlos.
- 7. Click the notification bell, click the notification, and **Approve** the article.
- 8. Log out, and log in again as Nick.
- 9. Click the notification bell, click the notification, and **Approve** the article.
- 10. Publish the article.
- 11. Repeat the process for a new article, but see what happens when a lawyer declines approval.



Explore A/B testing

You can read the Episerver CMS *Editor User Guide* to learn how to use this feature if your instructor did not demonstrate it.

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

Experiment with the A/B testing, as shown in the following screenshot:

	Changes to be published	Publish? 🗸
	Last changed by you , 6 seconds ago.	
	Publish Changes	
	Last published by Installer, Today, 7:28 Al	м
АЦ	/B Test Changes	
[∨ R	eady to Publish	
Сs	chedule for Publish	
ÐR	evert to Published	



Exercise A5 - Localizing content

In this exercise, you will localize some content into Swedish and Danish, including pages and blocks. Prerequisites: complete Exercise A1.

Enabling Danish language for the website content

1. Open the Training solution with the AlloyDemo project.

Make sure you use AlloyDemo, so you have lots of sample content pages that you can translate.

- 2. Start the site, and log in as Admin.
- 3. Navigate to CMS | Admin | Config | Manage Website Languages, and note that English and Swedish are already enabled by the Alloy (MVC) project template.
- 4. Click Danish (dansk) language.
- 5. Check the Enabled box, click Save, as shown in the following screenshot:

dansk - da		?
Define a new website language	that should be available to visitors on your website.	
Name	dansk	
	Enabled	
Template icon	~/app_themes/default/images/flags/da.	
Web address prefix	da	
Users/groups for creation and editing Change		
🚑 Everyone 🖉		
The second secon		
	📕 Delete	📙 Save 🔀 Cancel

6. Navigate to CMS | Edit | Pages | Start page.

🗋 Start

7. Click Tools | Language Settings, as shown in the following screenshot:

Name	Start	Visible to	Everyone Manage
Name in URL	start Change	Languages	en
Simple address	Change	ID, Type	5, Start Page
	Display in navigation		Tools 🗸
			Language Settings
			Manage Expiration and Archiving Permanently Mark as Being Edited

- 8. In Settings for Editors, click Change.
- 9. Check the dansk box, and click Save, as shown in the following screenshot:

anguage Sett.	ings for Page "Start		?
Settings for Edito	ors		
- Available Languages	3		
Languages that are de displayed to the websi set as "available". It is, previously been availa	efined as available languages on te visitors. Pages can only be cre however, also possible to acces able, but are perhaps not availabl	ly affect Edit mode and not the c ated in Edit mode in languages s and edit content on pages whi le now.	ontent that are ch have



- 10. In Settings for Site Visitors | Fallback Languages, click Change.
- 11. Set Swedish to fallback to Danish, and then English.
- 12. Set **Danish** to fallback to **Swedish**, with no second fallback, then click **Save**, as shown in the following screenshot:

Settings for Site Visi	tors					
Fallback Languages						
Fallback language replace: temporary information gaps language has been set for f	Fallback language replaces one language with another, when there are permanent or temporary information gaps for a language. Fallback language will not apply if a replacement language has been set for the pages in the structure.					
Visitor Language	Fallback Language 1	Failback Language 2				
English	•	T				
svenska	dansk 🔻	English 🔻				
dansk	svenska 🔻	T				
Save Cancel						

We could have given Danish a second fall back, but in the following steps, you will see what happens if you don't have one.

13. Close the Language Settings dialog box.

Translating content

- 1. In the **Navigation** pane, click **Sites**, and then switch to the **svenska** (Swedish) site, as shown in the screenshot:
- 2. Note the **Start** page has already been "translated" into Swedish in the Alloy (MVC) project template.



The Swedish Start page doesn't have any actual content so its Large content area is empty!

3. Edit the name of the page to **Hem** (home in Swedish) and the **Name in URL** to **hem**, as shown in the following screenshot:

🗋 Hem		Autosaved 5:06 PM Undo?			
Name	Hem	Visible to	Everyone Manage		
Name in URL	hem Change	Languages	<u>en</u> , sv		
Simple address	Change	ID, Type	5, Start Page		
	Display in navigation 🗶		Tools 🗸		
	Hem Alloy Plan Alloy Tra	ck Alloy Meet Abou	t us		

You can quickly toggle between languages for a page by clicking the language code links in the basic information area.

- 4. Publish the changes.
- 5. In **Navigation**, click **Sites**, and switch to the Danish language site, and note the page is visible to a visitor who asks for Danish (dansk) because it falls back to Swedish, as shown in the following screenshot:

🗋 Hem			× 🗭	Options ∨ :Ξ
This page is visible in dan	isk, but the content is in svenska. To edit it, you need to translate it to dansk.			Translate X
AIIOY	Hem	Search		٩



6. Click **Translate**. Note the page is NOT translated automatically for you. You must translate the text yourself (**Hjem** is Danish for home), and then click **Create**, as shown in the following screenshot:

dansk version:	
Root	Create Cancel
Name Hjem	The URL name is automatically generated based on the name you enter here

- 7. Publish the page.
- 8. View the site as a visitor and enter /da/ at the end of the address box. Note that when viewed in Danish, the Danish Start page (which is empty) is displayed, and there are no links to other pages like products, because Danish can only fallback to Swedish pages, as shown in the following screenshot:

Hjem - NOT FOR COMMIX	Maris	-		×
\leftarrow \rightarrow C \bigtriangleup localhost:60794/da/			☆]:
Hjem Se	arch	e	ρι ·	

9. Change /da/ to /sv/ to request the Swedish start page, and note that due to fallback language settings, when content is not available in Swedish, it can fall back to English content, for example links to product pages, as shown in the following screenshot:

Hem - NOT FOR COMME ×	Mark	-		×	
$\leftrightarrow \rightarrow \mathbb{C} \ \textcircled{0} \ \text{localhost:} 60794/sv/$			4	: :	
Hem Alloy Plan Alloy Track Alloy Meet About us Search		10	yr Q		

10. View the Alloy Meet page. It falls back to English, as shown in the following screenshot:



11. In the address bar, change **sv** to **da**, and note the 404 error due to strict language routing rules, as shown in the following screenshot:

The resource cannot be f X	Mark	-		×		
← → C û localhost:51110/da/alloy-meet/			☆] :		
Server Error in '/' Application.						
The resource cannot be found.						
Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed, had its name changed, or is temporarily URL and make sure that it is spelled correctly.	unavailable. Ple	ase revie	w the follo	wing		

- 12. Switch to Edit view.
- 13. Extract the folders and files in cmsdevfun_exercisefiles.zip.



There is a file in \cmsdevfun_exercisefiles\Module A\A5, named translations.pdf, with translations for all the content in English, Swedish, and Danish.

14. Translate the Alloy Meet page into Swedish, as shown in the following screenshot:

svenska version:	:
Start	Create Cancel
Name Legering Möte (Al	lloy Mee
The URL name is autom	atically generated based on the name you enter here
Required proper	ties
Unique selling points	1 🗄 projektspårning
	2 II Whiteboard skisser
	3 Inbyggd påminnelser
	4 ∷ Dela mötes resultat
	5 ∷ E-gränssnitt att begära m =-
	+

15. Translate the **Heading**, **Page description**, and **Main body** properties using the **translations.pdf** file, as shown in the following screenshot:



- 16. Publish the page.
- 17. Switch to the Danish site, translate and publish the Danish version of the Alloy Meet page.
- 18. View the site as a visitor. The Danish Alloy Meet page does not show blocks, because the blocks only exist in English, not Danish or Swedish, as shown in the following screenshot:





19. The Swedish version of the Alloy Meet page does show blocks (in English), as shown in the following screenshot:



Localizing content areas and blocks

ProductPage inherits from **StandardPage**, which has a **MainContentArea** that is not localized (because the property does not have [CultureSpecific] applied), so block references are shared by all language branches.

1. Open ~/Models/Pages/StandardPage.cs. Note the MainContentArea is not culture specific, as shown in the following code snippet:

```
public class StandardPage : SitePageData
{
    [Display(
        GroupName = SystemTabNames.Content,
        Order = 310)]
    [CultureSpecific]
    public virtual XhtmlString MainBody { get; set; }
    [Display(
        GroupName = SystemTabNames.Content,
        Order = 320)]
    public virtual ContentArea MainContentArea { get; set; }
}
```

 Open ~/Models/Blocks/TeaserBlock.cs. Note the Heading, Text, and Image properties are [CultureSpecific], as shown in the following code snippet:

```
[CultureSpecific]
[Required(AllowEmptyStrings = false)]
[Display(
    GroupName = SystemTabNames.Content,
    Order = 1)]
public virtual string Heading { get; set; }
```

3. Edit the site and switch to the Swedish version of the Alloy Meet page.



4. In the Assets pane, view Blocks, open the Alloy Meet folder, and edit the Customer testimonial wide teaser block, as shown in the following screenshot:



5. Translate the block into Swedish, as shown in the following screenshot:

For All Sites > Alloy Meet	1: Create Cancel	
Required prope	rties	
Heading	Delning i hela världen	
Text	"Alloy Meet är en mycket effektiv e-lösning för vår verksamhet. Förbättrade affärs mått realiseras genom tväravdelnings utbyte av information över hela världen." John Randle, HIGHTEC Inc	
Image	JohnRandle.jpg	

6. Publish the Swedish version of the block.

. .

7. View the site as a visitor. Note the block is translated for Swedish page.

Configuring the site to detect language preference from the browser

1. In Google Chrome, choose Settings, and search for lang, as shown in the following screenshot:

II Alloy - collabo	ration, con X Settings - Search results X		>
\rightarrow C \heartsuit	chrome://settings/search#lang		☆
Chrome	Search results	×	
Extensions	Search		
Settings	Set which search engine is used when searching from the omnibox.		
About	Google Manage search engines Languages		
	Change how Chrome handles and displays languages. <u>Learn more</u> Language and input settings		
	lang		



2. Click Language and input settings..., add Swedish and drag and drop it to the top of the list, as shown in the following screenshot:



- 3. Click **Done**, and close the **Settings** tab.
- 4. View the Start page as a visitor without a language code in the address bar, and the visitor sees the English version of the page, even though the browser is asking for Swedish, as shown in the following screenshot:



- 5. Log in and navigate to CMS | Admin | Config | System Settings.
- 6. Check the **Detect language via browser's language preference** box, and click **Save**, as shown in the following screenshot:



7. View the site as a visitor again, this time you see Swedish by default, as shown in the following screenshot:





8. All links from the Start page will include **/sv/** to request Swedish pages, as shown in the following screenshot:



Optional: Automatically translating content using Episerver Languages

Episerver Languages add-on provides easy access to a single interface for managing multiple languages and translations of content, with a built-in feature for automated translation. No additional license fee is required for the add-on, and Microsoft Azure Translator Text API can translate 2 million characters per month for free. It is an example of an add-on that registers itself as a gadget that can then be added to either the **Navigation** or the **Assets** panes.

You can skip this topic if you do not have a Microsoft Azure account. Skip to the task, Localizing the TinyMCE toolbar styles drop-down list on page 50.

- 1. Open the solution with the AlloyDemo project.
- 2. Navigate to Tools | NuGet Package Manager | Package Manager Console.
- 3. Enter the following command:

Install-Package -ProjectName AlloyDemo EPiServer.Labs.LanguageManager

- 4. Start the site, and log in as Admin.
- 5. Navigate to CMS | Edit.
- 6. In the **Navigation** pane, on the **Settings** menu, click **Add Gadgets**, as shown in the following screenshot:

	\$	Ŧ
✓ Pages Sites	Add Gadgets	
Q Search	Rearrange Gadgets	\square
B Root		
🖬 🏫 Start		≣∗
🖬 🗋 Alloy Pla	an	
🗉 🗋 Alloy Tra	ack	
🖬 🗋 Alloy Me	eet	

7. In the Gadgets picker, click Languages.



8. The **Languages** gadget will be added to the bottom of the **Navigation** pane, as shown in the following screenshot:



- 9. Sign in to the Azure portal with a Microsoft Azure account: http://portal.azure.com
- 10. Click + to create a new service, and search for **Translator Text API**, as shown in the following screenshot:



- 11. Read the description, and then click Create.
- 12. Set the following options:
 - Name: EpiserverLanguages
 - Pricing tier: F0 (2M Characters translated per month)

F0 is the free tier.

13. Click Create.

14. Navigate to the EpiserverLanguages service, as shown in the following screenshot:

\$	EpiserverLanguages Cognitive Services			
1	Search (Ctrl+/)	🛅 Delete		
		Essentials ^		
	Overview	Resource group (change)	API type	
6	Activity log	Episerver	Translator Text API	
		Status Active	Pricing tier Free	
<u></u>	Access control (IAM)	Location	Endpoint	
•	🖗 Tags	global	https://api.cognitive.microsoft.com/sts,	/v1.0
	Subscription name (change) Manage Subscription name (change) Manage Pay-As-You-Go Show a Subscription name (change)		Manage keys Show access keys	
RE	SOURCE MANAGEMENT	dd2951b0-6892-42e8-a753-1daa6e8f9dc0		
	🕈 Keys	Monitoring		
	🕰 Quick start	Characters Translated	Latency	Total Calls and Total E
•	• Pricing tier	100	100	100
	Properties			50
4	Locks	60	0	0
	Automation script		4 PM 4:30 PM	4 PM 4:30 PM

15. Click Show access keys...



16. In the **Manage keys** blade, click the copy button next to one of the two keys, as shown in the following screenshot:



17. In the Languages gadget, click the Settings menu, and click Manage Add-on Settings, as shown in the following screenshot:

∨ Languages				News & Events
English (Master)	Published	8/15/12, 8:56 PM	1	Management
🔲 svenska	Not created yet	<u>Create</u>	м	anage Website Languages
📰 dansk	Not created yet	<u>Create</u>	Manage Add-on Settings Remove Gadget	
+₽ ≡-			¢-	

18. In Admin view, in Language Manager, set Translator Provider to Bing Translate, and paste the key for the Subscription Key, as shown in the following screenshot:

Language Manag	ger
Be careful when you change ncorrect.	the configuration settings for the EPIServer Languages add-on. The add-on will not work properly if the settings are
- Provider settings	
Translator Provider	Bing Translator
Subscription Key	f837f3676646e4e7bac0f38716b515345
- Notification Provider	
Туре	•
Other settings	
Translation folder	Translation
Note: Translation folder is	the folder located under module root, the translation packages inside this folder will be imported automatically.
	Save

- 19. Click Save.
- 20. Navigate to Edit view and select About us in the Pages tree.



21. In Languages, for svenska, click Create, and then Auto-translate from English, as shown in the following screenshot:



22. After a few seconds, any properties that are culture specific will be translated automatically, as shown in the following screenshot:



- 23. Publish the Swedish page.
- 24. In Azure portal, note the **Monitoring** section, as shown in the following screenshot:

Monitoring		
Characters Translated	Latency	Total Calls and Total E
	200	6
	100	4
0.9K		2
0.6К	4:30 PM	4:30 PM
0.4K		
415 PM 430 PM 445 PM 5 PM Снижастерс такибатер 1.64 к		

25. Review the Episerver Languages User Guide online:

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



Localizing the TinyMCE toolbar styles drop-down list

- 1. Open the solution with the **AlloyDemo** project.
- 2. Start the site, and log in as **Admin**.
- 3. Edit the **Alloy Plan** product page and click inside the **Main body** property to see the TinyMCE toolbar.
- Click the Styles drop-down list box, and note the two named styles, Header 2 and Header 3, as shown in the screenshot:



Switching to Swedish

1. In the Global menu, click Admin, and then My Settings, as shown:



- 2. Click the **Display Options** tab. The Admin user has their **Personal Language** set to **Use system language**, which on my laptop, is English.
- 3. Change the language to Svenska and click Save, as shown in the following screenshot:

My Settings	?
User Information Display Options	
Language Settings Personal Language Svenska	
Views Uiews Uiews Views	
Panels to system default. Reset Views	
	ave

- 4. After a few seconds, the page refreshes, and shows all labels in Swedish.
- 5. Switch to Edit view, and navigate to CMS | Redigera, as shown in the following screenshot:

Dashboa	ard	CN	1S	Add-	ons
Redigera	Adn	nin	Ra	pporter	Besökargrupper

- Click in the Main body again and note that the label Styles has been localized into Swedish as Stilar, but the two choices, Heading 2 and Heading 3, have not, as shown in the screenshot:
- 7. Close the browser.



Localizing styles

- In Solution Explorer, open ~/Static/css/editor.css. This is the file that is used by TinyMCE to configure its options. The path to this file can be set through the Admin view or by editing the Web.config <episerver><applicationSettings uiEditorCssPaths="..." /> attribute.
- 2. Add two additional menu names, **Introduction** and **Alert Box**, and add a menu title, **Headings**, as shown in the following CSS:

```
p.introduction {
    ChangeElementType: true; /* allows change of element, i.e. h1 to p*/
```



```
EditMenuName: Introduction;
}
.alert-info {
    EditMenuName: Alert Box;
}
h2 {
    EditMenuTitle: Headings;
    EditMenuName: Header 2;
}
h3 {
    EditMenuName: Header 3;
}
.alert-info {
    background-color: #FFF8AA;
    border-color: #858585;
    color: #000000;
    font-family: Verdana;
    font-size: 12px;
}
.header.dim {
    margin: 2% 0;
    opacity: 0.3;
}
```

- 3. In ~/Resources/LanguageFiles, add a new XML file named TinyMCE.xml.
- 4. Modify its content as shown in the following markup, and note that where an edit menu name had a space in the CSS, it has been replaced with an underscore in the matching XML element name:

```
<?xml version="1.0" encoding="utf-8" ?>
<languages>
 <language name="English" id="en">
   <editorstyleoptions>
      <introduction>Introduction</introduction>
      <alert box>Alert Box</alert box>
      <headings>Headings</headings>
      <header_2>Heading 2</header_2>
      <header_3>Heading 3</header_3>
   </editorstyleoptions>
 </language>
 <language name="Svenska" id="sv">
   <editorstyleoptions>
      <introduction>Introduktion</introduction>
      <alert_box>Varningsruta</alert_box>
      <headings>Rubrikerna</headings>
      <header_2>Rubrik 2</header_2>
      <header_3>Rubrik 3</header_3>
   </editorstyleoptions>
 </language>
</languages>
```

5. Start the website, and log in as Admin.



6. Edit Alloy Plan, and click the styles drop-down, as shown in the following screenshot:



Although the names of the styles in the drop-down list, the Publish button and the blue information message have been localized into Swedish, the content types and their properties have not.

If the editor has chosen Swedish as their personal language and they edit the product page in **All Properties** view, Episerver localizes as much of the user experience labelling as possible, but it cannot automatically localize things like the page type name, custom group (tab) names like **SEO**, and custom property names like **Title** and **Keywords**, as shown in the following screenshot:

Start 〉		Autosparad 14:02 <u>ångra?</u>	Inga ändringar att publicera	Alternativ 🗸 🗐
Namn Namn i URL	Alloy Plan alloy-plan <u>Ändra</u>	Synlig för Språk	Alla <u>Hantera</u> en	
Enkel adress	Ändra	ID, typ	6, Product	
	✔Visa i navigering		Verktyg 🗸	
SEO Innehåll	Inställningar			
Title	Alloy Plan, online project			
Keywords	Project management project methodology online			

Defining the localization text values

- In ~/Resources/LanguageFiles, copy and paste ContentTypeNames.xml by pressing Ctrl + C, then Ctrl + V.
- 2. Rename the copied file to ContentTypesNames-sv.xml.
- 3. Find the <language> element and modify it as follows:

<language name="Svenska" id="sv">

4. Find the **<productpage>** element and modify it as follows:

```
<productpage>
<name>Produkt</name>
<description>Används för att presentera en specifik produkt</description>
</productpage>
```

- 5. In ~/Resources/LanguageFiles, copy and paste PropertyNames.xml.
- 6. Rename the copy to PropertyNames-sv.xml.
- 7. Find the <language> element and modify it as follows:

<language name="Svenska" id="sv">

8. Find the <meta...> elements and modify them as follows:

```
<metadescription>
    <caption>Sidbeskrivning</caption>
    <help>Används som meta beskrivning och allmänt som en ingress</help>
```



```
</metadescription>
<metakeywords>
<caption>Nyckelord</caption>
</metakeywords>
<metatitle>
<caption>Titel</caption>
</metatitle>
```

- 9. In ~/Resources/LanguageFiles, copy and paste GroupNames.xml.
- 10. Rename the copy to GroupNames-sv.xml. Modify it as follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<languages>
 <language name="Svenska" id="sv">
   <headings>
     <heading name="Contact">
        <description>Kontakta</description>
      </heading>
     <heading name="Default">
        <description>Standard</description>
      </heading>
     <heading name="News">
        <description>Nyheter</description>
      </heading>
      <heading name="Products">
        <description>Produkter</description>
      </heading>
     <heading name="MetaData">
        <description>Sökmotoroptimering</description>
      </heading>
      <heading name="SiteSettings">
        <description>Webbplatsinställningar</description>
      </heading>
      <heading name="Specialized">
        <description>Specialiserad</description>
      </heading>
   </headings>
 </language>
</languages>
```

Viewing the localizations

- 1. Start the site, and log in as Admin.
- 2. Edit the Alloy Plan page, and switch to All Properties view.
- 3. Note the parts of the user interface that are now localized into Swedish that weren't before: the page type, the group name/tab, and the property names, as highlighted in the following screenshot:

Start > Alloy Plan			Inga ändringar att publicera	Alternativ 🗸 🗐
Namn	Alloy Plan	Synlig för	Alla <u>Hantera</u>	
Namn i URL	alloy-plan <u>Ändra</u>	Språk	en	
Enkel adress	Ändra	ID, typ	6 Produkt	
	✔Visa i navigering		Verktyg 🗸	
Sökmotoroptimering	Innehåll Inställningar			
Titel	Alloy Plan, online project			
Nyckelord	Project management project methodology online			



4. Close the browser.

Optional task: Localizing all content types and property names

Use Google (or Bing) translation to localize more of the content types and their property names.

Localizing views for visitors

- 1. In the AlloyDemo project, expand the Resources/LanguageFiles folder, and open Views.xml.
- 2. Add an element inside the <language name="English" id="en"> element, as shown in bold in the following markup:

```
<language name="English" id="en">
  <productpage>
    <changed>Changed:</changed>
 </productpage>
```

3. After the closing language element, add a new language element for the Swedish translation, as shown in the following markup:

```
<language name="Swedish" id="sv">
  <productpage>
    <changed>ändrats:</changed>
 </productpage>
</language>
```

- 4. In the AlloyDemo project, expand the Views/ProductPage folder, and open Index.cshtml.
- 5. Inside the <h1> element that outputs the name of the page, add a <small> element that outputs the Changed property formatted as a long date string, as shown in the following markup:

```
<h1 @Html.EditAttribtes(x => x.CurrentPage.PageName)>
   @Model.CurrentPage.PageName
   <small>@Html.Translate("/productpage/changed")
   @(Model.CurrentPage.Changed.ToLongDateString())</small>
</h1>
```

6. Start the site, navigate to Alloy Meet page, and note the Changed output is translated and formatted using US English culture, as shown in the following screenshot:

Start / Alloy Plan



7. Change to the Swedish site, and note the Changed date is translated and formatted using English format, as shown in the following screenshot:

Start / Alloy Plan

Alloy Plan ändrats: den 12 mars 2018

8. Close the browser.



Exercise A6 – Resetting the Admin account

In this exercise, you will add functionality to reset the Admin account (if necessary).

Prerequisites: complete Exercise A1.

Adding the reset admin feature

- 1. If you haven't done so already, extract the folders and files in cmsdevfun_exercisefiles.zip.
- 2. Drag and drop the \cmsdevfun-exercisefiles\Module A\A6\Features\ folder into the AlloyDemo project.
- 3. Expand the **Features** folder and review the files included, as shown in the following screenshot:
 - a. **ResetAdmin.cs**: a static class to enable the feature.
 - b. ResetAdmin.cshtml: a Razor file for the user interface of the feature.
 - **ResetAdminController.cs**: a controller that performs the work C. of the feature.
- 4. Open ResetAdminController.cs file and modify the username and password if you want.
- 5. Open the Startup.cs-changes.txt file and copy and paste the statements into the appropriate place in the Startup.cs file.

Resetting the admin account

- 1. Start the AlloyDemo website.
- 2. On the Reset Admin page, click Yes, do it!, as shown in the following screenshot:

Reset Admin	<	Θ -	- 🗆 X	<
\leftrightarrow \rightarrow C 🛈 localhost:54	1887/ResetAdmin/Ind	ex	☆	•
Reset Admin				
Are you sure that you want	to reset the Admin u	ser account?		
User name: Admin, Passw WebAdmins	ord: Pa\$\$w0rd, Emai	l: admin@alloy.com,	Member of:	
Yes, do it!				
Log me in to Episerver CMS.				

- 🔺 🚄 Features
 - RegisterPersonas 🔺 🧲 ResetAdmin
 - C# ResetAdmin.cs
 - @ ResetAdmin.cshtml
 - ▶ C# ResetAdminController.cs Startup.cs-changes.txt
 - 🛍 Web.config



3. On the Reset Admin page, click Log me in to Episerver CMS, as shown in the following screenshot:



- 4. Log in using the new username and password.
- 5. Close the browser.

Disabling admin reset

- 1. Open the Startup.cs file.
- 2. Comment out or delete the statement that calls UseResetAdmin().
- 3. Save changes and close the file.



Exercise A7 – Identifying website features

In this exercise, you will identify the features that existing Episerver websites have implemented. **Prerequisites:** none.

Choose one of the following public websites created with Episerver CMS, and identify the following features:

- Identify what markup is part of a shared layout.
- Identify at least three page types and list their likely properties including data types.
- Identify at least three block types and list their likely properties including data types.
- Identify at least one interactive feature, like a form that the visitor can interact with.

Episerver websites

- <u>https://www.roehampton.ac.uk/</u>
- <u>https://www.mazdausa.com/</u>
- <u>https://www.absolut.com/</u>
- <u>https://www.wexphotovideo.com/</u>
- <u>https://www.southwesternrailway.com/</u>
- <u>https://www.gatwickairport.com/</u>



Module B – Defining Content Types

Goal

The overall goal of the exercises in this module is to implement typical examples of content types with templates and layouts while following good practice. You will:

- 1. Set up an **Empty** Episerver website and define a **Start** page type with a page template.
- 2. Define media types to handle generic files, image files, and SVG files.
- 3. Create shared layouts for page templates, follow good practice to define a base page type and a base page controller that uses constructor parameter injection for dependencies, and define a page view model for use in layouts, views, and controllers.
- 4. Define a **Standard** page with a **MainBody** property and an alternative layout, a **Product** page with **UniqueSellingPoints** property and a nested layout, and add a navigation menu to the root layout.

Exercise B1 – Setting up the AlloyTraining website

In this exercise, you will set up an Empty Episerver website and update it to use the latest version of Episerver CMS. You will:

- Add Bootstrap files.
- Add helper classes used in later exercises to save you time.
- Define a Start page type.
- Create a page template for the Start page type.
- Localize the Start page type for editors who speak Swedish but not English

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

Creating AlloyTraining from the Empty Episerver Web Site project template

1. In Visual Studio, open your previous solution, and navigate to File | Add | New Project...

You can use a new solution if you prefer.

- 2. In the left section, navigate to Installed | Templates | Visual C# | Episerver.
- 3. In the middle section, select Episerver Web Site project, and enter the following options:

You MUST name your project AlloyTraining. The solution classes have been written assuming that project name and so all the namespaces will use it, e.g., AlloyTraining.Models.Pages.StartPage, and so on.

- Name: AlloyTraining
- Target: .NET Framework 4.6.1 or later compatible version.
- 4. Click OK.
- 5. Choose the **Empty** template, as shown in the screenshot, and click **OK**.
- In Solution Explorer, right-click Solution 'Training' and click Set StartUp Projects.
- New Episerver Web Site AlloyTraining Select a template:

7. Click Current selection, and then click OK.

You will now be able to quickly switch between running multiple projects by selecting a project in Solution Explorer and pressing *Ctrl* + *F*5.



Installing some add-ons and updating the Episerver NuGet packages and database

- 1. In Visual Studio, navigate to Tools | NuGet Package Manager | Package Manager Console.
- 2. In Package Manager Console, set these two options:
 - a. Package source: All
 - b. Default project: AlloyTraining
- 3. Enter the following command to install Episerver Search indexing service:

Install-Package -ProjectName AlloyTraining EPiServer.Search

It is worth installing Episerver Search now so that you can use the Global search and search boxes in the Navigation and Assets panes as a CMS Editor and Administrator.

4. Enter the following command to install Episerver Search integration with CMS:

Install-Package -ProjectName AlloyTraining EPiServer.Search.Cms

5. Enter the following command to install Episerver Forms:

Install-Package -ProjectName AlloyTraining EPiServer.Forms

 Enter the following command to update all packages using restrictions defined in packages.config: Update-Package -ProjectName AlloyTraining -ToHighestMinor

You might need to restart Visual Studio a couple of times to update Entity Framework, as shown in the following screenshot:



- 7. In Package Manager Console, set these two options:
 - a. Package source: All
 - b. Default project: AlloyTraining
- 8. Enter the following command to update the CMS database:

Update-EPiDatabase

9. If you see an error message, as shown in the following screenshot, then close Visual Studio, restart, reopen the solution, select the correct **Default project**, and try again:

Package Manager	Console					r P	х
Pac <u>k</u> age source:	All	- Ø	Default project:	EmptyDemo	- ¥	5	
PM> Update-EPi There is no 'n PM>	Database uget packages' director	у					4
100 % 👻 🔍						•	
Error List Output	Package Manager Console						

Reviewing authentication and authorization in an Episerver CMS Empty site

You will not make any changes in this task; you are only reviewing the default configuration.

- 1. In the AlloyTraining project, open ~/Web.config.
- 2. Find the <authentication> element, as shown in the following configuration:

```
<authentication mode="Forms">
    <forms name=".EPiServerLogin"</pre>
```



```
loginUrl="Util/login.aspx"
    timeout="120"
    defaultUrl="~/" />
</authentication>
```

In an empty Episerver site, authentication uses Forms, aka ASP.NET Membership. 3. Find the <membership> and <rolemanager> elements, as partially shown in the following configuration: <membership defaultProvider="MultiplexingMembershipProvider" ...> <providers> <clear /> <add name="MultiplexingMembershipProvider" ... provider1="SqlServerMembershipProvider" provider2="WindowsMembershipProvider" /> <add name="WindowsMembershipProvider" ... /> <add name="SqlServerMembershipProvider" ... connectionStringName="EPiServerDB" ... minRequiredPasswordLength="6" minRequiredNonalphanumericCharacters="0" ... /> </providers> </membership> <roleManager enabled="true" defaultProvider="MultiplexingRoleProvider" cacheRolesInCookie="true"> <providers> <clear /> <add name="MultiplexingRoleProvider" ... provider1="SqlServerRoleProvider" provider2="WindowsRoleProvider" ... /> <add name="WindowsRoleProvider" ... /> <add name="SqlServerRoleProvider" ... connectionStringName="EPiServerDB" applicationName="/" /> </providers> </roleManager>

In the Empty Episerver site project template, membership and role providers are configured to use either the Windows or SQL Server providers, with a multiplexing provider configured to use SQL Server first. This allows new roles and users to be created. You can remove these and add alternative providers, for example, Microsoft Azure Active Directory.

Adding Bootstrap and Business logic

- 1. If you haven't already, then extract the folders and files in cmsdevfun_exercisefiles.zip.
- Drag and drop, or copy, the Business and Static folders from \cmsdevfun-exercisefiles\Module B\B1 to the root of the AlloyTraining project, as shown in the following screenshot:

📕 🛃 📜 🗢 AlloyTraining	g					-	×
File Home Share	View						~ 🕐
← → × ↑ 🖡 « cms	devfun-exercisefiles > Module	eB ≯ B1 ≯ AlloyTra	ining >	~ U	Search A	AlloyTraining	Q
📌 Quick access	Name	Date modified	Туре	Size			
	🜏 Business	21/11/2017 19:18	File folder				
🐳 Dropbox	🜛 Controllers	21/11/2017 19:18	File folder				
\land OneDrive - Episerver	👌 Resources	21/11/2017 19:18	File folder				
	🤳 Static	21/11/2017 19:18	File folder				
🤝 This PC	🤳 Views	21/11/2017 19:18	File folder				
< Network	😹 Web.config-changes.txt	05/12/2017 13:45	Text Document		1 KB		
6 items 2 items selected							



Make sure you drag and drop to the AlloyTraining project, NOT AlloyDemo!

3. In Solution Explorer, expand the Business and Static folders, as shown in the following screenshot:



- 4. Note the following folders and files were added:
 - Extension methods for use later in the exercises to generate menus and so on.
 - Initialization module for registering an Admin user account if using SQL Server provider.
 - Static application files for stylesheets, JavaScript libraries, and graphics.

Ü	If you are using Visual Studio 2015, ensure that the added files are part of the project by right-clicking the
	files and selecting Include In Project , as shown in the preceding screenshot (on the right), otherwise you
	will get compile errors due to missing classes.

5. Click **Build | Build Solution** to ensure the website compiles.

Testing the Empty Episerver website

- 1. In Solution Explorer, click AlloyTraining.
- 2. Start the website by navigating to Debug | Start Without Debugging or press Ctrl + F5.
- 3. You will see a HTTP 404 error message, as shown in the following screenshot:

This is expected, because the site is empty, and does not yet have a start page.



4. Enter /EPiServer/CMS/ at the end of the address box, as shown in the following screenshot:





The Episerver Web Site - Empty project template configures ASP.NET Membership providers to allow a member of the Windows group named Administrators to be recognized as an administrator of the CMS.
 Log in using a *local* Windows account that is a member of the *local* Windows group named Administrators. If you don't have a local Windows account, move on to the next task.
 In an Episerver training room, you can enter the following details for a local Windows account, as shown in the following screenshot: Name: CMSUser, Password: CMSUser
 Digital Experience Cloud[™]
 Log n
 When you bain, cookies will be used.

If you can't authenticate with a Windows account or you don't want to

You do NOT need to do this task if you logged in with CMSUser or another Windows administrator account! Skip ahead to task Defining a class of string constants for grouping content types and a Start page type on page 64.

If you do not have a *local* Windows account that is a member of the *local* Windows group named Administrators or if you see the following error message, then you can add a setting to Web.config to create a SQL-stored account instead:

	Mark	_		\times
The trust relationship be: ×				
← → C ☆ () localhost:58794/Util/login.aspx?ReturnUrl=%2fEPiServer			☆]:
Server Error in '/' Application.				
				- 1
	,			
The trust relationship between this workstation and the primar	ү аот	ain ta	allea.	
Description: An unhandled exception occurred during the execution of the current web request. Please review the stac the error and where it originated in the code.	k trace for r	nore infor	mation abo	out
Exception Details: System.SystemException: The trust relationship between this workstation and the primary domain	failed.			
				-
4				•

A good reason to avoid Windows accounts is to remove dependencies. If your project stores users in the CMS database it can easily be moved to another computer without having to configure Windows accounts.

- WARNING! The password of an SQL-stored account cannot be reset, so following these steps will delete an existing SQL-stored account named Admin (if it already exists) and recreate it with a known password. It will create the WebAdmins group (if does not already exist) and assign full access rights for the role to the Root page.
 - 1. In AlloyTraining, open Web.config.
 - 2. Add an entry to <appSettings>, as shown in the following markup:

```
<add key="alloy:RegisterAdmin" value="true" />
```



3. Find the <membership> element and set the defaultProvider to SqlServerMembershipProvider, as shown in the following markup:

<membership defaultProvider="SqlServerMembershipProvider" ...</pre>

4. Find the <roleManager> element and set the defaultProvider to SqlServerRoleProvider, as shown in the following markup:

<roleManager enabled="true" defaultProvider="SqlServerRoleProvider" ...

- 5. Save and close Web.config.
- 6. In AlloyTraining, open ~\Business\Initialization\RegisterAdminInitializationModule.cs.
- 7. Review the file to understand what it does, as shown in the following code:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.Framework;
using EPiServer.Framework.Initialization;
using EPiServer.Security;
using System.Configuration;
using System.Web.Security;
namespace AlloyTraining.Business.Initialization
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
    public class RegisterAdminInitializationModule : IInitializableModule
    {
        private const string roleName = "WebAdmins";
        private const string userName = "Admin";
        private const string password = "Pa$$w0rd";
        private const string email = "admin@alloy.com";
        public void Initialize(InitializationEngine context)
        {
            string enabledString =
                ConfigurationManager.AppSettings["alloy:RegisterAdmin"];
            bool enabled;
            if (bool.TryParse(enabledString, out enabled))
            {
                if (enabled)
                {
                    #region Use ASP.NET Membership classes to create the role and user
                    // if the role does not exist, create it
                    if (!Roles.RoleExists(roleName))
                    {
                        Roles.CreateRole(roleName);
                    }
                    // if the user already exists, delete it
                    MembershipUser user = Membership.GetUser(userName);
                    if (user != null)
                    {
                        Membership.DeleteUser(userName);
                    }
                    // create the user with password and add it to role
                    Membership.CreateUser(userName, password, email);
                    Roles.AddUserToRole(userName, roleName);
                    #endregion
```



```
#region Use EPiServer classes to give full access to root of
content tree
                    var security = context.Locate.Advanced
                        .GetInstance<IContentSecurityRepository>();
                    IContentSecurityDescriptor permissions = security
                        .Get(ContentReference.RootPage)
                        .CreateWritableClone() as IContentSecurityDescriptor;
                    permissions.AddEntry(new AccessControlEntry(
                        roleName, AccessLevel.FullAccess));
                    security.Save(ContentReference.RootPage,
                        permissions, SecuritySaveType.Replace);
                    permissions = security
                        .Get(ContentReference.WasteBasket)
                        .CreateWritableClone() as IContentSecurityDescriptor;
                    permissions.AddEntry(new AccessControlEntry(
                        roleName, AccessLevel.FullAccess));
                    security.Save(ContentReference.WasteBasket,
                        permissions, SecuritySaveType.Replace);
                    #endregion
                }
            }
        }
        public void Uninitialize(InitializationEngine context) { }
   }
}
```

- 8. Start the site, enter /EPiServer/CMS/, and confirm that you can log in as a CMS admin.
- 9. Close the browser.
- 10. In AlloyTraining, open Web.config.
- 11. Modify the entry in <appSettings> to disable the registration of Admin, as shown in the following markup:

<add key="alloy:RegisterAdmin" value="false" />

12. Save and close Web.config.

You now have full access to CMS administrator features without needing a Windows account.

Defining a class of string constants for grouping content types and a Start page type

Content types can be grouped when shown as a list for the editors. You will start by defining a static class with string constants for the group names you will use in your site.

- 1. In **Solution Explorer**, right-click **AlloyTraining** project, and click **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter SiteGroupNames.cs for the name, and click Add.
- 3. Modify the file, as shown in the following code:

namespace AlloyTraining



```
{
    public static class SiteGroupNames
    {
        public const string Specialized = "Specialized";
        public const string Common = "Common";
        public const string News = "News";
    }
}
```

Remember that you can copy and paste or drag and drop solution files into your project to save time.

- 4. In AlloyTraining, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 5. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Type, enter StartPage.cs for the name, and click Add.
- 6. In the [ContentType] attribute, change the DisplayName to Start.
- 7. In the [ContentType] attribute, set the GroupName to SiteGroupNames.Specialized and the sort index (i.e. Order) within that group to 10.
- 8. In the [ContentType] attribute, set the **Description** of "The home page for a website with an area for blocks and partial pages."
- 9. In the body of the class, uncomment the **MainBody** property with all its attributes and change its **Order** to **20**.
- 10. Add a string **Heading** property with appropriate attribute values. **Order** values are used to sort properties within a tab group. It is good practice to use multiples of ten so that future properties can be added between existing ones.

Your complete page type class should look something like the following code:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Pages
{
    [ContentType(DisplayName = "Start",
        // your code will have a random GUID here
        GroupName = SiteGroupNames.Specialized, Order = 10,
        Description = "The home page for a website with an area for blocks and partial
pages.")]
    public class StartPage : PageData
    {
        [CultureSpecific]
        [Display(Name = "Heading", Description =
            "If the Heading is not set, the page falls back to showing the Name.",
            GroupName = SystemTabNames.Content, Order = 10)]
        public virtual string Heading { get; set; }
        [CultureSpecific]
        [Display(Name = "Main body",
            Description = "The main body will be shown in the main content area of the
page, using the XHTML-editor you can insert for example text, images and tables.",
            GroupName = SystemTabNames.Content, Order = 20)]
        public virtual XhtmlString MainBody { get; set; }
    }
}
```

SystemTabNames.Content is the default so setting the GroupName attribute to this value is optional. Code in this exercise book, and in cmsdevfun_exercisefiles.zip, will not have GUIDs set for content types



so that you can use the solution code and it will match based on namespace and class name instead of GUID values.

Creating a Start page instance

- 1. Start the website by navigating to **Debug** | **Start Without Debugging** or press *Ctrl* + *F*5.
- 2. Enter /EPiServer/CMS/ at the end of the address box.
- 3. Log in as a CMS admin, for example, CMSUser or Admin.
- 4. Navigate to **CMS** | **Admin** | **Content Type**, click **Start**, and note that the synchronization process has registered your page type, including its properties, as shown in the following screenshot:

Admin Config Content Type	10-		line all	Otert						
Manage Page Types Create New Page Type Copy Page Type	[Sp The I	home pa	age for a v	Start website with	an area for blocks an	nd partial pa	iges.			
Convert Pages	ſ	- Inforn	nation —							
Page Types		From o	ode							
[Specialized] Start		Name								
Block Types		StartPa	age							
System Types		Display	/ name							
123 * ABC *		Start								
Trash									📄 Setting	js
Content Folder										
Content Polder	4	Add Pro	perty							
Content Assets Folder		N	lame	Field	Туре	Required	Localized	Searchable	Tab	From
Content Assets Folder			ame	name						
Content Assets Folder			leading	Heading	Long string (>255)		Yes	Yes	Content	Yes

Members of the virtual role **CmsAdmins** can modify many of the values that are initially set by your code attributes by clicking **Settings**. These changes will override your code, even if you subsequently change your code. **CmsAdmins** can use the **Revert to Default** button to reset to the code attribute values.

- 5. Navigate to CMS | Edit.
- 6. Navigation pane shows only a Root page, as shown in the following screenshot:

EPiServer CMS - Edit x		antelato:///4		
Pages Sites Tasks Project Items	The second secon	+		Image: Second se
☐ Root =-	Name Simple address	Root Change ØDisplay in navigation	Visible to Languages ID, Type	Everyone <u>Manage</u> en 1, Root Page Tools v
+ ≡- ¢- > Recent	Settings Sort subpages Sort index	According to sort index		

7. Click +, and then click New Page, as shown in the following screenshot:

	\$	ب ا	Te	+	ο	
✓ Pages Sites Tasks	Project Items		BRO	P Ne	ew Pag	e
Q Search				🔳 Ne	ew Blo	ck
🗏 Root		≣-		_		_
			Nan	ne		



- 8. Enter the name Start, and then click OK.
- 9. Note the following:
 - a. Pencil icon indicates a saved draft, so the page is not published yet, and it won't be visible to visitors, and the person icon can be hovered over to show who is editing the page.
 - b. Blue information icon warns that you must publish this page for visitors to see changes.
 - c. The two custom properties, **Heading** and **MainBody**, are grouped and sorted under the **Content** tab in **All Properties** view, as shown in the following screenshot:

				• • •
EPiServer CMS - Edit ×				0 – U X
← → C () localhost:64344/EPiServe	er/CMS/#context=epi.cms.	contentdata:///6		⊖, ☆ (=)
Dashboard CMS				12pr ? 💄 Admin 🔍
Edit Admin Reports Visitor Groups				
\$ ¥		▲_		
V Pages Sites Tasks Project Items	□ Start		(r	Publish2
Q Search				
Root		Autosaved 9	9:49 AM <u>Undo?</u>	
	Name	Start	Visible to	Everyone Manage
Admin	n, Today, 9:49 AM RL	start <u>Change</u>	Languages	en
	Simple address	<u>Change</u>	ID, Type	6, Start
		✓Display in navigation		Tools 🗸
	Content Settings			
	Category	Add one or more categories		
			(c.)	
	Heading	Welcome!		
+ ≡- ¢-	Main body	Paragraph - B <i>I</i>		· • • • • • • • • • • • • • • • • • • •
> Recent		This is the best Start page	e ever!	
E Project: None (use primary drafts) →				
There is no On-Page Editing (C	OPE) yet because y	ou have not created a	a page template.	
10 Entervalues for Loading	and Main hady [
TO. EITHER VALUES FOR HEADING	, anu main bouy . E	be creative!		
Every page has an inherited pr	operty named Cat	egory . If you do not v	vant to use it, you	can hide it. Our
Episerver CMS – Advanced De	evelopment training	g course shows you he	ow.	
11 Click the Dublich? button	and then click the	ne green Dublich hutt	on	
12. Navigate to CMS Admir	n Config Mana	ge Websites, and clic	ck Add Site , as sho	own in the
Dashboard CMS Add-ons			R	1/V ? 🧘 Admin Q



- 13. Enter the following details, as shown in the following screenshot:
 - a. Name: AlloyTraining



- b. URL: [copy from browser address box]
- c. **Start page**: click [...] to select the Start page you just created.
- d. Use site-specific assets: checked

Use site-specifi Edit Website	c assets enables the For This Site folder in	n the Assets pane.
Settings associated wi	h the site. C:Episerver/Training/AlloyTraining/License.config" does not exist	
General Name URL Start page	AlloyTraining http://localhost.58794/ Start [5] Vise site-specific assets	
Host Names —		
	🗶 Delet	te Site 🛃 Save 🔀 Cancel

- 14. Click Save.
- 15. Switch to **Edit view** and note that the **Start** page now shows a "house" icon and the Global menu has a "globe" to switch to Live view, as shown in the screenshot.
- 16. Switch to **Live view** and note that the Start page still returns a 404 because it does not yet have a template.
- 17. Close the browser.

	Dashboard		CMS	repr 🔇			
	Edit	Admin	Reports	Visitor Groups			
				\$	Ŧ		
`	Pag	es Sites	s Tasks	Project Items			
(Q Sei	arch			\square		
	R	oot					
	A	Start			≡-		

Creating a Start page template

A page template in MVC is a combination of a controller and a view.

- 1. In AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter StartPageController.cs for the name, and click Add, as shown in the following screenshot:

Add New Item - AlloyTraining					? ×
▲ Installed	Sort by:	Default -	• =		Search Installed Templates (Ctrl+E)
✓ Visual C# Code Data General ▷ Web	iii Ş	Block Type Block Controller (MVC) Block Partial View (MVC Razor)	Visual C# Visual C# Visual C#	Î	Type: Visual C# A page type controller for an Episerver Project. It will reference a page type with the same name as the controller but removing the suffix "Controller".
Windows Forms WPF ▷ ASP.NET Core Episerver SQL Server		Block Template (Web Forms) Page Type	Visual C# Visual C#		
Storm Items PowerShell	Ę,	Page Controller (MVC)			
▷ Online	@ *	Page Partial View (MVC Razor)	Visual C#		
	\oplus	Page Template (Web Forms)	Visual C#	-	
Name: StartPageController	.cs				
					<u>A</u> dd Cancel



3. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Pages** namespace, as shown in the following screenshot:

StartPageController.cs + ×						
🗑 AlloyTraini	ng	 AlloyTraining.Controllers. 	itartPageController	 Index(StartPage currentPage) 		
1	□using System.Collections.Generic;					
2	using System.Linq;					
3	using System.Web.Mvc;					
4	using EPiServer;					
5	using EPiServer.Core;					
6	using EPiServer.Framework.DataAnnot	tations;				
7	using EPiServer.Web.Mvc;					
8						
9	namespace AlloyTraining.Controllers	5				
10	{					
11	public class StartPageControlle	er : PageController <start< th=""><th>'age></th><th></th></start<>	'age>			
12	{					
13	public ActionResult Index(§	tartPage currentPage)				
14	{ 😔 🗸					
15	/* Implementation		own view model class the	t you pass to the view or		
16	* you can pass t using a	Alloy Iraining.Models.Pages;	SO246 The type or namespa	ice name 'StartPage' could not be found		
17	using	System.Web.WebPages;	(are you missing a using directive	e or an assembly reference?)		
18	return View(curre	ualify 'StartPage'				
19	}	auni, statt age	using EPiServer.Web.Mvc;			
20	} Genera	ate type 'StartPage'	using Alloy Harning, Models, Pages	,		
21	L} Spell c	heck 'StartPage'				
	Add re	ference to 'AlloyDemo'.	Preview changes			

- 4. In AlloyTraining, right-click Views, and add a new folder named StartPage.
- 5. Right-click StartPage, and choose Add | New Item..., or press Ctrl + Shift + A.
- 6. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the name, and click Add, as shown in the following screenshot:

Add New Item - AlloyTraining		?	×
▲ Installed	Sort by: Default - 📰 🔚 Search Installed Templates (Ctrl+E)	ρ.
 ✓ Visual C# Code Data General ▷ Web Windows Forms WPF 	Image: Second system Visual C# Type: Visual C# Image: Second system A Razor view for an Episerv reference a page type with name. Image: Second system Visual C# Image: Second system Visual C#	er Project the same	. It will
 ASP.NET Core Episenver SQL Server Storm Items PowerShell 	Page Type Visual C# C ^a Page Controller (MVC) Visual C#		
▷ Online	Page Partial View (MVC Razor) Visual C#		
Name: Index.cshtml	Add	Can	cel

- Most of the project and item templates provided by Episerver CMS Visual Studio Extension are well designed. But the name of this one is misleading, partly because there is no actual difference between a View and a Partial View; all .cshtml files are the same.
 - 7. Change the @model to use the correct page type, as shown in the following code:

@model AlloyTraining.Models.Pages.StartPage

U The existing code uses **PropertyFor** to output the **MainBody** which allows it to be edited on-page.

 Add an <h1> element to output the Heading if it exists, otherwise fall back to the Name, as shown in the following markup:

<h1< th=""><th><pre>@Html.EditAttrik</pre></th><th>oute</th><th>es(m</th><th>=></th><th><pre>m.Heading)></pre></th></h1<>	<pre>@Html.EditAttrik</pre>	oute	es(m	=>	<pre>m.Heading)></pre>
	<pre>@(Model.Heading</pre>	??	Mode	1.1	Name)
<th>></th> <th></th> <th></th> <th></th> <th></th>	>				

If you see green squiggles under Episerver methods like EditAttributes() or PropertyFor(), you can suppress compiler warning CS1702 as described in the course book on page 171.



9. Start the website, and note the page template is used to render the **Start** page, as shown in the following screenshot:



- 10. Enter /EPiServer/CMS/ at the end of the address box, and log in as a CMS admin.
- 11. Navigate to CMS | Edit., and note the On-Page Editing (OPE) experience due to the page template and use of **PropertyFor** and **EditAttributes** in the view, as shown in the following screenshot:

Pages Sites Tasks Project Items Search		
Root	🕒 Start	
♠ Start Ξ	Main body This is the best Start page EVER!	

- 12. Verify that the on-page edit logic for the <h1> tag works, i.e. that when you edit the text it is the Heading property value that is updated, and that when a value exists for Heading it is that value that is rendered to the visitor, otherwise the Name is rendered.
- 13. Empty the **Heading** property value and publish the change to verify that the fallback works, i.e. that the page name is displayed to the visitor and to the editor.
- 14. Close the browser.

Localizing to the Swedish language for the Start page and its properties

1. Drag and drop or copy the **Resources** folder from **\cmsdevfun-exercisefiles\Module B\B1** to the root of the **AlloyTraining** project.

Make sure you use **AlloyTraining**, because the demo project already has language files. 2. Expand the folder named ~\Resources\LanguageFiles\ and open the file named ContentTypes.xml, and note its contents, as shown in the following markup: <?xml version="1.0" encoding="utf-8" ?> <languages> <language name="English" id="en"> <contenttypes> <startpage> <name>Start</name> <description>The home page for a website with an area for blocks and partial pages.</description> <properties> <heading> <caption>Heading</caption> <help>If the Heading is not set, the page falls back to showing the Name.</help> </heading> <mainbody> <caption>Main body</caption>

Copyright © Episerver AB. All rights reserved.



```
<help>The main body will be shown in the main content area of the page,
using the XHTML-editor you can insert for example text, images and tables.</help>
          </mainbody>
        <properties></properties>
      </startpage>
    </contenttypes>
  </language>
  <language name="Svenska" id="sv">
    <contenttypes>
      <startpage>
        <name>Start</name>
        <description>Hemsidan för en webbplats med ett område för block och
sidor.</description>
        <properties>
          <heading>
            <caption>Rubrik</caption>
            <help>Om rubriken inte är inställd, faller sidan tillbaka för att visa
namnet.</help>
          </heading>
          <mainbody>
            <caption>Huvudkroppen</caption>
            <help>Huvudkroppen kommer att visas i huvudinnehållet på sidan med hjälp
av XHTML-redigeraren som du kan infoga tex, bilder och tabeller.</help>
          </mainbody>
        </properties>
      </startpage>
    </contenttypes>
  </language>
</languages>
```

- 3. Open Web.config.
- 4. In the <episerver.framework> element, add the following markup:

Copy and paste this element from \cmsdevfun-exercisefiles\Module B\B1\Web.config-changes.txt.

```
<le><localization fallbackBehavior="Echo, MissingMessage, FallbackCulture"
fallbackCulture="en">
    <providers>
        <add virtualPath="~/Resources/LanguageFiles" name="languageFiles"
            type="EPiServer.Framework.Localization.XmlResources
            .FileXmlLocalizationProvider, EPiServer.Framework.AspNet" />
        </providers>
        </localization>
```

Breaking change in CMS 11 Assembly name has changed to EPiServer.Framework.AspNet. For CMS 10, you would have used just EPiServer.Framework.

Testing the localization

- 1. Start AlloyTraining by navigating to Debug | Start Without Debugging or press Ctrl + F5.
- 2. Enter /EPiServer/CMS/ at the end of the address box, and log in.
- 3. In the **Global** menu, click your user name menu, and then **My Settings**, as shown in the following screenshot:




4. Click the **Display Options** tab. The Admin user has their **Personal Language** set to **Use system language**, which on my laptop, is English. Change the language to **Svenska** and click **Save**, as shown in the following screenshot:

My Settings			?
User Information Displa	y Options		
Language Settings			
Personal Language	Use system language	7	
	Use system language	1	
Minun	English		
VIEWS	Svenska		
Limit touch support	Dansk		
Reset all views to syster	Suomi	ike added, deleted or moved gadgets and restore the	
panels to system default	Nederlands		
	Norsk		
Reset Views	Deutsch		
	Español		
	Français		
	Italiano		
	日本語 (Japanese)		Save
	中文(中国) (Chinese (Simplified, China))		

5. Edit the **Start** page, switch to **All Properties** view, and note the property names and tool tips of your custom **Heading/Rubrik** and **Main body/Huvudkroppen** have been localized into Swedish, as shown in the following screenshot:

Namn	Start	Synlig för	Alla Hantera
Namn i URL	start <u>Ändra</u>	Språk	en
Enkel adress	<u>Ändra</u>	ID, typ	5, Start
	✓Visa i navigering		Verktyg 🗸
SEO Innehåll	Inställningar		
Kategori	Lägg till en eller flera kategorier +]	
Rubrik	Welcome!		
Om rubriken inte är ins	tälld faller sidan tillbaka för att visa namnet		
Huvudkroppen 😸 💀 🗃 🛩 🎁 🖾 🐲 🗇 🖬 🖬 🐼			
	B I :Ξ 1Ξ Stilar - ∽ ∽	Q	
	This is the best Start page EVER!		

6. Switch to on-page editing and note the same:



7. Pull down the Global menu, click your user name menu, click **Mina installningar**, click **Visningsalternativ**, and switch back to **Anvand systemsprak**, as shown in the following screenshot:

Mina inställningar			?
Användarinformation	sningsalternativ		
Språkinställningar —			
Personligt språk	Svenska 🔻		
	Använd systemspråk		
Vyer	English Svenska Dansk		
Återställ alla vyer till star gadgets, och återställer	Suomi Nederlands	t alla anpassningar, t.ex. tillagda, borttagna och flyttade	
Återställ vyer	Norsk Deutsch Español		
	Français		
	Italiano 日本語 (Japanese) 中文(中国) (Chinese (Simplified, China))	SI	para

8. Close the browser.



Exercise B2 - Managing media assets

In this exercise, you will define some media types to enable files to be uploaded.

Prerequisites: complete Exercise B1.

Enabling upload of any file by defining a media type

Before uploading files, a little code is needed for the system to be able to recognize media.

- 1. In AlloyTraining, expand Models, right-click Media, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Media Type, enter AnyFile.cs for the name, and click Add.
- 3. Change the **DisplayName** to **Any File**.
- 4. Add a Description of "Use this to upload any type of file."
- 5. Delete the commented example property and the commented MediaDescriptor attribute.

Your code should look something like the following:

Uploading files

- 1. Start the **AlloyTraining** website, and log in as a CMS Admin.
- 2. Navigate to CMS | Admin | Content Type, and select Any File, as shown in the following screenshot:

Dashboard CMS	HPH 🗷 ? 💄 Admin 🔍
Edit Admin Reports Visitor Groups	
Admin Config Content Type Manage Page Types Create New Page Type	Any File 📀
Copy Page Type Convert Pages	Use this to upload any type of file. Information Nome
▼ Page Types 123 ▲ BBC ▲ [Specialized] Start	AnyFile Display name
Block Types Media Types 123 - 880 -	Any File
Any File System Types	Add Property
Root Page Trash Content Folder	Name Field name Type Required Localized Searchable Tab From code
Content Assets Folder	

3. Navigate to CMS | Edit.



4. In **Assets** pane, create a folder named **Documents** in **For This Site**, as shown in the following screenshot:



It does not matter if **Blocks** or **Media** is currently selected; they both share the same folder structure.

- 5. Click Media and select the Documents folder.
- 6. Start File Explorer.
- 7. From the folder \cmsdevfun-exercisefiles\Assets\Documents, drag and drop SurfaceRT.png, EditorialTexts.txt, and Translations.pdf into the Documents folder, as shown in the following screenshots:



8. Leave the browser running, and in Visual Studio, in **Solution Explorer**, expand the **App_Data** folder, toggle **Show All Files**, refresh to show the **blobs** folder, and note that each uploaded file has its own folder that matches a content GUID in the CMS database, and a file for a version, as shown in the following screenshot:

- 087c77a0b00d47d08e2a046eb7a3ea17
 - 2953f3c5ee404d889000853759e2b974.pdf
- - 3fda6975adee4b9881e2bdb696b8cbdf.jpg
- - 3e0e74d938344cf896c46bdc9cb9a845.txt

The default BLOB provider uses the filesystem. You can configure alternative BLOB providers, for example, to use Microsoft Azure Blob Storage or Amazon Web Services S3.



- 9. Back in the browser, edit the **Start** page.
- 10. In the **Main body**, create a bulleted list, drag and drop the three files, and note that links to the media files are created, and there is no special handling for images, as shown in the following screenshot:

E + O Q E A E	Blocks Media
🕒 Start Autosaved 4:07 PM	 Image: Image: Im
Welcome! Main body	□ Documents
EditorialTexts.txt SurfaceRT.png	Upload files by dropping them here or click to browse
• translations.pdf	EditorialTexts.txt
	SurfaceRT.png
	translations.pdf

- 11. **Publish** the Start page.
- 12. Switch to **Live** view, and click on each link to see the effect for visitors, as shown in the following screenshot:

translations.pdf	×		Mark	_		×
\leftrightarrow \rightarrow C \triangle \bigcirc localh	ost:63136/siteassets/d	ocuments/translation	s.pdf		ର୍ 🕁	:
Translati	Ingliah (en)	Sendih (m) Laping Wato (May Mart)	Danish (64) Lagoring Made (May Mast)		# + -	
points	White board sketches	Whiteboard skisser	White board skitser			-

Customize handling of images by defining an image type

Before uploading image files, a little code is needed for the system to be able to recognize images as a special type of media and therefore customize how they are handled when dragged and dropped into a rich text property.

The previously uploaded image, SurfaceRT.png, will remain registered as an "Any File" content type. To give it the special image handling, it would have to be deleted and uploaded again. There is a built-in Admin view feature to convert pages, but it cannot be used to convert other content types.

- 1. In AlloyTraining, expand Models, right-click Media, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Media Type, enter ImageFile.cs for the name, and click Add.
- 3. Modify the class to inherit from ImageData.
- 4. Change the DisplayName to Image File.
- 5. Add a Description of "Use this to upload image files."



- 6. Uncomment the MediaDescriptor attribute and set the file extensions to: png,gif,jpg,jpeg
- 7. Uncomment the Description property and delete its Display attribute.

Your code should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using EPiServer.Framework.DataAnnotations;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Media
{
    [ContentType(DisplayName = "Image File",
        // your code will have a GUID
        Description = "Use this to upload image files.")]
    [MediaDescriptor(ExtensionString = "png,gif,jpg,jpeg")]
    public class ImageFile : ImageData
    {
        [CultureSpecific]
        [Editable(true)]
        public virtual string Description { get; set; }
    }
}
```

Make sure you inherit from ImageData, not MediaData!

Enabling upload of SVG files by defining a media type

ImageData does not recognise Scalable Vector Graphics (SVG) files so we must define a separate content type to handle them.

- 1. In AlloyTraining, expand Models, right-click Media, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Media Type, enter SvgFile.cs for the name, and click Add.
- 3. Modify the class to inherit from ImageData.
- 4. Change the **DisplayName** to **SVG File**.
- 5. Add a **Description** of "Use this to upload Scalable Vector Graphic (SVG) images."
- 6. Uncomment the MediaDescriptor attribute and set the file extensions to: svg
- 7. Delete the **Description** property and its **Display** attribute.
- 8. Override the Thumbnail property to return the BinaryData property.

Your code should look something like the following:



```
// instead of generating a smaller bitmap file for thumbnail,
// use the same binary vector image for thumbnail
public override Blob Thumbnail { get => base.BinaryData; }
}
```

Uploading images

- 1. Start the AlloyTraining site, and log in as a CMS Admin.
- 2. Navigate to CMS | Edit.
- 3. In **Assets** pane, click **Media**, and create a folder named **Products** in **For All Sites**, as shown in the following screenshot:



4. Upload images of the three products from **cmsdevfun_exercisefiles.zip** in the folder **\Assets \Products ** into the **Products** folder, as shown in the following screenshot:



One feature of inheriting from ImageData instead of MediaData is the automatic thumbnail generation.



5. Upload the file named **paw.svg** in the folder **\Assets \Misc** into the **For All Sites** folder, as shown in the following screenshot:

∓ ☆	
V Blocks Media Forms	
Q Search	\supset
∎ 💽 For All Sites	≣∗
For This Page	
C Upload files by dropping them here or click to browse	
paw.svg	≣-

- 6. Leave the browser running, and in Visual Studio, in Solution Explorer, in the **App_Data** folder, refresh the **blobs** folder, and note that each uploaded product image has its own folder that matches a content GUID in the CMS database, and a file for a version, and a file for a thumbnail, as shown in the following screenshot:
 - 🔺 🧉 App_Data
 - Interpretation
 - 087c77a0b00d47d08e2a046eb7a3ea17
 - 31cace57a78240b3abc6539def943599
 - fdadf506073c45bcbf3a2a08d7dc0b3e.png
 - fdadf506073c45bcbf3a2a08d7dc0b3e_Thumbnail.png
 - 7e840163f6994affb84cc61029aaac4f
 - 0113c81f14f1401fb2063bbcc52f94ff.png
 - 0113c81f14f1401fb2063bbcc52f94ff_Thumbnail.png
 - 8d9ba2ad53d249318af17aec1bce6e5c
 - daa2ab9b7c65445684481f6cec44c4e4
 - 0e4aed8a3e364af589828318738d7490.png
 - 0e4aed8a3e364af589828318738d7490_Thumbnail.png
 - de0 8878c469fc069225787
 - EPiSen 2015 Ta.mdf
 - EPiSen
 - EPiServerErrors.log.20170911.log
 - GeoLiteCity.dat
- 7. Back in the browser, double-click each image in the **Assets** pane, switch to **All Properties** view, edit the **Name** and **Description** properties, and then **Publish** them:
 - AlloyMeet.png: Name: Alloy Meet, Description: Logo of the Alloy Meet product.
 - AlloyPlan.png: Name: Alloy Plan, Description: Logo of the Alloy Plan product.
 - c. AlloyTrack.png: Name: Alloy Track, Description: Logo of the Alloy Track product.

Make sure that you publish the changes to the image metadata, or in a later exercise you will not see the names and descriptions properly.

- 8. Edit the Start page.
- 9. Drag and drop one of the product images into the **Main body** and note the image renders as an element instead of a clickable hyperlink.
- 10. Select the image.



11. In the toolbar, click **Insert/Edit Image**, click **Appearance**, and change the height to **90**, as shown in the following screenshot:

×

- 12. Click Update.
- 13. Publish the **Start** page.
- 14. Switch to Live view, to view the page as a visitor.
- 15. Right-click the page and choose View page source.
- 16. Note the HTML generated for visitors, especially the URLs for the documents and images that you uploaded as media assets, as shown in the following screenshot:



/siteassets/ is the For This Site folder.
/globalassets/ is the For All Sites folder.

Deleting referenced content

- 1. Switch to Edit view.
- In Assets, on the Media tab, select the For All Sites/Products folder, double-click Alloy Meet to edit it in All Properties view, and note the warning at the top that says it is referenced by one item, as shown in the following screenshot:

For All Sites > Products > alloy Meet		
+ <u>Back</u> Changes made h	nere will affect at least <u>1 item</u>	
Name	Alloy Meet	
Name in URL	alloymeet.png Change	

3. In **Assets**, select the context menu for **Alloy Meet**, and choose **Move to Trash**, as shown in the following screenshot:





4. Note the warning that this media is used on the Start page, as shown in the following screenshot:

Move Media to Trash	×
Would you like to move the media item to the trash?	
The media is used in the following places. To avoid errors on the site, make ${\rm s}$ not used anywhere.	ure that the media is
1 link in total	🖉 Refresh list
a 🤜 Alloy Meet	
🕒 Start	View
Move to Trash	Anyway Cancel

5. Click Cancel.

U If you click **Move to Trash Anyway**, then the embedded image on the Start page would return 404. You could then **View Trash** and **Restore** the image back to its original location.

6. Close the browser.

Exercise B3 – Implementing design patterns and conventions

Once you start working on this exercise, do not try to build or run the website until you have completed all the tasks in the exercise. If you build or run the website in the middle of the exercise, then you will see exceptions.

In this exercise, you will create a layout file which will be used up by all page templates in the website, a base page type that will be inherited from by all the page types in the site, a base page controller that will be inherited from by all the page templates in the site, and a view model to make our Views and Layouts more flexible.

- The layout will be named **_Root.cshtml**.
- The layout will be created in the folder ~\Views\Shared\Layouts and contain references to Bootstrap files you added in Exercise B1.
- You will use IContentLoader to retrieve all pages on the level below the Start page in the page tree.

The interface and its methods will be explained in more detail later in the course, this exercise merely shows how you can easily retrieve and display pages in the site programmatically.

Prerequisites: complete Exercises B1 and B2.

~\Views\Shared\Layouts_Root.cshtml
<head></head>
<body></body>
~\Views\StartPage\Index.cshtml

Creating a page type base

This class will not be a page type, but it will serve as a base class that inherits and extends PageData.

- 1. In AlloyTraining, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter SitePageData.cs for the name, and click Add.
- 3. Import the EPiServer.Core namespace.
- 4. Modify the class to be abstract and inherit from PageData.
- 5. Define five public virtual properties:
 - MetaTitle: string
 - MetaKeywords: string
 - MetaDescription: string
 - PageImage: ContentReference
 - TeaserText: string



ullet Do not apply any attributes to the properties yet. You will do that in the next section.

Applying property attributes

Before you add attributes, you will define some string constants to use in your site.

- 1. In AlloyTraining, right-click AlloyTraining project, and click Add | New Item..., or press Ctrl + Shift + A.
- In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter SiteTabNames.cs for the name, and click Add.
- 3. Modify the file, as shown in the following code, and note the following:
 - To see properties on the SEO tab, the user must have Edit access level (access right).
 - To see properties on the Site Settings tab, the user must have Administer access level.

```
using EPiServer.DataAnnotations;
using EPiServer.Security;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining
{
    [GroupDefinitions]
    public static class SiteTabNames
    {
        [Display(Order = 10)] // to sort horizontal tabs
        [RequiredAccess(AccessLevel.Edit)]
        public const string SEO = "SEO";
        [Display(Order = 20)]
        [RequiredAccess(AccessLevel.Administer)]
        public const string SiteSettings = "Site Settings";
    }
}
```

- 4. In AlloyTraining, open SitePageData.cs.
- 5. Apply attributes to the properties to:
 - Make the following properties support a plain text editor with multiple rows: MetaDescription, TeaserText.
 - Make PageImage property only able to point to images.
 - Make the following properties support multiple language branches: MetaTitle, MetaKeywords, MetaDescription, TeaserText.
 - Make the following properties appear on the SEO tab: MetaTitle, MetaKeywords, MetaDescription.
 - Make the following properties appear on the Content tab: PageImage, TeaserText.
 - Sort the properties within each tab appropriately.
 - Limit the MetaTitle to between 5 and 60 characters (Google's recommendation for page titles to get good SEO).

Your complete class should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
```



```
using EPiServer.Web;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Pages
{
    public abstract class SitePageData : PageData
    {
        [CultureSpecific]
        [Display(Name = "Meta title",
            GroupName = SiteTabNames.SEO, Order = 100)]
        [StringLength(60, MinimumLength = 5)]
        public virtual string MetaTitle { get; set; }
        [CultureSpecific]
        [Display(Name = "Meta keywords",
            GroupName = SiteTabNames.SEO, Order = 200)]
        public virtual string MetaKeywords { get; set; }
        [CultureSpecific]
        [Display(Name = "Meta description",
            GroupName = SiteTabNames.SEO, Order = 300)]
        [UIHint(UIHint.Textarea)] // multi-row text editor
        public virtual string MetaDescription { get; set; }
        [Display(Name = "Page image",
            GroupName = SystemTabNames.Content, Order = 100)]
        [UIHint(UIHint.Image)] // filters to only show images
        public virtual ContentReference PageImage { get; set; }
        [CultureSpecific]
        [Display(Name = "Teaser text",
            GroupName = SystemTabNames.Content, Order = 200)]
        [UIHint(UIHint.Textarea)]
        public virtual string TeaserText { get; set; }
    }
}
   Drag and drop \cmsdevfun-exercisefiles \Module B\B3 \Resources folder into AlloyTraining project.
```

The existing \Resources\LanguageFiles\ContentType.xml will be overwritten with a new one that has entries to localize any page types that inherit from SitePageData and its meta properties. Even without needing Swedish, it is still worth having a localization language file to translate between programmer-speak, e.g. the MetaDescription property, and editor-speak, e.g. "Page description".

Creating a view model interface

- 1. In AlloyTraining, right-click ~\Models\, and add a new folder named ViewModels.
- 2. Right-click ViewModels, and click Add | New Item..., or press Ctrl + Shift + A.
- 3. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Interface, enter IPageViewModel.cs for the name, and click Add.
- 4. Modify the interface to define a covariant generic type that derives from SitePageData, with readonly properties named CurrentPage, StartPage, MenuPages, and Section, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using EPiServer.Core;
using System.Collections.Generic;
namespace AlloyTraining.Models.ViewModels
{
    public interface IPageViewModel<out T> where T : SitePageData
```

Copyright © Episerver AB. All rights reserved.



```
{
    T CurrentPage { get; }
    StartPage StartPage { get; }
    IEnumerable<SitePageData> MenuPages { get; }
    IContent Section { get; }
}
```

Section property will be used to reference the page that is shown in the top-level navigation menu. For example, the news article Alloy Saves Bears is in the About us section.

Creating a default view model class

- 1. In AlloyTraining, right-click ViewModels, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter PageViewModel.cs for the name, and click Add.
- 3. Modify the class to implement the interface, as shown in the following code:

```
using System.Collections.Generic;
   using AlloyTraining.Models.Pages;
   using EPiServer.Core;
   namespace AlloyTraining.Models.ViewModels
   {
       public class PageViewModel<T>
            : IPageViewModel<T> where T : SitePageData
        {
            public T CurrentPage { get; set; }
            public StartPage StartPage { get; set; }
            public IEnumerable<SitePageData> MenuPages { get; set; }
            public IContent Section { get; set; }
            public PageViewModel(T currentPage)
            {
                CurrentPage = currentPage;
            }
       }
       public static class PageViewModel
        ł
            public static PageViewModel<T> Create<T>(T currentPage)
                where T : SitePageData
            {
                return new PageViewModel<T>(currentPage);
            }
       }
   }
The static PageViewModel class with its static Create<T>() method is a convenience for creating
```

Ine static PageViewModel class with its static Create<T>() method is a convenience for creating PageViewModel instances without having to specify the type because generic methods can use type inference while constructors cannot.

Creating site content icons

If you do not apply Episerver's **ImageUrl** attribute, then when editors create new pages and blocks, a missing icon is shown. You will create custom icons and apply them to your content types.



🖌 🚄 Static

CSS

a Gontenticons

🗠 epi-edu-icon-block.jpg

epi-edu-icon-page.jpg

🖂 epi-edu-icon.jpg

epi-edu-icon-search.jpg

epi-edu-icon-commerce.jpg

- In AlloyTraining, copy the contenticons folder from \cmsdevfun_exercisefiles\Module B\B3\Static to the Static folder in AlloyTraining, as shown in the screenshot:
- 2. Right-click **AlloyTraining**, and click **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter SiteContentIcons.cs for the name, and click Add.
- 4. Modify the contents, as shown in the following code:

using EPiServer.DataAnnotations;

```
namespace AlloyTraining
{
    public class SiteImageUrlAttribute : ImageUrlAttribute
    {
        public SiteImageUrlAttribute()
            : base("~/Static/contenticons/epi-edu-icon.jpg") { }
        public SiteImageUrlAttribute(string path)
            : base(path) { }
    }
    public class SitePageIconAttribute : ImageUrlAttribute
    {
        public SitePageIconAttribute()
            : base("~/Static/contenticons/epi-edu-icon-page.jpg") { }
    }
    public class SiteBlockIconAttribute : ImageUrlAttribute
    ł
        public SiteBlockIconAttribute()
            : base("~/Static/contenticons/epi-edu-icon-block.jpg") { }
    }
    public class SiteStartIconAttribute : ImageUrlAttribute
    ł
        public SiteStartIconAttribute()
            : base("~/Static/contenticons/epi-edu-icon-start.jpg") { }
    }
    public class SiteSearchIconAttribute : ImageUrlAttribute
    {
        public SiteSearchIconAttribute()
            : base("~/Static/contenticons/epi-edu-icon-search.jpg") { }
    }
    public class SiteCommerceIconAttribute : ImageUrlAttribute
    {
        public SiteCommerceIconAttribute()
            : base("~/Static/contenticons/epi-edu-icon-commerce.jpg") { }
    }
    public class SiteContainerIconAttribute : ImageUrlAttribute
    {
        public SiteContainerIconAttribute()
            : base("~/Static/contenticons/epi-edu-icon-container.jpg") { }
    }
}
```



Modifying Start page to use the base classes

- 1. In AlloyTraining, open StartPage.cs.
- 2. Modify the class to (a) inherit from **SitePageData** and (b) apply the **SiteStartIcon** attribute, as shown in the following code:

```
[ContentType(...)]
[SiteStartIcon]
public class StartPage : SitePageData
```

Creating a shared layout

- 1. In AlloyTraining, right-click ~\Views\Shared, and add a new folder named Layouts.
- 2. Drag and drop or copy the **_Root.cshtml** file from **\cmsdevfun-exercisefiles\Module B\B3\Views\Shared\Layouts** folder to the same folder in **AlloyTraining**.
- 3. Open the _Root.cshtml file, as shown in the following markup, and note the following:
 - The import of some namespaces for pages, view models and extension methods.
 - @model directive allows any type that implements the view model interface to be passed to the layout.
 - The MetaTitle (or Name as a fallback) of the page is used for the <title>
 - Links to stylesheet and script files are added to the <head>
 - @Html.RenderEPiServerQuickNavigator() adds the epi Quick Access menu

```
@using AlloyTraining.Business.ExtensionMethods
@using AlloyTraining.Models.ViewModels
@using AlloyTraining.Models.Pages
@model IPageViewModel<SitePageData>
<html lang="@Model.CurrentPage.Language">
          <head>
                    <meta charset="utf-8" />
                    <meta http-equiv="X-UA-Compatible" content="IE=10" />
                    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
                    <title>@(Model.CurrentPage.MetaTitle ?? Model.CurrentPage.Name)</title>
                    <meta name="keywords" content="@Model.CurrentPage.MetaKeywords" />
                    <meta name="description" content="@Model.CurrentPage.MetaDescription" />
                    <link rel="stylesheet" href="@Url.Content("~/Static/css/bootstrap.css")" />
                    <link rel="stylesheet"
                                   href="@Url.Content("~/Static/css/bootstrap-responsive.css")" />
                    <link rel="stylesheet" href="@Url.Content("~/Static/css/media.css")" />
                    <link rel="stylesheet" href="@Url.Content("~/Static/css/style.css")" />
                    <link rel="stylesheet" href="@Url.Content("~/Static/css/editmode.css")" />
                    <script type="text/javascript"</pre>
                                        src="@Url.Content("~/Static/js/jquery.js")"></script>
                    <script type="text/javascript"
                                        src="@Url.Content("~/Static/js/bootstrap.js")"></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script</script></script></script></script></scri
          </head>
          <body>
                    @Html.RenderEPiServerQuickNavigator()
                    <div class="container">
                              <div class="row">
                                        <div id="header">
                                                  <div class="span2">
                                                            <img src="/Static/gfx/logotype.png" />
                                                  </div>
                                                  <div class="span10">
                                                            @if (User.Identity.IsAuthenticated)
                                                            {
```



Setting the layout as the default

- 1. In AlloyTraining, right-click ~\Views, and choose Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Web, choose MVC 5 View Page (Razor), enter _ViewStart.cshtml for the Name, and click Add.
- 3. Delete all the HTML markup in _ViewStart.cshtml.
- 4. Set the layout to the path for _Root.cshtml, as shown in the following code:

```
@{
   Layout = "~/Views/Shared/Layouts/_Root.cshtml";
}
```

ViewStart.cshtml is executed when the View() method is called inside a controller's action method. It is not executed when the PartialView() method is called. This is the only difference between a "full" and "partial" .cshtml file. All .cshtml files themselves are actually the same.

Using the view model in the Start page view

- 1. Open ~\Views\StartPage\Index.cshtml.
- 2. Import namespaces for view models, modify the **@model** directive to use the default view model, and use the **CurrentPage** throughout the view markup, as shown in the following code:

```
@using AlloyTraining.Models.ViewModels
@using EPiServer.Web.Mvc.Html
@model PageViewModel<AlloyTraining.Models.Pages.StartPage>
<h1 @Html.EditAttributes(m => m.CurrentPage.Heading)>
    @(Model.CurrentPage.Heading ?? Model.CurrentPage.Name)
</h1>
</h1>
</div>
@Html.PropertyFor(m => m.CurrentPage.MainBody)
</div>
```

Registering a dependency resolver

To enable controllers to have constuctors with parameters to inject dependencies you must register a dependency resolver with ASP.NET MVC. You will create an initialization module to make sure that Episerver's integration with StructureMap is registered as the resolver when the website starts up.

You will learn more about initialization modules in Module F.

1. In AlloyTraining, right-click the Business folder and add a new folder named DependencyResolvers.



- 2. Drag and drop or copy the StructureMapDependencyResolver.cs file from \cmsdevfunexercisefiles\Module B\B3\Business\DependencyResolvers folder to the same folder in AlloyTraining.
- 3. In AlloyTraining, expand the Business folder, right-click the Initialization folder, and choose Add | New Item..., or press *Ctrl* + *Shift* + *A*.
- 4. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Initialization Module, enter RegisterDependencyResolverInitializationModule.cs for the name, and click Add.
- 5. Import the AlloyTraining.Business.DependencyResolvers, System.Web.Mvc, and EPiServer.ServiceLocation namespaces.
- 6. Modify the class to make it implement the IConfigurableModule interface.
- 7. Use Visual Studio IntelliSense to implement the missing ConfigureContainer method, as shown in the following screenshot:

.0 [InitializableModule] L1 [ModuleDependency(typeof(EPiServer.Web.Initia public class RegisterDependencyResolverInitia	lizationModule))] lizationModule : IConfigurableModule
Implement interface Implement interface explicitly Generate constructor 'RegisterDependencyResolverInitializationModule()'	SourceConfigurationContext)
.8 .9 .9 public void Uninitialize(InitializationEn 20 { 21 //Add uninitialization logic 22 } 23 } 24 }	<pre> { public void ConfigureContainer(ServiceConfigurationContext context) { throw new NotImplementedException(); } public void Initialize(InitializationEngine context) Preview changes restantiants restantiants</pre>

8. Implement the ConfigureContainer and Initialize methods, as shown in the following code:

```
using AlloyTraining.Business.DependencyResolvers; // StructureMapDependencyResolver
using EPiServer.Framework; // [InitializableModule], [ModuleDependency]
using EPiServer.Framework.Initialization; // InitializationEngine
using EPiServer.ServiceLocation; // IConfigurableModule, ServiceConfigurationContext
using System.Web.Mvc; // DependencyResolver
```

```
namespace AlloyTraining.Business.Initialization
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
    public class RegisterDependencyResolverInitializationModule : IConfigurableModule
    ł
        public void ConfigureContainer(ServiceConfigurationContext context)
        ſ
            DependencyResolver.SetResolver(
                new StructureMapDependencyResolver(context.StructureMap()));
            //Implementations for custom interfaces can be registered here.
            context.ConfigurationComplete += (o, e) =>
            ł
                //Register custom implementations that should be used in favour of the
default implementations
            };
        }
        public void Initialize(InitializationEngine context) { }
        public void Uninitialize(InitializationEngine context) { }
    }
}
```



The ConfigureContainer method is where you can add statements to replace Episerver services like IContentLoader with your own implementations, and where you can register your own custom servies.

Creating a base page controller

To enable every page to have a shared layout with a **Log out** link, and to have access to the content loader service, you will create a base page controller with a **Logout** action method.

- 1. In AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter PageControllerBase.cs for the name, and click Add.
- 3. Modify the class to make it abstract, restrict the generic type to be derived from SitePageData, and to have a Logout action method, as shown in the following code:

```
using AlloyTraining.Business.ExtensionMethods;
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using EPiServer;
using EPiServer.Core;
using EPiServer.Filters;
using EPiServer.Web.Mvc;
using System.Linq;
using System.Web.Mvc;
using System.Web.Security;
namespace AlloyTraining.Controllers
{
    public abstract class PageControllerBase<T>
        : PageController<T> where T : SitePageData
    {
        protected readonly IContentLoader loader;
        public PageControllerBase(IContentLoader loader)
        {
            this.loader = loader;
        }
        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Index");
        }
        protected IPageViewModel<TPage> CreatePageViewModel<TPage>(
            TPage currentPage) where TPage : SitePageData
        {
            var viewmodel = PageViewModel.Create(currentPage);
            viewmodel.StartPage = loader.Get<StartPage>(ContentReference.StartPage);
            viewmodel.MenuPages = FilterForVisitor.Filter(
                loader.GetChildren<SitePageData>(ContentReference.StartPage))
                .Cast<SitePageData>().Where(page => page.VisibleInMenu);
            viewmodel.Section = currentPage.ContentLink.GetSection();
            return viewmodel;
        }
    }
}
```



FilterForVisitor removes unpublished pages, pages that the current visitor does not have Read access rights to, and pages without a template. You will learn more about this class in Module E.

Modifying Start page controller to use the base class

- 1. Open StartPageController.cs.
- 2. Modify the class to derive from PageControllerBase, as shown in the following code:

```
public class StartPageController : PageControllerBase<StartPage>
```

3. In the Index action method, call the CreatePageViewModel method and pass the current page as a parameter, and then pass the view model into the view, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using EPiServer;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class StartPageController : PageControllerBase<StartPage>
    {
        public StartPageController(IContentLoader loader) : base(loader)
        {
        }
        public ActionResult Index(StartPage currentPage)
        {
            return View(CreatePageViewModel(currentPage));
        }
    }
}
```

Calling the inherited CreatePageViewModel method creates an instance of the default page view model and sets its current page, section, top level menu, and start page that we will use later.

Testing the website

1. Start the **AlloyTraining** website and note the layout is used for visitors, as shown in the following screenshot:





2. Log in and note the layout is used for on-page editing, as shown in the following screenshot:



Defining site settings on Start page for administrators only

It is good practice to store settings that affect the site on the Start page for the site.

- 1. In the AlloyTraining project, open StartPage.cs.
- 2. Add a localizable property named **FooterText**, and put it under the **Site Settings** tab, as shown in the following code:

```
[CultureSpecific]
[Display(Name = "Footer text",
    Description = "The footer text will be shown at the bottom of every page.",
    GroupName = SiteTabNames.SiteSettings, Order = 10)]
public virtual string FooterText { get; set; }
```

- 3. Start the **AlloyTraining** site, and log in as a CMS admin.
- 4. Navigate to CMS | Admin | Config | Edit Tabs, and note that the two tabs defined in SiteTabNames static class, SEO and Site Settings, have been registered from code, as shown in the following screenshot:

Dashboard CMS Add-ons Edit Admin Reports Visitor Groups							
Admin Config Content Type	Edit Tabs	available under the Edit tab. \	When you create or	edit a property, you can choose t	the tab under which yo	u want to di	(splay the
Edit Frames	Name	Display Name	Sort Index	Requires Access Level	From code	Edit	Delete
Edit Tabs	Site Settings		20	Administer	Yes		×
 Property Configuration Edit Custom Property Types 	SEO		10	Change	Yes	<i>i</i>	*
Security	Information		10	Read	No	a	×
Permissions for Functions	Scheduling		20	Read	No		*
r Tool Settings Plug-in Manager	Advanced		30	Change	No	2	*
Mirroring	Shortcut		40	Read	No	1	×
Rebuild Name for Web Addresses Search Configuration	Categories		50	Read	No	ø	×
Search Configuration							

To see and edit properties on the SEO tab, a user must have Change access rights. To see and edit properties on the Site Settings tab, a user must have Administer access rights. Do not allow users to edit a property protected in this way using On-Page Editing because that would bypass this security. Therefore, properties on the Site Settings tab should never be output in the view using PropertyFor!

5. Navigate to CMS | Admin | Admin | Administer Groups, and add a new group named WebEditors.



- 6. Navigate to CMS | Admin | Admin | Create User, and add a new user named Edward and make him a member of WebEditors.
- Navigate to CMS | Admin | Admin | Set Access Rights, select Root, add the virtual role named CmsEditors, set all access rights except Administer, click Save, as shown in the following screenshot:

Set Acces	s Rig	ints to	r "Roo)t"																	?
Restore access ri items.	ights in E	PiServer	CMS for it	ems that	you have	, for example	e, co	com	nplete	ely re	mov	ed ac	cess	to. Y	ou ca	an ch	ange	all rig	hts or	n all	
 Root Recycle I For All Si Start 	Bin tes																				
Add Users/0	Groups																				
🕮 CmsAdmins	Read	Create 💽	Change 🖌	Delete	Publish	Administer															
CmsEditors																					
Everyone																					
Inherit settings	s from pa	arent item																			
Apply settings	for all su	ubitems																			
																				📙 Sa	ave

Since **CmsEditors** do not have **Administer** access rights to the **Root** (and all its children), they will not be able to see or edit properties on the **Site Settings** tab in **All Properties** view.

8. Edit the **Start** page, switch to **All Properties** view, click **Site Settings**, enter a value for the footer text, and note that it has tabs for **SEO** and **Site Settings**, as shown in the following screenshot:

Name	Home	Visible to	Everyone Manage
Name in URL	home Change	Languages	en
Simple address	Change	ID, Type	5, Start
	✓Display in navigation		Tools 🗸
Content SEO	Site Settings Settings		
Footer text	Designed in Sweden		

We are not currently rendering the Footer text in the layout. As a challenge, output its value in a <footer> element in the _Root.cshtml layout. Hint: write an extension method that returns the FooterText property from the StartPage. A potential solution is in \cmsdevfun-exercisefiles\Module B\B3\Business\ExtensionMethods\LayoutExtensionMethods.cs and \cmsdevfun-exercisefiles\Module B\B3\Views\Shared\Layout_Root-changes.txt.

9. Log out as a CMS admin, and log back in as Edward.



10. Edit the **Start** page, switch to **All Properties** view, and note that **Edward** or other members of **CmsEditors** do not have the ability to see **Site Settings**, as shown in the following screenshot:

Dashboard CMS	
Edit Reports	
► + • • • • • • • • • • • • • • • • • •	+
Name	Start
Name in URL	start Change
Simple address	Change
	✓Display in navigation
Content SEO	Settings
Page title	

11. Click the **SEO** tab with three properties that have been localized, as shown in the following screenshot:

ers that will be shown in Google search results.
ers that will be shown in Google search results.

- If the localization strings are not loading, try switching to Swedish, and then back to English. Doing this forces the cached strings to reload and normally fixes the problem.
- 12. Click the Content tab and note the new properties for Page image and Teaser text.
- 13. Pull down the Global menu, and click the globe icon to switch to visitor view, as shown in the following screenshot:



epi		?	💄 Admin	Q	12	Heading
	То	View	Mode [http://lo	calhos	t:587	94/en/]

- 14. You are still logged in and can see the **epi** menu.
- 15. Click Log out. You are now back to an anonymous visitor.



Exercise B4 – Creating page types with a shared layout and navigation

Prerequisites: complete Exercises B1 to B3.

In this exercise, you will create a page type named Standard that will be used for generic pages in the site.

- It will inherit from SitePageData class, making sure it gets all the SEO properties and any other properties from this class.
- It will have a MainBody property for body text.
- It will use a **_LeftNavigation.cshtml** layout to have a navigation submenu.
- It will have a template, displaying some of the properties on the page type.
- You will create an instance of the Standard page named "About us".

~\ Views\Shared\Layouts_ <head> </head> <body></body>	Root.cshtml
~\Views\Shared\Layouts\	_LeftNavigation.cshtml
	~\Views\StandardPage\Index.cshtml

You will create a page type named Product that will be used for product pages in the site.

- It will inherit from StandardPage.
- It will have a template with two thirds for main content, and one third for related content.
- You will create three instances of Product page, "Alloy Meet", "Alloy Track", and "Alloy Plan".

~\Views\Shared\Layouts_Root.cshtml					
<head></head>					
<body></body>					
~\Views\Shared\Layouts_RightRelatedContent.cs ~\Views\ProductPage\Index.cshtml	shtml				



You will add a menu to the site to navigate between children of the Start page.

Creating the Standard page type

- 1. In AlloyTraining, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Type, enter StandardPage.cs for the Name, and click Add.
- 3. Modify the class to inherit from **SitePageData**.
- 4. Change the **DisplayName** to **Standard**.
- 5. Set group to SiteGroupNames.Common.
- 6. Add a Description of "Use this page type unless you need a more specialized one."
- 7. Apply the [SitePagelcon] attribute to the class.
- 8. Uncomment the MainBody property with all its attributes and change its Order to 310.

The Order in this example is set in relation to the other properties that were added on the Content tab in the SitePageData base class that this page type inherits from, for example, the TeaserText property has Order = 200 and we want the MainBody property to be displayed below it in All Properties view.

Your code should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Pages
{
    [ContentType(DisplayName = "Standard",
        GroupName = SiteGroupNames.Common,
        Description = "Use this page type unless you need a more specialized one.")]
    [SitePageIcon]
    public class StandardPage : SitePageData
    ł
        [CultureSpecific]
        [Display(Name = "Main body",
            Description = "The main body will be shown in the main content area of the
page, using the XHTML-editor you can insert for example text, images and tables.",
            GroupName = SystemTabNames.Content,
            0rder = 310)]
        public virtual XhtmlString MainBody { get; set; }
    }
}
```

Creating a left navigation layout

- 1. In AlloyTraining, right-click ~\Views\Shared\Layouts, and choose Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter _LeftNavigation.cshtml for the Name, and click Add.
- 3. Modify the view as shown in the following markup, and note the following:
 - @model allows any view model that has a CurrentPage property whose type derives from SitePageData to be passed to the layout.
 - This layout is nested inside the **_Root.cshtml** layout.



• Bootstrap is used to divide the layout into one third for the (future) left navigation submenu and two thirds for the body of the web page.

Creating a controller for the Standard page type

- 1. In Solution Explorer, in AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press *Ctrl* + *Shift* + *A*.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter StandardPageController.cs for the Name, and click Add.
- 3. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Pages** namespace.
- 4. Modify the class to derive from PageControllerBase<StandardPage>, and the Index action method to use a view model, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class StandardPageController : PageControllerBase<StandardPage>
    {
        public StandardPageController(IContentLoader loader) : base(loader)
        {
        }
        public ActionResult Index(StandardPage currentPage)
        {
            return View(CreatePageViewModel(currentPage));
        }
    }
}
```

() M

Make sure you pass the view model created by the CreatePageViewModel method to the View method, not the current page, or you will see a type mismatch runtime exception message.

5. On the Build menu, click Build Solution.

Creating a view for the Standard page

- 1. In AlloyTraining, right-click Views, and add a new folder named StandardPage.
- 2. Right-click StandardPage, and choose Add | New Item..., or press Ctrl + Shift + A.



- 3. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the name, and click Add.
- 4. Modify the view, as shown in the following markup:

You would not normally output the MetaTitle in the body of a page but we will do it just as a simple example since we have not defined a Heading property.

Creating an instance of the Standard page

1. Start the AlloyTraining website and log in.

Log out

- 2. Add a new Standard page named About us as a child of the Start page.
- 3. In on-page editing, set the **Main body** to some text, for example: "Alloy improves the effectiveness of project teams by putting the proper tools in your hands. Communication is made easy and inexpensive, no matter where team members are located." Note the two-thirds layout, as shown in the following screenshot:



4. Switch to **All Properties** view and set some approproate **SEO** properties, for example, for the **Page title**: "About us title", and **Page description**: "Alloy improves the effectiveness of project teams by putting the proper tools in your hands. Communication is made easy and inexpensive, no matter where team members are located."

bout us

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

- 5. Publish the page.
- 6. Switch to Live view and note the page's title.

Creating a selection factory for themes

- 1. In AlloyTraining, right-click Business, click Add | New Folder..., and name it SelectionFactories.
- 2. Right-click SelectionFactories, click Add | New Item..., or press Ctrl + Shift + A.



- 3. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter ThemeSelectionFactory.cs for the name, and click Add.
- 4. Import the EPiServer.Shell.ObjectEditing namespace.
- 5. Modify the class to implement ISelectionFactory.
- 6. Modify the GetSelections method to return a list of three select items for stylesheet class themes numbered 1 to 3, as shown in the following code:

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

```
namespace AlloyTraining.Business.SelectionFactories
{
    public class ThemeSelectionFactory : ISelectionFactory
    {
        public IEnumerable<ISelectItem> GetSelections(ExtendedMetadata metadata)
        {
            return new List<SelectItem>
            {
                new SelectItem { Value = "theme1", Text = "Theme 1" },
                new SelectItem { Value = "theme2", Text = "Theme 1" },
                new SelectItem { Value = "theme2", Text = "Theme 2" },
                new SelectItem { Value = "theme3", Text = "Theme 3" }
            };
        }
}
```

Creating the Product page type

- 1. In AlloyTraining, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Type, enter ProductPage.cs for the Name, and click Add.
- 3. Modify the class to inherit from StandardPage.
- 4. Change the **DisplayName** to **Product**.
- 5. Group the page under **Specialized**.
- 6. Add a Description of "Use this for software products that Alloy sells."
- 7. Apply the [SiteCommercelcon] attribute to the class.
- 8. Delete the MainBody property.
- 9. Add the following property and decorate it with the theme selection factory so the Editor can choose one theme:
 - Name: Theme
 - Type: string
- 10. Set the default theme to "theme1".
- 11. Add the following property and allow it to be localized into multiple languages:
 - Name: UniqueSellingPoints
 - Type: IList<string>
 - GroupName: SystemTabNames.Content
 - Order: 320
 - Required: force the editor to provide a value when adding a new product page

Your code should look something like the following:

Copyright © Episerver AB. All rights reserved.



```
using AlloyTraining.Business.SelectionFactories;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using EPiServer.Shell.ObjectEditing;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Pages
{
    [ContentType(DisplayName = "Product",
        GroupName = SiteGroupNames.Specialized, Order = 20,
        Description = "Use this for software products that Alloy sells.")]
    [SiteCommerceIcon]
    public class ProductPage : StandardPage
    {
        public override void SetDefaultValues(ContentType contentType)
        {
            base.SetDefaultValues(contentType);
            Theme = "theme1";
        }
        [SelectOne(SelectionFactoryType = typeof(ThemeSelectionFactory))]
        [Display(GroupName = SystemTabNames.Content, Order = 310)]
        public virtual string Theme { get; set; }
        [CultureSpecific]
        [Display(Name = "Unique selling points",
            GroupName = SystemTabNames.Content, Order = 320)]
        [Required]
        public virtual IList<string> UniqueSellingPoints { get; set; }
    }
}
```

Creating a right related content layout

- 1. Right-click ~\Views\Shared, and add a new folder named Layouts.
- 2. Right-click ~\Views\Shared\Layouts, and choose Add | New Item..., or press Ctrl + Shift + A.
- 3. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter _RightRelatedContent.cshtml for the Name, and click Add.
- 4. Modify the view as shown in the following mark, noting the following:
 - @model allows any view model that has a CurrentPage property that derives from SitePageData to be passed to the layout.
 - This layout is nested inside the _Root.cshtml layout.
 - Bootstrap is used to divide the layout into two thirds for the body of the web page and one third for the (optional) related content.



```
</div>
<div class="span4">

@RenderSection("RelatedContent", required: false)
</div></div>
```

Creating a controller for the Product page type

- 1. In **AlloyTraining**, expand **Controllers**, and right-click and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter ProductPageController.cs for the Name, and click Add.
- 3. Fix the compilation error by clicking the light bulb, and choose the option to import the AlloyTraining.Models.Pages namespace.
- 4. Modify the class to derive from PageControllerBase<T>, and the Index action method to use a view model, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class ProductPageController
        : PageControllerBase<ProductPage>
    {
        public ProductPageController(IContentLoader loader) : base(loader)
        {
        }
        public ActionResult Index(ProductPage currentPage)
        {
            return View(CreatePageViewModel(currentPage));
        }
    }
}
```

5. On the **Build** menu, click **Build Solution**.

Creating a view for the Product page

- 1. In AlloyTraining, right-click Views, and add a new folder named ProductPage.
- 2. Right-click **ProductPage**, and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 3. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.
- 4. Modify the view as shown in the following mark, noting the following:
 - This view is nested inside the _RightRelatedContent.cshtml layout.
 - The body of the view will be injected into the layout where **RenderBody()** was called, inside the first two-thirds <div>.
 - The section RelatedContent will be injected into the layout inside the last third <div>.

```
@using AlloyTraining.Models.ViewModels
@using AlloyTraining.Models.Pages
@model PageViewModel<ProductPage>
@{
Layout = "~/Views/Shared/Layouts/_RightRelatedContent.cshtml";
}
```



```
<h1 <pre>@Html.EditAttributes(x => x.CurrentPage.Name)>@Model.CurrentPage.Name</h1>
<p class="introduction"
   @Html.EditAttributes(x => x.CurrentPage.MetaDescription)>
  Model.CurrentPage.MetaDescription
<div class="row">
    <div class="span8 clearfix">
       @Html.PropertyFor(x => x.CurrentPage.MainBody)
    </div>
</div>
@section RelatedContent
{
    <div @Html.EditAttributes(x => x.CurrentPage.PageImage)>
        <img src="@Url.ContentUrl(Model.CurrentPage.PageImage)" />
    </div>
    <div class="block colorBox @Model.CurrentPage.Theme">
        <h2 @Html.EditAttributes(x => x.CurrentPage.Name)>
           @Model.CurrentPage.Name
        </h2>
         x.CurrentPage.UniqueSellingPoints)>
           @foreach(string usp in Model.CurrentPage.UniqueSellingPoints)
           {
               <span class="label label-inverse">@usp</span>
           }
        </div>
}
```

Creating instances of Product page type

- 1. Build and start the AlloyTraining website.
- 2. Add three Product pages under the Start page, using the following table:

	Alloy Meet	Alloy Plan	Alloy Track
Unique selling points	 Project tracking White board sketch Built-in reminders Share meeting results Email interface to request meetings 	 Project planning Reporting and statistics Email handling of tasks Risk calculations Direct communication to members 	 Shared timeline Project emails To-do lists Workflows Status reports
Page description	You've never had a meeting like this before!	Project management has never been easier!	Projects have a natural lifecycle with well- defined stages.
Main body	Participants from remote locations appear in your meeting room, around your table, or stand presenting at your white board.	Planning is crucial to the success of any project. Alloy Plan takes into consideration all aspects of project planning; from well-defined objectives to staffing, capital investments and management support. Nothing is left to chance.	From start-up meetings to final sign-off, we have the solutions for today's market-driven needs. Leverage your assets to the fullest through the combination of Alloy Plan, Alloy Meet and Alloy Track.
Theme	theme1	theme2	theme3



3. Set **PageImage** for each product page to the appropriate media asset, and publish the pages, as shown in the following screenshot:



4. In **Navigation** pane, on **Pages** tab, drag and drop the product pages to manully order them alphabetically, as shown in the following screenshot:



Although we could set the Start page to sort its children alphabetically, that would put About us first.

Limiting available page types

- 1. In AlloyTraining, open StartPage.cs.
- Modify the start page type to limit its children to standard pages, as shown highlighted in the following code:

```
[ContentType(...)]
[SiteStartIcon]
[AvailableContentTypes(Include = new[] { typeof(StandardPage) })]
public class StartPage : SitePageData
```

- 3. In AlloyTraining, open StandardPage.cs.
- 4. Modify the standard page type to limit its children to standard pages or their derived page types, except product pages, as shown highlighted in the following code:

```
[ContentType(...)]
[SitePageIcon]
[AvailableContentTypes(Include = new[] { typeof(StandardPage) },
      Exclude = new[] { typeof(ProductPage) })]
public class StandardPage : SitePageData
```

If you were to now try to create a Start page under the existing Start page, it will not be shown in the list of choices, but you could create one under the Root. If you were to now try to create a Product page under the About us page, it will not be shown in the list of choices, but you could create one under Start.



Creating a partial view for the navigation menu

1. In AlloyTraining, right-click ~\Views\Shared, and choose Add | New Item..., or press Ctrl + Shift + A.

Make sure you add the file to ~\Views\Shared, and NOT to ~\Views\Shared\Layouts.

- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter _NavigationMenu.cshtml for the name, and click Add.
- 3. Modify the view, as shown in the following markup:

```
@using AlloyTraining.Business.ExtensionMethods
@using AlloyTraining.Models.Pages
@using AlloyTraining.Models.ViewModels
@using EPiServer.Core
@model IPageViewModel<SitePageData>
<1i>
       @Html.ContentLink(ContentReference.StartPage)
       @foreach (SitePageData page in Model.MenuPages)
               @Html.ContentLink(page.ContentLink)
           }
       @if (User.Identity.IsAuthenticated)
       {
           <a href="/en/logout">Log out @User.Identity.Name</a>
       }
       else
       {
           <a href="@FormsAuthentication.LoginUrl?ReturnUrl="
           @Model.CurrentPage.PageLink.ExternalURLFromReference()">Log in</a>
```

Updating the root layout to use the navigation menu

- 1. Open ~\Views\Shared\Layouts_Root.cshtml.
- Inside the <div class="span10">, delete the @if statement that outputs the log in/out hyperlinks, and replace it with a call to output the navigation menu partial view, as shown in the following markup:

```
<div class="span10">
    @Html.Partial("_NavigationMenu")
</div>
```



3. Start the **AlloyTraining** website, and use the menu to navigate between pages, as shown in the following screenshot:



Helping CMS Editors by revealing help text

We have been setting a Description for most page type properties, but you probably haven't seen where they appear because the default behaviour is quite hidden. A CMS Editor must know to hover their mouse over the label for a property and wait a few seconds for a tooltip to appear. We will make it more obvious.

- 1. In the AlloyTraining project, add a new folder named ClientResources.
- 2. In the ClientResources folder, add a new folder named EditView.
- 3. In the EditView folder, create a CSS stylesheet named helpText.css.
- 4. Modify its contents, as shown in the following code:

```
.Sleek .dijitTabPaneWrapper .epi-form-container section row label[title]:after {
    background-color: #C9C9C9;
    border-radius: 10px 10px 10px 10px;
    border: 1px solid #ACACAC;
    color: #FFFFFF;
    content: "i";
    display: inline-block;
    font-size: 1em;
    font-weight: bold;
   height: 14px;
   line-height: 14px;
   margin: 0 0 0 5px;
   float: right;
   text-align: center;
   width: 14px;
}
.Sleek .dijitTabPaneWrapper .epi-formsRow label[title=""]:after,
.Sleek .dijitTabPaneWrapper .epi-form-container__section__row label[title=""]:after {
    display: none;
}
.Sleek .dijitTabPaneWrapper .epi-formsRow label[title]:hover:after,
.Sleek .dijitTabPaneWrapper .epi-form-container_section_row label[title]:hover:after
{
    background-color: #1ba4fa;
    border: 1px solid #1285de;
}
```

- 5. In AlloyTraining, add a new file named module.config.
- 6. Modify its contents, as shown in the following markup:

```
<?xml version="1.0" encoding="utf-8"?>
<module>
        <clientResources>
```



- 7. Start the AlloyTraining website and log in as a CMS administrator.
- 8. Edit the Start page and switch to All Properties view.
- 9. Hover your mouse over any of the "information" icons and note they change to blue and show a tooltip, as shown in the following screenshot:

Content	SEO	Site Settings Settings
Category	0	Add one or more categories +
OperaWritten		1,700
Heading	0	Welcome
Main body		he Heading is not set, the page falls back to showing the Name.
		Hello world

Read more about how this technique works on Daved Artemik's blog article: <u>https://beendaved.blogspot.co.uk/2016/09/simple-approach-to-tooltip-icons-for.html</u>



Module C – Rendering Content Templates

Goal

The overall goal of the exercises in this module is to see how you can customize the experience for visitors. You will:

- 1. Implement a content area property on the Start page and create a partial page template for Product pages to enable them to be rendered in the content area.
- 2. Create a partial template for all pages to enable them to be rendered in a content area.
- 3. Define display options to allow content editors to apply tags to select between multiple templates.
- 4. Apply tags programmatically to allow developers to apply tags to select between multiple templates
- 5. Create a display channel to set a tag automatically based on information in an incoming request to select between multiple templates.

Exercise C1 – Creating partial templates for product pages and image files for use in content areas

In this exercise, you will create a content area on the start page and add pages (and later blocks) to it. **Prerequisites:** complete Exercises B1 to B4.

Adding a content area to the Start page type and template

The content area on the Start page will allow only blocks or standard pages to be included.

- 1. In AlloyTraining, open ~\Models\Pages\StartPage.cs.
- 2. Add a **ContentArea** property named **MainContentArea** with appropriate attributes, as shown in the following code:

MainContentArea must only allow content references to: Standard pages, blocks, images, and folders.

```
[CultureSpecific]
[Display(Name = "Main content area",
    Description = "Drag and drop images, blocks, folders, and pages with
partial templates.",
    GroupName = SystemTabNames.Content,
    Order = 30)]
[AllowedTypes(typeof(StandardPage),
    typeof(BlockData), typeof(ImageData), typeof(ContentFolder))]
public virtual ContentArea MainContentArea { get; set; }
```

- 3. In AlloyTraining, open ~\Views\StartPage\Index.cshtml.
- 4. At the bottom of the view, output the **MainContentArea** property so that editors get an on-page editing experience, as shown in the following markup:

Testing the content area

1. Start the AlloyTraining website, and log in as a CMS admin.



- 2. In **Edit** view, open **Navigation** | **Pages**, drag and drop the three product pages into the **Main content area**, and note the warning message "The 'ProductPage' cannot be displayed".
- 3. Open **Assets** | **Media**, drag and drop the three product images from the **Products** folder into the **Main content area**, and note the warning message, "The 'ImageFile' cannot be displayed.", as shown in the following screenshot:

	▼ Image: Second s
Publish? 🗸 📙 📰	Q Search
🗅 Start	
Autosaved 10:59 AM Undo?	■ or All Sites
This is the best start page EVEP!	📄 Episerver Forms
	Products
Editorial Texts.txt SurfaceRT nng	
Translations.pdf	
more stuff	Documents
	🖪 For This Page
	Upload files by dropping them here
Main content area	or click to browse
The 'ProductPage' can not be displayed	
The 'ProductPage' can not be displayed	┿ Alloy Plan
The 'ProductPage' can not be displayed	
The 'imageFile' can not be displayed	Alloy Track
The 'ImageFile' can not be displayed	+ ≡- \$\$
You can drop content here, or <u>create a new block</u>	> Form Elements

4. In All Properties view, Main content area looks like the following screenshot:

Main content area	🗅 Alloy Meet
	🗅 Alloy Plan
	Alloy Track
	Alloy Meet
	🖃 Alloy Plan
	Alloy Track
	You can drop content here, or create a new block

5. Publish the changes to the Start page.

If you try to drag and drop the Start page into the content area it is not allowed because the MainContentArea only allows StandardPage, BlockData, ContentFolder, or ImageData content.

6. Pull down the **Global** menu and switch to visitor view. Note that instead of showing the warning message, visitors see nothing.

You will fix this issue by creating a partial page template for product pages and a template view for images.

Creating a partial content controller for product pages

1. In AlloyTraining, expand Controllers, and copy and paste the ProductPageController.cs file.


- 2. Rename the copy to ProductPagePartialController.cs.
- 3. Open ProductPagePartialController.cs.
- 4. Import the EPiServer.Web.Mvc namespace.
- 5. Rename the class to ProductPagePartialController.
- 6. Change the class to inherit from PartialContentController<ProductPage>.
- 7. Add the override keyword to the Index action method.
- 8. Modify the return statement to use **PartialView** and call PageViewModel's Create method passing the current page.

Using PartialView means it won't execute _ViewStart.cshtml, because partials do not need a layout.

The class should now look something like this:

```
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using EPiServer.Web.Mvc;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class ProductPagePartialController
        : PartialContentController<ProductPage>
        {
            public override ActionResult Index(ProductPage currentPage)
            {
                return PartialView(PageViewModel.Create(currentPage));
            }
        }
    }
}
```

Make sure that you call PartialView() method, not View() method, or the layout will render multiple times!

Creating a partial view for product pages

- 1. In AlloyTraining, right-click Views, and add a new folder named ProductPagePartial.
- 2. Expand ~\Views\ProductPage, and copy the Index.cshtml file into the folder ProductPagePartial.
- 3. Open ~\Views\ProductPagePartial\Index.cshtml.
- 4. Modify the contents, as shown in the following markup, and note the following:
 - The markup is like the "full" view, but it outputs only four properties: Name, MetaDescription, UniqueSellingPoints, and PageImage. This is a subset of the content.
 - The property output is wrapped in a <div> with a border, wrapped in a <div> with a Bootstrap **span4** class. This will allocate one third of a Bootstrap **row** width to each product.
 - The whole product is wrapped in a clickable hyperlink that would navigate the visitor to the "full" product page.



```
<p class="introduction"
               @Html.EditAttributes(x => x.CurrentPage.MetaDescription)>
               @Model.CurrentPage.MetaDescription
            <div>
                @foreach(string usp in
                    Model.CurrentPage.UniqueSellingPoints)
                {
                    <small class="label label-info" style="color:white;">@usp</small>
                }
            </div>
            <div @Html.EditAttributes(x => x.CurrentPage.PageImage)>
                <img src="@Url.ContentUrl(Model.CurrentPage.PageImage)" />
            </div>
        </a>
    </div>
</div>
```

Creating a template view for image files

1. In AlloyTraining, right-click ~\Views\Shared, and choose Add | New Item..., or press Ctrl + Shift + A.

```
    Make sure you add the file to ~\Views\Shared, and NOT to ~\Views\Shared\Layouts.
    In Add New Item - AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter ImageFile.cshtml for the Name, and click Add.
    Modify the contents, as shown in the following markup:

            @using EPiServer.Web.Mvc.Html
            @model_AlloyTraining.Models.Media.ImageFile
```

Testing the partial page and image templates

1. Start the **AlloyTraining** website, and note the visitor's view of the content area with three product pages and three product images rendered by their partial content templates:



2. Click each partial product page to confirm that it links to the correct full product page.



- 3. Right-click one of the product images, click **Inspect**, and note the following, as shown in the following screenshot:
 - A <div> element with Bootstrap class of row and equal-height that contains three <div> elements that were generated for each reference to a product page in the content area.
 - Three <div> elements with Bootstrap class of block and span4 that were output by the partial view.
 - _ × 🗋 Start × → C ☆ ③ localhost:58794 ☆ : Alloy Track Alloy Plan Alloy Meet You've never had a meeting **Project management has** Projects have a natural lifecycle with well-defined like this before! never been easier! ing White board sk stages. ing Reporting and st rs Share rr ng of tasks Risk ca e Pro ect en ls To-do lists W /s S 🕞 💼 🛛 Elements Console Sources **0** 1 × Audits Network Performance Memory Application Security <div class="row equal-height"> Styles Computed Event Listeners >>> ::before Filter :hov .cls +V<div> ▼<div class="block span4"> element.style { img { bootstrap.css:69 max-width: 100%; vertical-align: middle; border: ▶ 0; -ms_interpolation_mode: Inherited from a bootstrap.css:167 {
 color: ■#2980bd;
 tevt-decoration: ▶ none; /. </div </div </div> </div> </div> ty__div a:-webkit-any- user agent stylesheet
 link { color: webkit link; color: websa: cursor: auto; text-decoration: ▶ underline; html body div.container div.row.equal-height div div.block.span4 div.border a div img
- The URL used for the src attribute of the product images.

- 4. Log in as a CMS admin.
- 5. Due to using Bootstrap the output is responsive, and an editor can edit a product page using the partial page template and its context menu directly from the Start page, as shown in the following screenshot:





Creating a partial content template for folders

We might want to be able to add a reference to a folder inside a content area and see its name and how many items are in that folder.

- 1. In AlloyTraining, expand Models/ViewModels, right-click and choose Add | Class..., or press Shift + Alt + C.
- 2. Name the class ContentFolderViewModel.cs.
- 3. Modify the statements, as shown in the following code:

```
using EPiServer.Core;
```

namespace AlloyTraining.Models.ViewModels

```
{
    public class ContentFolderViewModel
    {
        public ContentFolder CurrentFolder { get; set; }
        public int ItemsCount { get; set; }
    }
}
```

- 4. In **AlloyTraining**, expand **Controllers**, and right-click and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 5. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Block Controller (MVC), enter ContentFolderController.cs for the name, and click Add.
- 6. Fix the compilation error by changing the base class from BlockController to PartialContentController.
- 7. Modify the statements to define a private field to store the IContentLoader and set it in the constructor, change the Index action method to use a parameter named currentContent, and create an instance of the ContentFolderViewModel and set its properties, as shown in the following code:

```
using AlloyTraining.Models.ViewModels;
using EPiServer;
using EPiServer.Core;
using EPiServer.Web.Mvc;
using System.Linq;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
```

```
{
    public class ContentFolderController : PartialContentController<ContentFolder>
    {
        private readonly IContentLoader loader;
        public ContentFolderController(IContentLoader loader)
        {
            this.loader = loader;
        }
        public override ActionResult Index(ContentFolder currentContent)
        ł
            var viewmodel = new ContentFolderViewModel
            {
                CurrentFolder = currentContent,
                ItemsCount = loader.GetChildren<IContent>
                    (currentContent.ContentLink).Count()
            };
            return PartialView(viewmodel);
        }
```



}

}

- 8. In **AlloyTraining**, right-click **Views**, and add a new folder named **ContentFolder**.
- 9. Right-click **ContentFolder**, and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 10. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the name, and click Add.
- 11. Modify the file, as shown in the following markup:

Testing the partial content template for folders

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- 2. Edit the **Start** page, and drag and drop the **Documents** folder from the **Assets** pane into the main content area, and note the name and number of items is displayed, as shown in the following screenshot:

Documents folder contains 3 items.	Large content area	×
	 Documents Alloy Meet jumbotron Alloy Plan teaser Alloy Track teaser Alloy Meet teaser Alloy Meet teaser You can drop content here, or create a new block 	

- 3. Publish the **Start** page.
- 4. Close the browser.



Exercise C2 – Creating a partial template for all pages

In this exercise, you will create partial page templates to enable any site page to render inside a content area.

Page templates can be defined for base classes and then inherited by any page type that derives from that base class. You will:

- Define a partial page template for all pages to render the page Name and MetaDescription in a "full" 3/3 width view with yellow background colour.
- Define a partial page template for all pages to render the page Name and MetaDescription in a "wide" 2/3 width view with pink background colour.
- Define a partial page template for all pages to render the page Name and MetaDescription in a "narrow" 1/3 width view with green background colour.
- Define SiteTags class with string constants for the three tag values: "Full", "Wide", and "Narrow".

Prerequisites: complete Exercises B1 to B4, C1.

Creating a partial content controller for all pages

- 1. In AlloyTraining, right-click AlloyTraining, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter SiteTags.cs for the name, and click Add.
- 3. Modify the file, as shown in the following code:

```
namespace AlloyTraining
{
    public static class SiteTags
    {
        public const string Full = "full";
        public const string Wide = "wide";
        public const string Narrow = "narrow";
    }
}
```

 ${}^{
u}$ These string constants will be used to apply tags to content templates.

- 4. In AlloyTraining, expand Controllers, and copy and paste the ProductPagePartialController.cs file.
- 5. Rename the copy to AllPagesPartialController.cs.
- 6. Open AllPagesPartialController.cs.
- 7. Rename the class to AllPagesFullPartialController.
- 8. Change all references from ProductPage to SitePageData.
- 9. Apply TemplateDescriptor attribute to mark this class as allowing page template inheritance.
- 10. Modify the call to PartialView to pass in the name of a view: SiteTags.Full

The class should now look something like this:

```
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using EPiServer.Web.Mvc;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    [TemplateDescriptor(Inherited = true,
        Tags = new[] { SiteTags.Full }, AvailableWithoutTag = true)]
    public class AllPagesFullPartialController
```



```
: PartialContentController<SitePageData>
{
    public override ActionResult Index(SitePageData currentPage)
    {
        return PartialView(viewName: SiteTags.Full,
            model: PageViewModel.Create(currentPage));
    }
}
When you explicitly aposity a view name, the secret path lacks for that pamed view in Wiewer)
```

When you explicitly specify a view name, the search path looks for that named view in ~\Views\Shared, i.e. ~\Views\Shared\Full.cshtml

Creating additional partial page templates for "wide" and "narrow"

- 1. Open AllPagesPartialController.cs.
- 2. Inside the namespace, copy the entire class and its attribute to the clipboard.
- 3. Paste twice to create two copies of the class.
 - a. Rename the first copy to: AllPagesWidePartialController
 - b. Rename the second copy to: AllPagesNarrowPartialController
- 4. Change the TemplateDescriptor attributes of the two copies to set a Tag named "wide" or "narrow" and make the two copies only available if the tag is set.
- 5. Explicitly pass the name of a partial view to use instead of Full: Wide or Narrow.

Your two copied classes should look something like the following code:

```
[TemplateDescriptor(Inherited = true,
     Tags = new[] { SiteTags.Wide }, AvailableWithoutTag = false)]
public class AllPagesWidePartialController
    : PartialContentController<SitePageData>
{
    public override ActionResult Index(SitePageData currentPage)
    ł
        return PartialView(viewName: SiteTags.Wide,
            model: PageViewModel.Create(currentPage));
    }
}
[TemplateDescriptor(Inherited = true,
    Tags = new[] { SiteTags.Narrow }, AvailableWithoutTag = false)]
public class AllPagesNarrowPartialController
    : PartialContentController<SitePageData>
{
    public override ActionResult Index(SitePageData currentPage)
    {
        return PartialView(viewName: SiteTags.Narrow,
            model: PageViewModel.Create(currentPage));
    }
}
```

Creating partial views for all pages

- 1. Expand ~\Views\ProductPageParial and copy the Index.cshtml file into the folder ~\Views\Shared.
- 2. Rename it to Full.cshtml.
- 3. Open ~\Views\Shared\Full.cshtml.
- 4. Modify the contents, as shown in the following markup, and note the following:



- PageViewModel<T> now uses a SitePageData instead of ProductPage, so that the template will work with any type of page on the site.
- It has a Bootstrap class of span12 for "full" width.
- It has a style that sets the background color to light yellow.
- It outputs three properties in this order vertically: Name, MetaDescription, and PageImage.

```
@using AlloyTraining.Models.ViewModels
@using AlloyTraining.Models.Pages
@model PageViewModel<SitePageData>
<div class="block span12" style="background-color: lightyellow;">
    <div class="border">
        <a href="@Url.ContentUrl(Model.CurrentPage.ContentLink)">
            <h2 <pre>@Html.EditAttributes(x => x.CurrentPage.Name)>
                @Model.CurrentPage.Name</h2>
            <p class="introduction"
                @Html.EditAttributes(x => x.CurrentPage.MetaDescription)>
                Model.CurrentPage.MetaDescription
            <div @Html.EditAttributes(x => x.CurrentPage.PageImage)>
                <img src="@Url.ContentUrl(Model.CurrentPage.PageImage)" />
            </div>
        \langle a \rangle
    </div>
</div>
```

- 5. Save and close Full.cshtml.
- 6. Copy and paste Full.cshtml twice:
 - Rename the first copy: Wide.cshtml
 - Rename the second copy: Narrow.cshtml
- 7. Open Wide.cshtml.
- 8. Modify the contents, as shown in the following markup, and note the following:
 - It has a Bootstrap class of **span8** for "wide" 2/3 width.
 - It has a style that sets the background color to light pink (lavenderblush).
 - It outputs two properties in this order vertically: PageImage and Name.

- 9. Open Narrow.cshtml.
- 10. Modify the contents, as shown in the following markup, and note the following:
 - It has a Bootstrap class of **span4** for "narrow" 1/3 width.
 - It has a style that sets the background color to pale green.



• It outputs two properties in this order vertically: Name and MetaDescription.

Testing the partial page template

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- 2. Edit the About us page, and switch to All Properties view.
- 3. In Assets pane, click Media.
- 4. Upload the image ~\Assets\Misc\FindReseller.png to the folder For This Site.
- 5. Drag and drop the image **FindReseller.png** to the **Page image** property of the **About us** page, as shown in the following screenshot:

te + 💿 Q	+		A 🖪	F 🗘
Start >			■ Options ∨	Q Search
				o For All Sites
Name	About us	Visible to	Everyone Manage	■ For This Site =•
Name in URL	about-us Change	Languages	en	Products
Simple address	Change	ID, Type	7, Standard	🖪 For This Page
	✓Display in navigation		Tools 🗸	
SEO Content	Settings			
Category	Add one or more categories	+		Upload files by dropping them here or click to browse
Page image				FindReseller.png
Teaser text	Fin	ndReseller.png		

- 6. Publish the change.
- 7. Edit the Start page.
- 8. Drag and drop the **About us** page into the main content area, as shown in the following screenshot, and note that it uses the "full" partial page template:



$ \begin{array}{ c c c c c } \hline \hline & EPiServer CMS - Edit & \times \\ \hline \hline & & & \\ \hline \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \hline \\ \hline & & & \\ \hline \hline \\ \hline \hline & & & \\ \hline \hline \hline \\ \hline \hline \hline \\ \hline \hline \hline \hline$	Mark – □ × MS/#viewsetting=viewlanguage:///en&context=epi.cms.contentdata:///5 ☆ :
 Pages Sites Tasks Project Items Search Root Start Alloy Meet Alloy Plan Alloy Track 	H Options ∨ ⋮≡ Start
C About us	About us Alloy improves the effectiveness of project teams by putting the proper tools in your hands. Communication is made easy and inexpensive, no matter where team members are located.
+ =- ¢- > Recent	

- 9. Switch to All Properties view.
- 10. Select the context menu for the reference to the **About us** page, and note there are no display options to assign alternative tags, as shown in the following screenshot:

Main content area	Documents		
	Alloy Plan	1	Edit
	🕒 Alloy Track	21	Personalize
	Alloy Meet		Move Outside Group
	Alloy Plan		Move Up Move Down
	Alloy Track		Remove
	🕒 About us		
	You can drop content here, or <u>create a new block</u>		

In the next exercise, you will allow the content editor to choose display options to apply one of three "tags" to switch between the three partial page templates.



Exercise C3 – Enabling editors to apply tags manually using display options

In this exercise, you will define three display options to alloy an editor to apply tags manually to individual content items in a content area to switch between partial page templates.

Prerequisites: complete Exercises B1 to B4, C1 and C2.

Adding display options using an initialization module

Display options for applying tags to content references should be registered during initialization.

- 1. In AlloyTraining, right-click Business, and add a folder named Initialization.
- 2. In ~\Business\Initialization, add an Episerver | Initialization Module named DisplayOptionsInitializationModule.cs.
- 3. Modify the class, as shown in the following code:

```
using EPiServer.Framework;
using EPiServer.Framework.Initialization;
using EPiServer.Web;
namespace AlloyTraining.Business.Initialization
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
    public class DisplayOptionsInitializationModule : IInitializableModule
    ł
        public void Initialize(InitializationEngine context)
        {
            var options =
                context.Locate.Advanced.GetInstance<DisplayOptions>();
            options.Add(id: SiteTags.Full, name: "Full", tag: SiteTags.Full);
            options.Add(id: SiteTags.Wide, name: "Wide", tag: SiteTags.Wide);
            options.Add(
                id: SiteTags.Narrow, name: "Narrow", tag: SiteTags.Narrow);
        }
        public void Uninitialize(InitializationEngine context) { }
    }
}
```

- 4. In ~\Resources\LanguageFiles, add an XML file named DisplayOptions.xml.
- 5. Modify the file, as shown in the following markup:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<languages>
 <language name="English" id="en">
   <displayoptions>
     <full>Full</full>
      <wide>Wide</wide>
      <narrow>Narrow</narrow>
   </displayoptions>
 </language>
 <language name="Swedish" id="sv">
    <displayoptions>
     <full>Full</full>
      <wide>Bred</wide>
      <narrow>Smal</narrow>
    </displayoptions>
 </language>
```



</languages>

Testing display options

- 1. Start the AlloyTraining website and log in as a CMS admin.
- 2. Edit the Start page and switch to All Properties view.
- 3. Click Content tab and scroll down to the Main content area property.
- 4. Select **About us**, and click the context menu to choose a **Display As** option, for example **Narrow**, as shown in the following screenshot:

Main content area	Documents]	
	🕒 Alloy Meet	/	Edit	
	🗋 Alloy Plan		Display As: Narrow 🕨	Display options
	🕒 Alloy Track	2	Personalize	 Automatic
	Alloy Meet			0.5.1
	Alloy Plan	T ↓	Move Op Move Down	O Wide
	Alloy Track		Remove	Narrow
	🗅 About us	-*		
	You can drop content here, or create a new block			

- 5. Publish the change and note the partial page template used to render the **About us** page has been changed to use the **Narrow** template (that doesn't show the image and it has a green background).
- 6. Try applying the **Wide** display option, and note it renders the image above the title, without a description, as shown in the following screenshot:



- 7. Publish the page.
- 8. Close the browser.

Display As options are shown for all content references in a content area, even if they would have no affect. For example, the references to the three images show the same Display As options, but the images do not have different templates. The references to the three product pages can have the display options applied because ProductPage inherits indirectly from SitePageData. It is possible to filter display options based on the content type of the current selection, but that is an advanced technique not covered in this course.



Exercise C4 – Applying tags to content areas with code

In this exercise, you will create two content areas on the product page and then control which partial page template is used by programmatically applying a tag: full, wide, or narrow.

Prerequisites: complete Exercises B1 to B4, C1 to C2.

Adding a content area to the Product page type and template

- 1. In AlloyTraining, open ~\Models\Pages\ProductPage.cs.
- 2. Add two **ContentArea** properties with appropriate attributes: **MainContentArea** and **RelatedContentArea**, as shown in the following code:

```
[Display(Name = "Main content area",
    Description = "Drag and drop blocks and pages with partial templates.",
    GroupName = SystemTabNames.Content,
    Order = 330)]
public virtual ContentArea MainContentArea { get; set; }
[Display(Name = "Related content area",
    Description = "Drag and drop blocks and pages with partial templates.",
    GroupName = SystemTabNames.Content,
    Order = 340)]
public virtual ContentArea RelatedContentArea { get; set; }
```

- 3. In AlloyTraining, open ~\Views\ProductPage\Index.cshtml.
- 4. Import the AlloyTraining namespace, as shown in the following markup:

@using AlloyTraining

5. Above @section RelatedContent, render the MainContentArea property so that editors get an on-page editing experience, set Bootstrap classes of row and equal-height, and set the wide site tag, as shown in the following markup:

6. At the bottom of @section RelatedContent, before the close brace, render the RelatedContentArea property so that editors get an on-page editing experience, set Bootstrap classes of row and equal-height, and set the narrow site tag, as shown in the following markup:

Testing the content areas on product pages

- 1. Start the **AlloyTraining** website, and log in as a CMS admin.
- 2. Edit one of the product pages, for example, Alloy Meet.



3. Drag and drop **About us** from the **Navigation** pane's **Pages** tree, into the **Main content area**, as shown in the following screenshot:



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.



4. Drag and drop **About us** from the **Navigation** pane's **Pages** tree, into the **Related content area**, and note that different partial page templates are used (Wide.cshtml and Narrow.cshtml) due to the tags applied programmatically, as shown in the following screenshot:



- 5. Publish the page.
- 6. Close the browser.



Exercise C5 – Applying tags automatically using a display channel

In this exercise, you will create a page template for Start page optimized for mobile devices and a display channel that automatically sets the mobile tag based on information in an incoming HTTP request.

Prerequisites: complete Exercises B1 - B4.

Creating a mobile page template controller and view

- 1. In AlloyTraining, open ~\Controllers.
- 2. Copy and paste the StartPageController.cs file.
- 3. Rename the copy to StartPageMobileController.cs, and open it.
- 4. Import the EPiServer.Framework.Web namespace, as shown in the following code:

using EPiServer.Framework.Web;

5. Apply an attribute to ensure this controller is only used when the "mobile" rendering tag is set, as shown in the following code:

Episerver defines the RenderingTags class with a Mobile string constant value of "mobile".

- 6. In AlloyTraining, right-click Views, and add a new folder named StartPageMobile.
- 7. Expand ~\Views\StartPage, and copy the Index.cshtml file into the folder StartPageMobile.
- 8. Open ~\Views\StartPageMobile\Index.cshtml.
- 9. At the bottom of the view, delete the outputting of the **MainContentArea** using **PropertyFor**, and replace it with an enumeration of the filtered items in the content area that outputs a link to each page, as shown in the following markup:

FilteredItems removes any content references that the current visitor should not be able to see.

Creating a display channel

- 1. In AlloyTraining, right-click ~\Business, and add a new folder named DisplayChannels.
- 2. Right-click ~\Business\DisplayChannels, and add a new class named MobileDisplayChannel.
- 3. Import the EPiServer.Web namespace.
- 4. Make the class inherit from **DisplayChannel**.
- 5. Use Visual Studio to implement the abstract class, as shown in the following screenshot:

7 ⊟namespace AlloyTraining.Business.DisplayChannels					
or a public class MobileDisplayChannel : DisplayChannel					
Implement Abstract Class 12	CS0534 'MobileDisplayChannel' does not implement inherited abstract member 'DisplayChannel.IsActive(HttpContextBase)'				
-	<pre>/// { public override string ChannelName => throw new NotImplementedException(); public override bool IsActive(HttpContextBase context) { throw new NotImplementedException(); } } Preview changes Fix all occurrences in: Document Project Solution</pre>				

- 6. For the ChannelName property, return the mobile rendering tag.
- 7. For the IsActive method, return if the current request is from a mobile device.

Your completed class should look something like the following code:

```
using EPiServer.Framework.Web;
using EPiServer.Web;
using System.Web;
namespace AlloyTraining.Business.DisplayChannels
{
    public class MobileDisplayChannel : DisplayChannel
    {
        // C# 6.0 syntax for a read-only property
        public override string ChannelName => RenderingTags.Mobile;
        public override bool IsActive(HttpContextBase context)
        {
            return context.Request.Browser.IsMobileDevice;
        }
    }
}
```

Testing the display channel

- 1. Start the **AlloyTraining** website.
- 2. Press F12 to show developer tools.
- 3. In Chrome's developer tools pane, click **Toogle device toolbar**, or press *Ctrl* + *Shift* + *M*, and note that by default Chrome shows a Responsive page, as shown in the following screenshot:



4. In the device toolbar, choose **iPhone 5**, click **Refresh** or press *F5*, and note Chrome now shows the "mobile" page template response with links instead of partial content, as shown in the following screenshot:



	iPhone 5 🔻	320	×	568	100% 🔻	\bigcirc
	- Start - Alcyblant - AlcyPlan - AlcyPlan - AlcyTrack - Achat us - Login					
We This in the best Alloy Plan Alloy Track Alloy Track	Icome!					

- 5. Close the developer tools.
- 6. Log in as a CMS admin.
- 7. Edit the Start page.
- 8. In the toolbar, click **Toggle view settings**, and select the **mobile** channel, as shown in the following screenshot:



Improving the editors' preview

You can combine a display channel with some common resolutions, and a background image, but this only affects editors preview options. It has no affect at runtime for visitors.

- 1. In AlloyTraining, open ~\Business\DisplayChannels\MobileDisplayChannel.cs.
- 2. Define a class named **iPhone5** that implements the **IDisplayResolution** interface, and use Visual Studio to implement the interface, as shown in the following screenshot:



3. Return appropriate values for the four properties, as shown in the following code:

```
public class iPhone5 : IDisplayResolution
{
    public string Id => "iphone5";
```



```
public string Name => "iPhone 5 (320 x 568)";
public int Width => 320;
public int Height => 568;
}
```

4. Copy and paste the class for an **iPhone4** resolution, as shown in the following code:

```
public class iPhone4 : IDisplayResolution
{
    public string Id => "iphone4";
    public string Name => "iPhone 4 (320 x 480)";
    public int Width => 320;
    public int Height => 480;
}
```

5. In MobileDisplayChannel, override the ResolutionId property, as shown in the following code:

```
public override string ResolutionId => "iphone5";
```

In AlloyDemo, you will see that a base class has been used for the resolutions and strings are retrieved from resource files using LocalizationService. This is recommended. In this exercise, we have used a simplified example to highlight the main functionality of DisplayChannel and IDisplayResolution.

- 7. Right-click ~\Resources\LanguageFiles, and add a new XML file named DisplayChannels.xml.
- 8. Modify the file, as shown in the following markup:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<languages>
 <language name="English" id="en">
   <displaychannels>
      <displaychannel name="mobile">
        <name>Mobile</name>
      </displaychannel>
   </displaychannels>
   <resolutions>
      <iphone4>iPhone 4 (320 x 480)</iphone4>
      <iphone5>iPhone 5 (320 x 568)</iphone5>
   </resolutions>
 </language>
  <language name="Swedish" id="sv">
   <displaychannels>
      <displaychannel name="mobile">
        <name>Mobil</name>
      </displaychannel>
   </displaychannels>
   <resolutions>
      <iphone4>iPhone 4 (320 x 480)</iphone4>
      <iphone5>iPhone 5 (320 x 568)</iphone5>
   </resolutions>
  </language>
</languages>
```

Testing the improvements

1. Start the AlloyTraining website, and log in as a CMS admin.



2. In the page editing toolbar, click **Toogle view settings**, select **Mobile**, and note the default resolution is **iPhone 5**, which has a built-in background image simulating that phone, as shown in the following screenshot:



Display channels can only be associated with one resolution, and that association is one-way. This means that when the **Mobile** display channel is selected, the **iPhone 5 (320 x 568)** resolution will be selected by default. But other resolutions must be manually selected. And if an editor leaves the channel set to **Automatic**, and then selects a non-Automatic resolution, the **Mobile** channel will not be selected. For example, if the Editor chooses a combination of **Automatic** and **iPhone 5 (320 x 568)** they will not see the result of having the **Mobile** display channel active.

- 3. Select **iPhone 4**, and note that although it changes resolution, it does not have a background image.
- 4. Close the browser.
- 5. From the solution files, drag and drop the \cmsdevfun-exercisefiles\Module C\C5\ClientResources folder and the \cmsdevfun-exercisefiles\Module C\C5\module.config file into AlloyTraining, as shown in the following screenshot:



6. Start the **AlloyTraining** website, and log in as a CMS admin, and select the **iPhone 4** resolution, as shown in the following screenshot:



7. Close the browser.



Module D – Working with Blocks

Goal

The overall goal of the exercises in this module is to implement working examples of various block types and uses for blocks. You will:

- 1. Create a controller-less block for efficiency.
- 2. Create a block with a controller.
- 3. Create a preview template for blocks and partial pages.
- 4. Move important block properties to the basic info area.
- 5. Use a block type as a property type.

Exercise D1 – Creating a controller-less block for editorial content

In this exercise, you will create a block type with a block template that consists of a view without a controller.

Controller-less blocks are more efficient than block templates with controllers, so you should use them whenever possible.

Prerequisites: complete Exercises B1 to B4, C1 to C4.

```
If you have not completed Exercises C1 to C4, then you can just complete the task, Adding a content area to the Product page type and template in Exercise C4 on page 120, to define two content areas on a product page, and then you can complete exercise D1.
```

Creating an editorial block type

- In AlloyTraining, expand Models, right-click Blocks, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Block Type, enter EditorialBlock.cs for the Name, and click Add.
- 3. Change the **DisplayName** to **Editorial**.
- 4. Add a Description of "Use this for a rich editorial text that will be reused in multiple places."
- 5. Apply the [SiteBlockIcon] attribute to the class.
- 6. Add an XhtmlString property named MainBody, and allow it to be localized into multiple languages:

Your code should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Blocks
{
    [ContentType(DisplayName = "Editorial",
        GroupName = SiteGroupNames.Common,
        Description = "Use this for a rich editorial text that will be reused in
multiple places.")]
    [SiteBlockIcon]
    public class EditorialBlock : BlockData
    {
        [CultureSpecific]
        [Display(
```



```
Name = "Main body",
Description = "The main body will be shown in the main content area of the
page, using the XHTML-editor you can insert for example text, images and tables.",
GroupName = SystemTabNames.Content,
Order = 10)]
public virtual XhtmlString MainBody { get; set; }
}
}
```

Creating a controller-less block template

- 1. In AlloyTraining, right-click ~\Views\Shared, and choose Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter EditorialBlock.cshtml for the name, and click Add.
- 3. Modify the view to use EditorialBlock as its model and to output the MainBody property to support on-page editing, as shown in the following markup:

```
@model AlloyTraining.Models.Blocks.EditorialBlock
<div class="block span">
    @Html.PropertyFor(m => m.MainBody)
</div>
```

Creating an instance of the block and using it in multiple places

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- 2. In Assets pane, click Blocks.
- 3. Add a new folder named Editorials to the For This Site folder.
- 4. In the **Editorials** folder, add a new **Editorial** block named **Customer Quotes**, as shown in the following screenshot:



5. In the Main body write some quotes from Alloy's customers, as shown in the following screenshot:

For This Site > Editorials >	F Image: Constraint of the second s
Autosaved 8:18 AM Undo?	Products
+ Back This item is not used anywhere.	■ 📷 For This Site
Content Settings	Documents
Category Add one or more categories +	Editorials =-
Main body Main body Image: Style Image: Style<	E Customer Quotes
Cool, cool, cool Aoed Nadii	+ =- \$\$

6. Publish the block and note that it is not current used anywhere.



7. Edit the Alloy Plan page, and drag and drop the **Customer Quotes** block into its **Related content area** property, as shown in the following screenshot:

of	Changes to be published Publish? V	Blocks Media Q Search Image: Search Image: Search	\supset
	Alloy Plan Project planning, Reporting and statistics, Email nandling of tasks, Risk calculations, Direct communication to members	Editorials Products For This Page	≡*
	"Alloy Plan is great!" - The Queen of England You can drop content here, or Custon	E Customer Quotes	E∗
	<u>create a new block</u>	+ =-	¢-

- 8. Publish the page.
- 9. Edit the Alloy Track page and drag and drop the Customer Quotes block into its Related content area property.
- 10. Publish the page.
- 11. Edit the **About us** page, and drag and drop the **Customer Quotes** block into its **Main body** property, as shown in the following screenshot:

Main body	📚 👾 🖃 🕪 🛤 🗅 🐃 🍙 國
	B I I II Styles ▼ ∽ ∼ Q
	Alloy improves the effectiveness of project teams by putting the proper tools in your hands. Communication is made easy and inexpensive, no matter where team members are located.
	Path: p

- 12. Publish the page.
- 13. In the **Assets** pane, double-click **Customer Quotes**, and note the block now shows that it is referenced on three items, as shown in the following screenshot:



14. Click **3** items, and note it shows the pages that reference the shared block, as shown in the following screenshot:

Item	×
is used in the following places.	
	😂 Refresh list
Start	View
Start	View
Start	View
	Item is used in the following places. Start Start Start Start

15. Close the browser.



Exercise D2 - Creating a block with a controller for teaser content

In this exercise, you will create a block type with a block template that is a view with a controller.

Blocks with controllers are more resource hungry, so you should only use them when necessary. For example, when content stored in the CMS must be combined with data from a web service, or a calculated value. In this example, we will generate a random number for a visitor count that will be shown in the block.

Prerequisites: complete Exercises B1 to B4, C1 to C4.

If you have not completed Exercises C1 to C4, then you can just complete the task, Adding a content area to the Start page type and template in Exercise C1 on page 106, to define two content areas on a product page, and then you can complete exercise D2.

Creating a teaser block type

- 1. In AlloyTraining, expand Models, right-click Blocks, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Block Type, enter TeaserBlock.cs for the Name, and click Add.
- 3. Change the **DisplayName** to **Teaser**.
- 4. Add a **Description** of "Use this for rich text with heading, image and page link that will be reused in multiple places."
- 5. Apply the **[SiteBlockIcon]** attribute to the class.
- 6. Add the following four properties with appropriate attributes:
 - **TeaserHeading:** string (with support for multiple languages)
 - TeaserText: XhtmlString (with support for multiple languages)
 - TeaserImage: ContentReference (images only)
 - TeaserLink: PageReference

Your code should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using EPiServer.Web;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Blocks
{
    [ContentType(DisplayName = "Teaser",
        GroupName = SiteGroupNames.Common,
        Description = "Use this for rich text with heading, image and page link
that will be reused in multiple places.")]
    [SiteBlockIcon]
    public class TeaserBlock : BlockData
    {
        [CultureSpecific]
        [Display(Name = "Heading", Order = 10)]
        public virtual string TeaserHeading { get; set; }
        [CultureSpecific]
        [Display(Name = "Rich text", Order = 20)]
        public virtual XhtmlString TeaserText { get; set; }
        [Display(Name = "Image", Order = 30)]
        [UIHint(UIHint.Image)]
```



```
public virtual ContentReference TeaserImage { get; set; }
    [Display(Name = "Link", Order = 40)]
    public virtual PageReference TeaserLink { get; set; }
  }
}
```

Creating a teaser view model, controller, and view

For complex content types, its content template is often a combination of: a controller, a view model, and a view.

- 1. In AlloyTraining, right-click ViewModels, and click Add | New Item..., or press Ctrl + Shift + A.
- In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter TeaserBlockViewModel.cs for the Name, and click Add.
- 3. Modify the view model to have two properties: **CurrentBlock** and **VisitorCount**, as shown in the following code:

```
using AlloyTraining.Models.Blocks;
namespace AlloyTraining.Models.ViewModels
{
    public class TeaserBlockViewModel
    {
        public TeaserBlock CurrentBlock { get; set; }
        public int TodaysVisitorCount { get; set; }
    }
}
```

- 4. In AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press Ctrl + Shift + A.
- 5. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Block Controller (MVC), enter TeaserBlockController.cs for the Name, and click Add.
- 6. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Blocks** namespace.
- 7. Modify the class to create an instance of the teaser view model and set its properties, before passing it to a partial view, as shown in the following code:

```
using AlloyTraining.Models.Blocks;
using AlloyTraining.Models.ViewModels;
using EPiServer.Web.Mvc;
using System;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class TeaserBlockController : BlockController<TeaserBlock>
    {
        public override ActionResult Index(TeaserBlock currentBlock)
        {
            var viewmodel = new TeaserBlockViewModel
            {
                CurrentBlock = currentBlock,
                TodaysVisitorCount = (new Random()).Next(300, 900)
            };
            return PartialView(viewmodel);
        }
    }
}
```



ullet We have simulated some data for the visitor count using the Random class.

- 8. In AlloyTraining, right-click Views, and add a new folder named TeaserBlock.
- 9. Right-click TeaserBlock, and choose Add | New Item..., or press Ctrl + Shift + A.
- 10. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.
- 11. Modify the view, as shown in the following markup:

Creating an instance of the teaser block

- 1. Start the **AlloyTraining** website, and log in as a CMS admin.
- 2. In Assets pane, click Media.
- 3. Add a new folder named People to the For This Site folder.
- 4. Upload the three images of people in the ~\Assets\People folder, as shown in the following screenshot:



- 5. In Assets pane, click Blocks.
- 6. Add a new folder named **Teasers** to the **For This Site** folder.



7. Add a new **Teaser** block to the **Teasers** folder named **Alloy Meet Customer Testimonial**, as shown in the following screenshot:

New Block: For This Site > Teasers Name eet Customer Testimonia	
Other Block Types	
EDUCATION SERVICES	Teaser Use this for rich text with heading, image and page link that will be reused in multiple places.

- 8. Add values for the properties, as shown in the following screenshot:
 - a. Heading: Sharing Worldwide
 - b. Rich text: "Alloy Meet is a highly effective e-solution for our operations. Improved business metrics are realized through the cross-departmental sharing of information worldwide." John Randle, HighTec Inc
 - c. Image: JohnRandle.jpg (use the picker or drag and drop from Assets pane)
 - d. Link: Alloy Meet (use the picker or drag and drop from Navigation pane)

For This Site > Teasers > Alloy Meet Customer Testimonial				
		Autosaved 6:20 AM Undo?		
+ Back This it	em is not us	sed anywhere.		
Content	Settings			
Category		Add one or more categories +		
Heading		Sharing Worldwide		
Rich text		B I IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII		
Image		JohnRandle.jpg 😵		
Link		Alloy Meet 😵		

By default, blocks do not support on-page editing, so content editors must use All Properties view. In the next exercise, you will create a preview template for blocks to enable an on-page editing experience.

9. Publish the block.



10. Edit the **Start** page, drag and drop the **Alloy Meet Customer Testimonial** to the top of its main content area, and publish the page, as shown in the following screenshot:



This implementation of the teaser block is okay, but it has two limitations: (1) the context menu shows the display options that you defined for product pages, but the teaser block ignores the selection a content editor might make, and (2) editors must use **All Properties** view to change its properties. In the next exercise, you will fix both limitiations.



Exercise D3 – Creating a preview renderer for partial pages and shared blocks

In this exercise, you will create preview renderer for the teaser block.

The renderer will give previews of how the block will look when rendered with various templates. Prerequisites: complete Exercises B1 - B4, C1 - C4, D2.

Creating three teaser block templates for full, wide, and narrow

1. In AlloyTraining, open TeaserBlockController.cs.

{

2. Copy and paste the statements that define the **TeaserBlockController** class so that you have three controllers, and apply TemplateDescriptor to make the template resolver select them when the tags: "full", "wide", and "narrow" have been applied, as shown in the following code:

```
using AlloyTraining.Models.Blocks;
using AlloyTraining.Models.ViewModels;
using EPiServer.Framework.DataAnnotations;
using EPiServer.Web.Mvc;
using System;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
    [TemplateDescriptor(Tags = new[] { SiteTags.Full },
        AvailableWithoutTag = true)]
    public class TeaserBlockController : BlockController<TeaserBlock>
    {
        public override ActionResult Index(TeaserBlock currentBlock)
        {
            var viewmodel = new TeaserBlockViewModel
            {
                CurrentBlock = currentBlock,
                TodaysVisitorCount = (new Random()).Next(300, 900)
            };
            return PartialView(viewmodel);
        }
    }
    [TemplateDescriptor(Tags = new[] { SiteTags.Wide })]
    public class TeaserBlockWideController : BlockController<TeaserBlock>
    {
        public override ActionResult Index(TeaserBlock currentBlock)
        {
            var viewmodel = new TeaserBlockViewModel
            {
                CurrentBlock = currentBlock,
                TodaysVisitorCount = (new Random()).Next(300, 900)
            };
            return PartialView(viewmodel);
        }
    }
    [TemplateDescriptor(Tags = new[] { SiteTags.Narrow })]
    public class TeaserBlockNarrowController : BlockController<TeaserBlock>
        public override ActionResult Index(TeaserBlock currentBlock)
        ł
            var viewmodel = new TeaserBlockViewModel
            {
```



```
CurrentBlock = currentBlock,
TodaysVisitorCount = (new Random()).Next(300, 900)
};
return PartialView(viewmodel);
}
}
```

- 3. In **Views** folder, copy and paste the **TeaserBlock** folder twice to create two copies named: **TeaserBlockWide** and **TeaserBlockNarrow**.
- 4. Open ~\Views\TeaserBlockNarrow\Index.cshtml.
- 5. Add class **span4** to the outer <div>, and delete the <div> that outputs the image.
- 6. Open ~\Views\TeaserBlockWide\Index.cshtml.
- 7. Add class **span8** to the outer <div>.

Creating a preview view model, controller, and view

When a content editor edits the teaser block, we would like them to see a preview of what the block would look like if any of the three display options are selected. To do this, we need to simulate a page that has a property that contains an instance of a block. We can do this by defining a view model with a ContentArea and add the instance to the area. Then on the simulated page, we can easily output the block by calling PropertyFor three times, once for each display option.

- 1. In AlloyTraining, expand Models, right-click ViewModels, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter PreviewPageViewModel.cs for the name, and click Add.
- 3. Modify the contents, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using EPiServer.Core;
namespace AlloyTraining.Models.ViewModels
{
    public class PreviewPageViewModel : PageViewModel<SitePageData>
    {
        public PreviewPageViewModel(SitePageData currentPage,
            IContent contentToPreview) : base(currentPage)
        {
            this.PreviewArea = new ContentArea();
            this.PreviewArea.Items.Add(new ContentAreaItem
                { ContentLink = contentToPreview.ContentLink });
        }
        public ContentArea PreviewArea { get; set; }
    }
}
```

- 4. In **AlloyTraining**, expand **Controllers**, and right-click and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 5. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter PreviewPageController.cs for the Name, and click Add.
- 6. Modify its contents, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using EPiServer.Core;
using EPiServer.Framework.DataAnnotations;
using EPiServer.Framework.Web;
```

Copyright © Episerver AB. All rights reserved.



```
using EPiServer.ServiceLocation;
using EPiServer.Web;
using EPiServer.Web.Mvc;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    [TemplateDescriptor(Inherited = true,
        TemplateTypeCategory = TemplateTypeCategories.MvcController,
        Tags = new[] { RenderingTags.Preview },
        AvailableWithoutTag = false)]
    public class PreviewPageController :
        ActionControllerBase, IRenderTemplate<BlockData>
    {
        public ActionResult Index(IContent currentContent)
        {
            var loader = ServiceLocator.Current
                .GetInstance<EPiServer.IContentLoader>();
            var startPage = loader.Get<SitePageData>
                (ContentReference.StartPage);
            var viewmodel = new PreviewPageViewModel(
                startPage, currentContent);
            return View(viewmodel);
        }
    }
}
```

Our site enforces a rule for view models passed to layouts: they must have a CurrentPage property with a page instance in it. For our preview, we can get the Start page and pass that in. The preview won't use it, but some page must be passed in.

- 7. In AlloyTraining, right-click Views, and add a new folder named PreviewPage.
- 8. Right-click PreviewPage, and choose Add | New Item..., or press Ctrl + Shift + A.
- 9. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.
- 10. Modify the view, as shown in the following markup:

```
@using AlloyTraining
@using AlloyTraining.Models.ViewModels
Qusing EPiServer.Web.Mvc.Html
@model PreviewPageViewModel
<mark>@{</mark>
              Layout = null;
}
<head>
              <title>@(Model.CurrentPage.MetaTitle ?? Model.CurrentPage.Name)</title>
              <link rel="stylesheet" href="@Url.Content("~/Static/css/bootstrap.css")" />
              <link rel="stylesheet"</pre>
                                   href="@Url.Content("~/Static/css/bootstrap-responsive.css")" />
              <link rel="stylesheet" href="@Url.Content("~/Static/css/media.css")" />
              <link rel="stylesheet" href="@Url.Content("~/Static/css/style.css")" />
              <link rel="stylesheet" href="@Url.Content("~/Static/css/editmode.css")" />
              <script type="text/javascript"</pre>
                                          src="@Url.Content("~/Static/js/jquery.js")"></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script></script
              <script type="text/javascript"</pre>
                                          src="@Url.Content("~/Static/js/bootstrap.js")"></script>
</head>
 <body>
              <div class="container">
                            <div class="well well-large">
```



```
<div class="row">
                <div class="alert alert-info">Display Option: Full (default)</div>
            </div>
            <div class="row">
                @Html.PropertyFor(m => m.PreviewArea, new { Tag = SiteTags.Full })
            </div>
        </div>
        <div class="well well-large">
            <div class="row">
                <div class="alert alert-info span8">Display Option: Wide</div>
            </div>
            <div class="row">
                @Html.PropertyFor(m => m.PreviewArea, new { Tag = SiteTags.Wide })
            </div>
        </div>
        <div class="well well-large">
            <div class="row">
                <div class="alert alert-info span4">Display Option: Narrow</div>
            </div>
            <div class="row">
                @Html.PropertyFor(m => m.PreviewArea, new { Tag = SiteTags.Narrow })
            </div>
        </div>
    </div>
</body>
```

Testing the preview

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- Edit the Alloy Meet Customer Testimonial teaser block, and note that instead of only allowing All Properties view, it defaults to using an On-Page Editing view, and shows a preview of the Full (default) display option, as shown in the following screenshot:





3. Scroll down the preview page to see the Wide display option, as shown in the following screenshot:



4. Scroll down the preview page to see the **Narrow** display option, as shown in the following screenshot:



5. Close the browser.



Exercise D4 – Moving properties to the basic info area

In this exercise, you will move a property from the tabbed area up to the basic info properties area. **Prerequisites:** complete Exercises B1 – B4, C1 – C4, D2.

Understanding the existing basic info area

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- Edit the Start page. Note the on-page editing view contains several properties which are reached by scrolling beyond the top of the page to reveal the top gray area. These are called basic info properties and can be used to:
 - Give the page a simple address.
 - Set access rights for a page.
 - Change the name in the URL, and so on.

This basic info area is always displayed in the **All Properties** editing view, as shown in the following screenshot:

Name	Start	Visible to	Everyone Manage
Name in URL	start <u>Change</u>	Languages	en
Simple address	Change	ID, Type	5, Start Page
	Display in navigation		Tools 🗸

3. Edit the **Alloy Meet Customer Testimonial** teaser and view the basic information area. Note for a block, like **TeaserBlock**, it cannot have a URL, a simple address, or display in navigation, so there is empty space available that we can make better use of, as shown in the following screenshot:

For This Site > Teasers > Alloy Meet Customer Testimonial						
← <u>Back</u> Changes made here will affect at least <u>1 item</u>						
Name	Alloy Meet Customer Test	Visible to Languago ID, Type	es en 28, Teaser Tools 🗸			

Making use of space in the basic information area for blocks

- 1. In AlloyTraining, open ~/Models/Blocks/TeaserBlock.cs.
- 2. Modify the [Display] attribute for the TeaserHeading, TeaserImage, and TeaserLink properties to set the GroupName to PageHeader, as shown in the following code:

```
[Display(Name = "Link", Order = 40,
GroupName = SystemTabNames.PageHeader)]
public virtual PageReference TeaserLink { get; set; }
```

The "magic string" for SystemTabNames.PageHeader is "EPiServerCMS_SettingsPanel".

- 3. Start the AlloyTraining website, and log in as a CMS admin.
- 4. Edit the **Start** page.



5. In the main content area, edit the **Alloy Meet Customer Testimonial** teaser, switch to **All Properties** view, and note the three properties are now in the basic information area, as shown in the following screenshot:

Back Changes mad	le here will affect at least <u>1 item</u>		
Name	Alloy Meet Customer Test	Visible to	Everyone Manage
Heading	Sharing Worldwide	Languages	en
Image	JohnRandle.jpg 🛛 🚷	ID, Type	19, Teaser
Link	Alloy Meet 🔕		Tools 🗸
Content Setti	ngs		
Category	Add one or more categories +		
Rich text	Image: Styles Image: Styles Image: Styles		



Exercise D5 – Using a block as a content property type

In this exercise, you will define and use an employee block type as a property type.

Prerequisites: complete Exercises B1 to B4.

Using a block as a property type

1. In AlloyTraining, add an EmployeeBlock block type for storing information about employees: FirstName, LastName, and HireDate, as shown in the following code:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Blocks
{
    [ContentType(DisplayName = "Employee",
        GroupName = SiteGroupNames.Specialized,
        Order = 10,
        Description = "Use this to store information about an employee.")]
    [SiteBlockIcon]
    public class EmployeeBlock : BlockData
    {
        [Display(Name = "First name",
            GroupName = SystemTabNames.Content,
            Order = 10)]
        public virtual string FirstName { get; set; }
        [Display(Name = "Last name",
            GroupName = SystemTabNames.Content,
            Order = 20)
        public virtual string LastName { get; set; }
        [Display(Name = "Hire date",
            GroupName = SystemTabNames.Content,
            0rder = 30)]
        public virtual DateTime? HireDate { get; set; }
    }
}
```

- 4. In AlloyTraining, right-click ~\Views\Shared, and choose Add | New Item..., or press Ctrl + Shift + A.
- 5. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter EmployeeBlock.cshtml for the Name, and click Add.
- 6. Modify the view to use **EmployeeBlock** as its model and to output the **FirstName**, **LastName**, and **HireDate** properties, as shown in the following markup:

```
@model AlloyTraining.Models.Blocks.EmployeeBlock
<div class="block span">
    @Model.FirstName @Model.LastName
    @(Model.HireDate.HasValue ? "was hired on " +
        Model.HireDate.Value.ToString("dddd, d MMMM yyyy") : "")
</div>
```

- 7. In AlloyTraining, open ~\Models\Pages\StandardPage.cs.
- 8. Import the blocks namespace, as shown in the following code:

```
using AlloyTraining.Models.Blocks;
```

9. Add a property, as shown in the following code:



public virtual EmployeeBlock Author { get; set; }

- 10. In AlloyTraining, open ~\Views\StandardPage\Index.cshtml.
- 11. At the bottom of the view, render the Author property with support for on-page editing:
 @Html.PropertyFor(m => m.CurrentPage.Author)

Testing the block type property

- 1. Start the **AlloyTraining** website, and log in as a CMS admin.
- 2. Edit the About us page, switch to All Properties view, and click the Content tab.
- 3. Modify the Author property's three properties, as shown in the following screenshot:

Start >	
Name	About us
Name in URL	about-us Change
Simple address	Change
	✓ Display in navigation
_	
SEO Content	Settings
Category	Add one or more categories +
Author	
First name	Johan
Last name	Johansson
Hire date	6/13/2012, 12:00 AM 🔻

- 4. Publish the page.
- 5. Switch to **Live** view and note the author's details are rendered onto the page.
- 6. Close the browser.


Module E – Navigating Content

Goal

The overall goal of the exercises in this module is to implement various ways for a visitor to navigate a website. You will:

- 1. Create a child page listing block.
- 2. Use the child page listing block in a page for navigating news articles.
- 3. Create a submenu for navigation.
- 4. Create a search page for visitors and implement it using Episerver Search or Episerver Find.
- 5. Add a search box to the top navigation menu.

Exercise E1 – Creating a page listing block

In this exercise, you will create a new block type containing a listing of child pages.

The block will have two properties; a heading and a page picker to select a parent page to show a list of its children.

Prerequisites: complete Exercises B1 to B4.

Although Exercise D3 – Creating a preview renderer for partial pages and blocks is not a prerequisite, if you have not completed that exercise then your blocks will not have On-Page Editing (OPE) experience.

Creating a page listing block type

- 1. In AlloyTraining, expand Models, right-click Blocks, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Block Type, enter ListingBlock.cs for the name, and click Add.
- 3. Change the DisplayName to Listing.
- 4. Add a Description of "Choose a page in the tree, and its children will be listed, with a heading."
- 5. Apply the [SiteBlockIcon] attribute to the class.
- 6. Add the following properties with appropriate attributes:
 - a. Heading: string
 - b. ShowChildrenOfThisPage: PageReference

Your code should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Blocks
{
    [ContentType(DisplayName = "Listing",
        GroupName = SiteGroupNames.Common,
        Description = "Choose a page in the tree, and its children will be
listed, with a heading.")]
    [SiteBlockIcon]
    public class ListingBlock : BlockData
    {
```

```
[Display(Name = "Heading", Order = 10)]
public virtual string Heading { get; set; }
[Display(Name = "Show children of this page", Order = 20)]
public virtual PageReference ShowChildrenOfThisPage { get; set; }
}
}
```

Creating a page listing view model, controller, and view

- 1. In AlloyTraining, right-click ViewModels, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter ListingBlockViewModel.cs for the name, and click Add.
- 3. Add two properties: Heading and Pages, as shown in the following code:

```
using EPiServer.Core;
using System.Collections.Generic;
namespace AlloyTraining.Models.ViewModels
{
    public class ListingBlockViewModel
    {
        public string Heading { get; set; }
        public IEnumerable<PageData> Pages { get; set; }
    }
}
```

- 4. In **AlloyTraining**, expand **Controllers**, and right-click and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 5. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Block Controller (MVC), enter ListingBlockController.cs for the Name, and click Add.
- 6. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Blocks** namespace.
- 7. Modify the class to create an instance of the listing block view model and set its properties, before passing it to a partial view, as shown in the following code:

```
using AlloyTraining.Models.Blocks;
using AlloyTraining.Models.ViewModels;
using EPiServer;
using EPiServer.Core;
using EPiServer.Filters;
using EPiServer.Web.Mvc;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class ListingBlockController : BlockController<ListingBlock>
    {
        private readonly IContentLoader loader;
        public ListingBlockController(IContentLoader loader)
        {
            this.loader = loader;
        }
        public override ActionResult Index(ListingBlock currentBlock)
        {
            var viewmodel = new ListingBlockViewModel
```

}



```
{
            Heading = currentBlock.Heading
        };
        if (currentBlock.ShowChildrenOfThisPage != null)
        {
            IEnumerable<PageData> children = loader.GetChildren<PageData>(
                currentBlock.ShowChildrenOfThisPage);
            // Remove pages:
            // 1. that are not published
            // 2. that the visitor does not have Read access to
            // 3. that do not have a page template
            IEnumerable<IContent> filteredChildren =
                FilterForVisitor.Filter(children);
            // 4. that do not have "Display in navigation" selected
            viewmodel.Pages = filteredChildren.Cast<PageData>()
                .Where(page => page.VisibleInMenu);
        }
        return PartialView(viewmodel);
    }
}
```

- 8. In AlloyTraining, right-click Views, and add a new folder named ListingBlock.
- 9. Right-click ListingBlock, and choose Add | New Item..., or press Ctrl + Shift + A.
- 10. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.

11. Modify the view, as shown in the following markup, and note the following:

- When there are no pages to show, we render alert messages visible to content editors.
- PageData does not have a strongly-typed MainBody property, but the page it references might, so we can use the Property property to check if the current page in the listing has a MainBody and render it if it does.

```
@using EPiServer.Core
@model AlloyTraining.Models.ViewModels.ListingBlockViewModel
@if (Model.Pages == null)
{
    if (EPiServer.Editor.PageEditing.PageIsInEditMode)
    {
        <div class="label label-warning">Set the ShowChildrenOfThisPage property to a
page.</div>
    }
}
else
{
    <h2 @Html.EditAttributes(x => x.Heading)>@Model.Heading</h2>
    if (Model.Pages.Count() == 0)
    {
        <div class="label label-warning">The page selected has no children.</div>
    }
    foreach (PageData page in Model.Pages)
    {
        <div class="listresult theme1">
            <h3>@Html.ContentLink(page.ContentLink)</h3>
            @if (page.StartPublish.HasValue)
            {
                Published on
```



New Block: For This Site

Common

Name Children of Alloy Meet

Settings

Category

Heading

page

Show children of this

Listing

a heading.

Choose a page in the tree, and its children will be listed, with

Add one or more categories

Alloy Meet's child pages

Alloy Meet

+

⊗

Testing the page listing block

- 1. Start the **AlloyTraining** website, and log in as a CMS admin.
- 2. In Assets pane, click Blocks.
- 3. Add a new Listing block to the For This Site folder named Children of Alloy Meet, as shown in the screenshot:
- 4. Switch to **On-Page Editing** view, and note the warning to editors, as shown in the following screenshot:
- You must have completed Exercise D3 Creating a preview renderer for partial pages and shared blocks on page 135 to have the On-Page Editing experience.



- 5. Switch to **All Properties** view, and set the properties, as shown in the screenshot:
 - a. Heading: Alloy Meet's child pages
 - b. ShowChildrenOfThisPage: Alloy Meet
- 6. Switch to **On-Page Editing** view, and note the warning to editors, as shown in the following screenshot:

╘ + ⊙ Q ₽		A 🖬
For This Site > Children of Alloy Meet		Publish? 🗸 📘 📰
	Autosaved 8:00 AM Undo?	
+ Back This item is not used anywhe	re.	×
	Display Option: Full (default)	
Alloy Meet's child The page selected has no children.	l pages	

7. Publish the block.



8. Edit the **Start** page, and drag and drop **Children of Alloy Meet** into the top of its main content area, as shown in the following screenshot:

G-1	
Autosaved 5:14 PM <u>Undo?</u>	
Path: p	•
Children of Alloy Meet	
Alloy Meet Customer Testimonial	
	Autosaved 5:14 PM Undo?

- 9. Publish the page.
- 10. Add some standard pages underneath **Alloy Meet** in the page tree, named **Alpha**, **Beta**, and **Gamma**, as shown in the screenshot:
- 11. For each page, set their **Main body** properties using a Lorem Ipsum generator.
- Lorem Ipsum is dummy text of the printing and typesetting industry. Learn about and generate some at the following link: <u>https://www.lipsum.com/</u>

B Root	
🖬 🏠 Start	
Alloy Meet	
📑 Alpha	≣*
🕒 Beta	
🕒 Gamma	
🗋 Alloy Plan	
🗋 Alloy Track	
🗋 About us	

- 12. Switch to Live view.
- 13. View the Start page, and note the output, as shown in the following screenshot:

□ Start	×	Mark	-		×
< → C ∆	localhost:49353/en/			☆	:
			Æ	pr ·	
Alloy I	Meet's child pages				
Alpha Published on We	idnesday, 11 October 2017				
This is the A	lipha page.				
Beta Published on We This is the B	idnesday, 11 October 2017 Ieta page.				
Gamma Published on We	kinesday, 11 October 2017				
This is the G	amma page.				
					•

14. Close the browser.

You have now implemented a simple listing block. In the next exercise, you will use it to implement a news page that lists articles.



Exercise E2 - Creating a news landing page

In this exercise, you will create a news landing page type that lists news articles beneath it in the page tree.

You will use the standard page type as a base for the news landing page, you will use the page listing block from the previous exercise to add a news listing function to the news landing page, and you will create instances of the standard page under the news landing page to be its child news articles.

Prerequisites: complete Exercises B1 - B4, E1.

Creating a news landing page type

- In AlloyTraining, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Type, enter NewsLandingPage.cs for the Name, and click Add.
- 3. Modify the class to inherit from **StandardPage**.
- 4. Change the DisplayName to News Landing.
- 5. Add a **Description** of "Use this as a landing page for a list of news articles."
- 6. Put the page in the **News** group.
- 7. Apply the [SitePagelcon] attribute to the class.
- 8. Delete the **MainBody** property.
- 9. Add a ListingBlock property named NewsListing with appropriate attributes.

It would be nice if we could override the SetDefaultValues property and set the NewsListing property's Heading to the Name of the page and the Parent to reference this page, i.e. the news landing page will show a listing of its child pages, but unfortunately, the Name and PageLink properties are empty during SetDefaultValues. You could use an initialization module and listen for the CreatedContent event to set these properties automatically instead.

Your code should look something like the following:

```
using AlloyTraining.Models.Blocks;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.ComponentModel.DataAnnotations;
namespace AlloyTraining.Models.Pages
{
    [ContentType(DisplayName = "News Landing",
        GroupName = SiteGroupNames.News,
        Description = "Use this as a landing page for a list of news
articles.")]
    [SitePageIcon]
    public class NewsLandingPage : StandardPage
        [Display(Name = "News listing", Order = 315)]
        public virtual ListingBlock NewsListing { get; set; }
    }
}
```

Creating a news landing page template

 In AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press Ctrl + Shift + A.



- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter NewsLandingPageController.cs for the Name, and click Add.
- 3. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Pages** namespace.
- 4. Modify the class to derive from PageControllerBase<T>, and the Index action method to use a view model, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class NewsLandingPageController
        : PageControllerBase<NewsLandingPage>
    {
        public NewsLandingPageController(IContentLoader loader) : base(loader)
        {
        }
        public ActionResult Index(NewsLandingPage currentPage)
            return View(CreatePageViewModel(currentPage));
        }
    }
}
```

- 5. In AlloyTraining, right-click Views, and add a new folder named NewsLandingPage.
- 6. Right-click **NewsLandingPage**, and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 7. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.
- 8. Modify the view, as shown in the following markup:

```
@using AlloyTraining.Models.ViewModels
@using AlloyTraining.Models.Pages
@model PageViewModel<NewsLandingPage>
<mark>@{</mark>
   Layout = "~/Views/Shared/Layouts/_LeftNavigation.cshtml";
}
<h1 @Html.EditAttributes(m => m.CurrentPage.MetaTitle)>
   @(Model.CurrentPage.MetaTitle ?? Model.CurrentPage.Name)
</h1>
@Html.PropertyFor(m => m.CurrentPage.MetaDescription)
<div class="row">
    <div class="span8">
        @Html.PropertyFor(m => m.CurrentPage.MainBody)
        @Html.PropertyFor(m => m.CurrentPage.NewsListing)
    </div>
</div>
```

Testing the news landing page

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- 2. Under About us, add a new News Landing page named News & Events.
- 3. In On-Page Editing view, click the News listing property.



4. Set **Heading** to **The latest news for you**, and **Show children of this page** to reference the **News & Events** page, as shown in the following screenshot:

\$ ₹	
V Pages Sites Tasks Project Items	
Q Search	
B Root	Start > About us > C News & Events
and Start	Autosaved 8:29 AM
Alloy Meet	Start
🕒 Alloy Plan	Alloy Meet
Alloy Track	Alloy Plan Alloy Track
About us	News listing × tus
🕒 News & Events 🛛 🔔 🗶 🚍	A
	Heading The latest news for you Show children of this News & Events
	The page selected has no children.
+ =- \$	

- 5. Publish the page.
- 6. Under News & Events, add two Standard pages named Alloy Saves Bears and Join Us at Our Customer Event, and publish them.
- 7. View the **News & Events** page as a visitor and note the two news articles listed on the page.
- 8. Close the browser.



Exercise E3 – Improving navigation menus

In this exercise, you will replace the simple menu from an earlier exercise with a better-looking menu using an extension method named MenuList.

The basics of what this **MenuList** helper does is that it allows you to define a markup template for each item in your menu and it does filtering to remove unwanted items (for example, pages with no renderer can be filtered out).



Prerequisites: complete Exercises B1 to B4.

Improving the top navigation menu

- 1. In AlloyTraining, open ~\Views\Shared_NavigationMenu.cshtml.
- 2. Modify the view, as shown in the following markup:

To save time, and avoid line breaks, drag and drop or copy and paste the file from the exercise files ZIP.

```
Qusing EPiServer.Core
@using AlloyTraining.Business.ExtensionMethods
@using AlloyTraining.Models.ViewModels
@using AlloyTraining.Models.Pages
@model IPageViewModel<SitePageData>
<div class="alloyMenu">
   <div class="navbar">
       <div class="navbar-inner">
           <div class="container">
               <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-
collapse">
                   <span class="icon-bar"></span>
                   <span class="icon-bar"></span>
                   <span class="icon-bar"></span>
               \langle a \rangle
               <div class="nav-collapse">
                   <li
class="@(Model.CurrentPage.ContentLink.CompareToIgnoreWorkID(ContentReference.StartPag
e) ? "active" : null)">
                          @Html.ContentLink(ContentReference.StartPage)
                       @Html.MenuList(ContentReference.StartPage,
                   @
                       @Html.PageLink(item.Page)
                   )
                       @if (User.Identity.IsAuthenticated)
                          {
                              <a href="/en/logout">Log out @User.Identity.Name</a>
                          }
                          else
                          {
                              <a
```

href="@FormsAuthentication.LoginUrl?ReturnUrl=@Model.CurrentPage.PageLink.ExternalURLF
romReference()">Log in

```
</div>
          </div>
      </div>
   </div>
</div>
```

Adding a left navigation submenu

1. In AlloyTraining, right-click ~\Views\Shared, and choose Add | New Item..., or press Ctrl + Shift + A.

Make sure you add the file to ~\Views\Shared, and NOT to ~\Views\Shared\Layouts.

- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter _LeftNavigationMenu.cshtml for the Name, and click Add.
- 3. Modify the view, as shown in the following markup:

}

```
@using AlloyTraining.Models.ViewModels
@using AlloyTraining.Models.Pages
@using AlloyTraining.Business.ExtensionMethods
@model IPageViewModel<SitePageData>
@helper ItemTemplate(MenuItem firstLevelItem)
{
    <div class="accordion-heading">
        <a href="@Url.ContentUrl(firstLevelItem.Page.PageLink)"</pre>
class="@(firstLevelItem.Page.ContentLink.CompareToIgnoreWorkID(Model.CurrentPage.Conte
ntLink) ? "accordion-toggle active" : "accordion-toggle")" data-parent="#alloyDrop">
            @firstLevelItem.Page.PageName
            <i class="@(firstLevelItem.HasChildren.Value ? "icon-chevron-down right" :</pre>
"right")"></i>
        \langle a \rangle
    </div>
    <div id="collapse-@firstLevelItem.Page.ContentLink.ID" class="accordion-body</pre>
collapse @(firstLevelItem.Selected ? "in" : "")">
        @Html.MenuList(firstLevelItem.Page.ContentLink, SubLevelItemTemplate)
        </div>
}
@helper SubLevelItemTemplate(MenuItem subLevelItem)
{
    @Html.PageLink(subLevelItem.Page)
        @*To show more levels call Html.MenuList recursively here if
subLevelItem.Selected == true*@
    }
<div id="alloyDrop" class="accordion">
    <div class="accordion-group">
       @if (Model.Section != null)
            @Html.MenuList(Model.Section.ContentLink, ItemTemplate)
        }
    </div>
</div>
```

- 4. Open ~\Views\Shared\Layouts_LeftNavigation.cshtml.
- Add a statement to render the _LeftNavigationMenu, inside <div class="span4">, as shown in the following markup:



@Html.Partial("_LeftNavigationMenu", Model)

6. Start **AlloyTraining** website, and navigate to **About us** | **News & Events**, and note the top and left navigation menus, as shown in the following screenshot:



7. Close the browser.



Exercise E4 – Creating a search page for visitors

In this exercise, you will add a search page to the AlloyTraining site, allowing visitors to perform free-text searches on pages using either Episerver Find or Episerver Search.

If you drag and drop from the exercise files solution, then be careful not to add *both* controllers because they will conflict! The solution files named SearchPageControllerUsingEpiserverFind.cs and SearchPageControllerUsingEpiserverSearch.cs.

Prerequisites: complete Exercises B1 to B4.

The search page will be implemented with either Episerver Find (choose this if you will use DXC Service or you have paid for an Episerver Find license) or Episerver Search. To use Episerver Search, skip ahead to Creating a search page type on page 158, otherwise complete the following tasks.

Signing up for a free Episerver Find index

- 1. Open a web browser and navigate to http://find.episerver.com/
- 2. Click Sign up for a demo index or Sign Up in the top right corner.
- 3. Fill in the required information and click **Register**.
- 4. When the verification email arrives, copy and paste the link into your browser's address box and press ENTER to confirm your address and activate the index.

Verify your email for find.episerver.com					
no-reply@episerver.com Today, 09:17 You ¥	S Reply ∨				
Please verify your email address					
Thx for registering for EPiServer Find. To be able to create an index you need to verify your er Follow this link to verify your account http://find.episerver.com/verify/emailverification/023ea	nail. 74b-a131-				

If you don't receive your verification email, check your junk mail folder!

5. On the E-mail Activation verification page, click My Services, as shown in the following screenshot:

ebr E	mail Activation	×	+															-		×
\leftarrow	ightarrow O	find.e	piser	er.com/ve	erify/ema	ailverifica	ation/02	23ea74b-	a131-	15-	96ec651	0618e			☆	·	<u> </u>	1	٩	
E	oi 📙 Add Ons	DXC	Ti	aining	Partne	ers 📙	Social	Find												
	201											Weld	come c	s6dot	netco	re C	hange	Passwo	rd Log	Off
	OP I																	Му	Servio	es
	E-mail Act	tivatio	on																	
	Thank you for verif	ifying you	ir ema	il address																
Сору	right 2016 Episerve	er AB.																		
Epis	erver Find is powere	red by the	e grea	t open so	urce pro	ijects Ela	asticSea	arch and	node.js.											

- 6. In My Services, click Add Developer Service.
- 7. Fill in a name for your index. The name is technically irrelevant but should preferably represent what you're using the index for, such as, *CustomerNameTestIndex*.



8. Select at least **English**, **Swedish**, and **Danish** from the list of languages, check the terms and conditions checkbox, and click **Create Service**, as shown in the following screenshot:

Create D	eveloper Se	rvice			
1. Index Name Index name 2 Languages Arabic Catalan Dutch German Italian Romanian Thai 3. Terms and C	CmsAdvDevCourse	 Basque Cjk Finnish Hindi Persian Spanish Latvian 	 □ Brazilian □ Czech □ French □ Hungarian □ Polish ✓ Swedish 	 Bulgarian Danish Galician Indonesian Portuguese Turkish 	Index name The name of your index helps you identify it. The actual index name in will be a combination of your username and this name. Choose which languages the index should support. Select only as many as you need as each language carry a slight performance overhead while indexing.
I Accept the term	s and conditions 🛛 🖓		Cr	eate Service	Terms Please read through our terms and conditions and accept them.

9. You should now see a list of details about your index. Note the **Status**: it will probably be **NotCreated**, as shown in the following screenshot. If you wait a minute and refresh the page, it should say **CreatedWithStats**.

Index Details	My Service Overview
CmsAdvDevCourse Index Type: Developer Service Status: NotCreated Created: 2017-01-19 10:30:26	
Settings	Index
Number of documents: 10000 Languages: Danish, English, Swedish Allow Attachments:	Index Name: cs6dotnetcore_cmsadvdevcourse Public URL: https://es- api01.episerver.net/N760LIfGgukxM8f SrxKB42VcKU Private URL: https://es- api01.episerver.net/OMQINVpr VThfwGd002EDLzCUuhx

10. Note the **Configuration**, as shown in the following screenshot. In the next section, you will copy and paste this into the site's Web.config.



Installing and configuring Episerver Find

- 1. Open the solution with the AlloyTraining project.
- 2. Navigate to Tools | NuGet Package Manager | Package Manager Console.



3. Enter the following commands:

Install-Package -ProjectName AlloyTraining EPiServer.Find.Cms Update-EPiDatabase

- 4. Open ~\Web.config.
- 5. Add the following to the **<configSections>** element (you can copy and paste it from the Find Configuration page):

```
<section name="episerver.find" requirePermission="false"
    type="EPiServer.Find.Configuration, EPiServer.Find" />
```

 Add the following to the <configuration> element (you must copy and paste it from your Find Configuration page because your serviceURL contains a unique private account key):

```
<episerver.find
serviceUrl="https://es-eu-api01.episerver.net/PlS....kxv"
defaultIndex="episervertraining index58384"/>
```

- 7. Start the site, and log in as a CMS admin.
- 8. Navigate to CMS | Admin | Admin | Scheduled Jobs | EPiServer Find Content Indexing Job.
- 9. Click Start Manually, and wait for the job to complete, as shown in the following screenshot:

EPiServer Find Content Indexing Job

rerunning or scheduling of this jo	became content. During normal operation changes to content are being indexed as they are made without be applied to the second secon
The job is running Indexing jo	b [AlloyTraining] [content]: EPUKLPTMAPR: Indexing [en]: 0 to 52. Skipped: 0 item(s)
Settings History	
	Active
Scheduled job interval	second V
Next scheduled date	2017-01-19 00:00
	📙 Save 🚨 Start Manually 🔀 Stop Job

10. Click History. Note the number of content items indexed, as shown in the following screenshot:

Date	Duration	Status	Server	Message
1/19/2017 9:56:16 AM	15s	Succeeded	EPUKLPTMAPR	Indexing job [AlloyTraining] [content]: Reindexing completed. ExecutionTime: 0 hours 0 minutes 7 seconds Number of contents indexed: 55 Indexing job [Global assets and other data] [content]: Reindexing completed. ExecutionTime: 0 hours 0 minutes 8 seconds Number of contents indexed: 123

11. In the **Global** menu, click inside the **Search** box, and enter **track**. You should see multiple matches, with results shown from both Episerver Search (labeled **Files** and **Pages**) and Episerver Find (labeled **Find files** and **Find pages**), as shown in the following screenshot:

	የባቢ 🔇 ? 💄 Admin 🔼
	Search Track Search
Find files	EditorialTexts.txt
Find pages	Alloy Track - From start-up meetings to final
Files	EditorialTexts.txt
Pages	Alloy Track - From start-up meetings to final



12. Navigate to **CMS** | Admin | Config | Search Configuration. Note how the Search Providers have check boxes to disable them in **Global Search**, and they can be dragged and dropped to change the order in which they are called and displayed in the search results.

Reviewing the administrator's view of Episerver Find

1. In the Global menu, navigate to Find | Overview, as shown in the following screenshot:

For All Websites - in All languages	
Overview	
Index	
Index Name: episervertraining_index58384	
Find .NET API Version: 12.7.1.0	
Document Types The index contains 24 documents mappable to .NET objects. These are of 10 different types.	
Type Count	t
EPiServer.Core.ContentFolder, EPiServer	6
AlloyTraining.Models.Media.ImageFile, AlloyTraining	4
AlloyTraining.Models.Pages.ProductPage, AlloyTraining	3

2. Click the green box to slide out the **Explore** area, as shown in the following screenshot:

	Fo	or All Websites in All languages	•	Show Help?
ш	∕ Explore			
	Q Filter		Filter by Tyj	ре
		24 results	ContentFolder	6
	Name	Type	ImageFile	4
	numo	.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	ProductPage	3
	> Events	NewsPage	ContainerPage	3
	> Press Releases	NewsPage	AnyFile	2
	> Alloy Meet	ProductPage	NewsPage	2
	> About us	StandardPage	PageData	1
	> For This Site	ContentFolder	StartPage	1
	> Translations.pdf	AnyFile	StandardPage	1

3. Close the browser.

Creating a search page type

These steps apply for both Episerver Search and Episerver Find.

1. In AlloyTraining, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.



- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Type, enter SearchPage.cs for the Name, and click Add.
- 3. Modify the class to inherit from StandardPage.
- 4. Change the DisplayName to Search.
- 5. Group the page type under **Specialized**.
- 6. Add a Description of "Use this to enable visitors to search for pages and media on the site."
- 7. Apply the [SiteSearchIcon] attribute to the class.
- 8. Delete the MainBody property.

Your code should look something like the following:

```
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
namespace AlloyTraining.Models.Pages
{
    [ContentType(DisplayName = "Search",
      GroupName = SiteGroupNames.Specialized, Order = 30,
      Description = "Use this to enable visitors to search for pages and
media on the site.")]
    [SiteSearchIcon]
    public class SearchPage : StandardPage
    {
    }
}
```

Creating a search page view model

- 1. In AlloyTraining, expand Models, right-click ViewModels, and click Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Code, choose Class, enter SearchPageViewModel.cs for the Name, and click Add.
- 3. Inherit from PageViewModel<SearchPage> and fix the compile error using autofix to add the required constructor.
- 4. Modify the contents, to add some properties for the visitor's free-text search term and for the results, as shown in the following code:

```
using AlloyTraining.Models.Pages;
using System.Collections.Generic;
namespace AlloyTraining.Models.ViewModels
{
    public class SearchPageViewModel : PageViewModel<SearchPage>
    {
        public SearchPageViewModel(SearchPage currentPage) : base(currentPage)
        {
        }
        public string SearchText { get; set; }
        public List<Result> SearchResults { get; set; }
    }
    public class Result
    {
        public string Title { get; set; }
        public string Description { get; set; }
        public string Url { get; set; }
    }
```



}

We need to declare a class to represent each result because we want to keep search results abstract and not tied to a specific technology like Episerver Find or Episerver Search.

Implementing a search page controller using Episerver Find

- To use Episerver Search, skip ahead to Implementing a search page controller using Episerver Search on page 161.
 - 1. In AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press Ctrl + Shift + A.
 - 2. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter SearchPageController.cs for the name, and click Add.
 - 3. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Pages** namespace.
 - 4. Modify the class to inherit from **PageControllerBase**, create an instance of the search page view model and set its properties, before passing it to a view, as shown in the following code:

```
using AlloyTraining.Business.ExtensionMethods;
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using EPiServer.Find;
using EPiServer.Find.Cms;
using EPiServer.Find.Framework;
using EPiServer.Security;
using System.Linq;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class SearchPageController : PageControllerBase<SearchPage>
    {
        public SearchPageController(IContentLoader loader) : base(loader)
        {
        }
        public ActionResult Index(SearchPage currentPage, string q)
        {
            var viewmodel = new SearchPageViewModel(currentPage);
            viewmodel.StartPage = loader.Get<StartPage>(ContentReference.StartPage);
            viewmodel.MenuPages = FilterForVisitor.Filter(
                loader.GetChildren<SitePageData>(ContentReference.StartPage))
                .Cast<SitePageData>().Where(page => page.VisibleInMenu);
            viewmodel.Section = currentPage.ContentLink.GetSection();
            if (!string.IsNullOrWhiteSpace(q))
                var query = SearchClient.Instance
                    .Search<SitePageData>() // 1. only pages
                                            // 2. free-text query
                    .For(q)
                    // 3. only what the current user can read
                    .FilterForVisitor()
                    // 4. only under the Start page (to exclude Wastebasket)
                    .FilterOnCurrentSite();
```



```
var results = query.GetContentResult();
viewmodel.SearchText = q;
viewmodel.SearchResults = results
.Select(x => new Result
{
Title = x.MetaTitle ?? x.Name,
Description = x.MetaDescription?.TruncateAtWord(20),
Url = x.PageLink.ExternalURLFromReference()
}).ToList();
}
return View(viewmodel);
}
```

Implementing a search page controller using Episerver Search

If you have already implemented the search page controller using Episerver Find, skip ahead to Creating a search page view on page 163.
 In AlloyTraining, expand Controllers, and right-click and choose Add | New Item..., or press *Ctrl* + *Shift* + *A*.
 In Add New Item - AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter SearchPageController.cs for the Name, and click Add.
 Fix the compilation error by clicking the light bulb, and choose the option to import the

- 3. Fix the compilation error by clicking the light bulb, and choose the option to import the **AlloyTraining.Models.Pages** namespace.
- 4. Modify the class to inherit from **PageControllerBase**, create an instance of the search page view model and set its properties, before passing it to a view, using extension methods to generate the URL to the page and to truncate the descriptions of the pages to 20 words each, as shown in the following code:

```
using AlloyTraining.Business.ExtensionMethods;
using AlloyTraining.Models.Pages;
using AlloyTraining.Models.ViewModels;
using EPiServer.Core;
using EPiServer.Filters;
using EPiServer.Search;
using EPiServer.Search.Queries.Lucene;
using EPiServer.Security;
using System.Linq;
using System.Web.Mvc;
namespace AlloyTraining.Controllers
{
    public class SearchPageController : PageControllerBase<SearchPage>
    {
        private readonly SearchHandler searchHandler;
        public SearchPageController(IContentLoader loader,
            SearchHandler searchHandler) : base(loader)
        {
            this.searchHandler = searchHandler;
        }
        public ActionResult Index(SearchPage currentPage, string q)
        {
```

```
var viewmodel = new SearchPageViewModel(currentPage);
```



```
viewmodel.StartPage = loader.Get<StartPage>(ContentReference.StartPage);
    viewmodel.MenuPages = FilterForVisitor.Filter(
        loader.GetChildren<SitePageData>(ContentReference.StartPage))
        .Cast<SitePageData>().Where(page => page.VisibleInMenu);
    viewmodel.Section = currentPage.ContentLink.GetSection();
    if (!string.IsNullOrWhiteSpace(q))
    {
        // 1. only pages
        var onlyPages = new ContentQuery<SitePageData>();
        // 2. free-text query
        var freeText = new FieldQuery(q);
        // 3. only what the current user can read
        var readAccess = new AccessControlListQuery();
        readAccess.AddAclForUser(PrincipalInfo.Current, HttpContext);
        // 4. only under the Start page (to exclude Wastebasket, for example)
        var underStart = new VirtualPathQuery();
        underStart.AddContentNodes(ContentReference.StartPage);
        // build the query from the expressions
        var query = new GroupQuery(LuceneOperator.AND);
        query.QueryExpressions.Add(freeText);
        query.QueryExpressions.Add(onlyPages);
        query.QueryExpressions.Add(readAccess);
        query.QueryExpressions.Add(underStart);
        // get the first page of ten results
        SearchResults results = searchHandler.GetSearchResults(query, 1, 10);
        viewmodel.SearchText = q;
        viewmodel.SearchResults = results.IndexResponseItems
            .Select(item => new Result
            {
                Title = item.Title,
                Description = item.DisplayText?.TruncateAtWord(20),
                Url = ConvertToUri(item).ToString()
            }).ToList();
    }
    return View(viewmodel);
}
private Uri ConvertToUri(IndexResponseItem item)
{
    try
    {
        var url = new UrlBuilder(item.Uri);
        Global.UrlRewriteProvider.ConvertToExternal(url, item, Encoding.UTF8);
        return url.Uri;
    }
    catch
    {
        return default(Uri);
    }
}
```

}

}

Creating a search page view

- 1. In AlloyTraining, right-click Views, and add a new folder named SearchPage.
- 2. Right-click SearchPage, and choose Add | New Item..., or press Ctrl + Shift + A.
- 3. In Add New Item AlloyTraining, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.
- 4. Modify the view, as shown in the following markup, and note the following:
 - a. To support searching in Edit view, the form must POST instead of GET.
 - b. If there is search text, the Search Results are shown, and then either shows a warning if there are no matches, or the total hits and the list of results.

```
@using AlloyTraining.Models.ViewModels
Qusing EPiServer.Editor
@model SearchPageViewModel
<div class="row">
    <div class="span8">
        @using (Html.BeginForm(actionName: null,
          controllerName: null, routeValues: null,
          method: PageEditing.PageIsInEditMode ? FormMethod.Post : FormMethod.Get))
        {
            <input tabindex="1" name="q" value="@Model.SearchText" />
            <input type="submit" tabindex="2" class="btn" value="Search" />
        @if (!string.IsNullOrWhiteSpace(Model.SearchText))
            <div class="row">
                <div class="span8 grayHead">
                    <h2>Search Results</h2>
                </div>
            </div>
            if (Model.SearchResults.Count == 0)
            {
                <div class="row">
                    <div class="span8 SearchResults">
                        <span class="label label-warning">No matching results.</span>
                    </div>
                </div>
            }
            else
            {
                <div class="row">
                    <div class="span8 SearchResults">
                        <span class="label label-success">@Model.SearchResults.Count
matching results.
                        @foreach (var item in Model.SearchResults)
                        {
                            <div class="listResult">
                                <h3><a href="@item.Url">@item.Title</a></h3>
                                @item.Description
                                <hr />
                            </div>
                        }
                    </div>
                </div>
           }
        }
    </div>
```



```
<div class="span4">
    @Html.PropertyFor(m => m.CurrentPage.MainBody)
    </div>
</div><//div>
```

Creating and testing the search page

- 1. Start the **AlloyTraining** website, and log in as a CMS admin.
- 2. Add a new **Search** page named **Search** under the **Start** page.
- 3. Publish the Search page.
- 4. Switch to **Live view** and click **Search** to navigate to the **Search** page as a visitor.
- 5. Enter the search text **track**, and press *ENTER* or click **Search**, as shown in the screenshot:

Resetting the search index for Episerver Find

- 1. Navigate to Find | Configure | Index.
- 2. Click Clear Index, as shown in the following screenshot:



Start Alloy Meet Alloy Plan Alloy Track About us Search

:k					
Search	ı				
	_				

Search Results



Welcomel Best home page EVERI EditorialTexts.txt translations pdf Children of Alloy Meet Alloy Meet Customer Testimonial Alloy Meet Alloy Plan Alloy Track AlloyMeet.png AlloyPlan.png AlloyTrack.png About us Designed in Sweden

Alloy Track

Projects have a natural lifecycle with well-defined stages. From start-up meetings to final sign-off, we have the solutions for today's market-driven \dots

ick Clear Index , as shown in the following screenshot:							
🔧 Tools							
💭 Boosting	Index (P) Connectors						
Use this view to clear the content index and/or statistics for the we content structures and properties.	ebsite. This may be needed when extensive changes have been done to						
Clear content index Clearing the content index will empty the index completely, after which the website needs to be <u>reindexed</u> .	Clear statistics Clearing the statistics will remove all data, causing the statistics view to be empty.						
Clear index	Clear statistics						

Resetting the search index for Episerver Search

1. Navigate to CMS | Admin | Admin | Tools | Index Site Content, as shown in the following screenshot:



- 2. Note the date and time of the latest complete indexing, and then click Start Indexing.
- 3. Press *F*5 to refresh the page and see the latest complete indexing date and time.



Exercise E5 – Adding a search box to the top navigation menu

In this exercise, you will implement a search box to use the logic that was added in the Search Page in the previous exercise to perform searches from other places in your site, giving visitors to the site a better experience and more options.

You will add a search field and a button to the existing main navigation control, add a new property to the start page that holds a reference to the Search page and then write logic that, when the search button in the main navigation is clicked, adds the search text to the query string and does a redirect to the search page.



Prerequisites: complete Exercises B1 - B4, E4.

Adding a property to the Start page to reference the Search page

1. In AlloyTraining, open ~\Models\Pages\StartPage.cs, and add a property to reference an instance of the search page type, as shown in the following code:

```
[Display(Name = "Search page",
    Description = "If you add a Search page to the site, set this property to
reference it to enable search from every page.",
    GroupName = SiteTabNames.SiteSettings,
    Order = 40)]
[AllowedTypes(typeof(SearchPage))]
public virtual PageReference SearchPageLink { get; set; }
```

Adding a search box to the navigation menu

- 1. In AlloyTraining, open ~\Views\Shared_NavigationMenu.cshtml.
- 2. At the top of the view, add a statement to import Alloy's extension methods, as shown in the following code:

@using AlloyTraining.Business.ExtensionMethods

3. After the element, inside the nested <div> elements, add the following markup:

```
<div class="navbar-search pull-right">
 @if (Model.StartPage != null)
 {
    @if (PageReference.IsNullOrEmpty(Model.StartPage.SearchPageLink))
    {
        if (EPiServer.Editor.PageEditing.PageIsInEditMode)
        {
            <div class="alert alert-danger">To enable search across the site,
                set the SearchPageLink property.</div>
        }
    }
    else
    {
        <form action="@Model.StartPage.SearchPageLink.ExternalURLFromReference()"</pre>
              method="post">
            <input type="text" class="search-query" name="q"</pre>
                   id="SearchKeywords" placeholder="Search" />
            <input type="submit" class="searchButton" id="SearchButton" value="" />
        </form>
```





Enable the search box and test the website

- 1. Start the AlloyTraining website, and log in as a CMS admin.
- 2. Edit the Start page, and note the warning for editors, as shown in the following screenshot:



3. Switch to **All Properties** view, click **Site Settings**, and set the **Search page** property to reference the **Search** page, as shown in the following screenshot:

🗅 Start		Select Page ×
		Q Search
	Start	C Root
	start Change	🖬 🕒 Start
		C Search
		Alloy Meet Alloy Plan
		Alloy Track
Site Settings		🗋 About us

- 4. Publish the Start page.
- 5. Switch to Live view.
- 6. Navigate to any page, enter **track** in the search box, click the search button or press *ENTER*, and note you are directed to the **Search** page to see the results.



Module F – Working with Episerver Framework

Goal

The overall goal of the exercises in this module is to learn how to work with Episerver Framework and content APIs. You will:

- 1. Export and import content.
- 2. Implement FAQs by programmatically creating pages using content APIs.
- 3. Validate content by listening for publishing events.
- 4. Process content by implementing a scheduled job.
- 5. Decompile Episerver assemblies to better understand how to work with our APIs.

Exercise F1 – Exporting and importing content

In this exercise, you will export data from an Episerver CMS site, and then import it back in as if it were new content.

Prerequisites: complete Exercise A1.

Exporting news & events from AlloyDemo website

- 1. Open the Training solution with the AlloyDemo project.
- 2. Start the AlloyDemo website, and log in as Admin.
- 3. Navigate to CMS | Admin | Admin | Tools | Export Data, as shown in the following screenshot:



4. Click **Export content items**, and select **News & Events**, and note that by default the export will include sub items and any files (media assets) that the pages link to, as shown in the following screenshot:

Export content items
Select part of structure
E Root
Start
Alloy Plan
Alloy Track
Alloy Meet
About us
News & Events
Contacts
Management
Contact us
Become a reseller
How to buy
Campaigns
Search
Alloy Go
Recycle Bin
For All Sites
Customer Zone
 ✓ Include sub items ✓ Export files that the pages link to Automatically export dependent content types
Export content types



5. Click **Test Run with Error Log**, and note that you get no errors or warnings, as shown in the following screenshot:

Export Data	0
Export data from one EPiServer CMS to another.	
Export completed without errors or warnings	
Export content items	
Export content types	
Export frames	
Export tabs	
Export categories	
Export visitor groups	
	🕞 Export 📄 Test Run with Error Log

- 6. Click Export.
- 7. Start File Explorer, open your Downloads folder, and note that a file has been created named ExportedFile.episerverdata, as shown in the following screenshot:

🖊 📝 📙 🖛 Down	loads			-		×
File Home Sh	nare View					~ 🕐
← → ~ ↑ 🖊 ›	This PC > Downloads >		√ Ö	Search Downloads		Q,
🗸 💻 This PC	^ Name ^	Date	Туре	Size	Length	
> 📃 Desktop	Install	14/12/2016 08:34	File folder			
> 🚼 Documents	ExportedFile.episerverdat	ta 10/06/2017 14:18	EPISERVERDAT	A File 1,713 KB		
> 👆 Downloads						
> 👌 Music	¥					
2 items						

- 8. Copy the file, and change its file extension to ZIP.
- 9. Open the ZIP file, and note the contents, as shown in the following screenshot:

I → ↓	Compresse	ed Folder Tools	ExportedFile - Copy					- 0	×
File Home Share	View	ατασ							~ 🕜
← → × ↑ 🔢 > This PC > Downloads > ExportedFile - Copy > 🗸 🕐 Search ExportedFile - Copy 🔎								R	
🍊 OneDrive	Name	Туре	Compressed size	Password protected	Size		Ratio	Date modified	
This DC	epi.fx.blob:	File folder							
	📙 handleddata	File folder							
Desktop	[Content_Types]	XML Document	1 KB	No		1 KB	31%	10/06/2017 14:1	8
Documents	epiDefinition	XML Document	1 KB	No		1 KB	0%	10/06/2017 14:1	8
👆 Downloads	📄 epiMedia	XML Document	1 KB	No		4 KB	83%	10/06/2017 14:1	8
b Music	📄 epiPostContent	XML Document	1 KB	No		1 KB	0%	10/06/2017 14:1	8
Pictures	📄 epix	XML Document	19 KB	No	47	1 KB	97%	10/06/2017 14:1	8
Videos	📄 idmap	XML Document	1 KB	No		6 KB	83%	10/06/2017 14:1	8
Lindows (C:)									
🗙 data (\\ep.se\glc									
Network									
8 items								8	

The epi.fx.blob: folder contains the media assets like images used in the news & event articles exported. The epix.xml file contains the page and block content and their property values.

Importing news & events to AlloyDemo website

- 1. Start the AlloyDemo website, and log in as Admin.
- 2. Navigate to CMS | Admin | Admin | Tools | Import Data, choose the ExportedFile.episerverdata file, and select Alloy Plan as the content destination.



3. Clear the **Update existing content items with matching ID** check box, and click **Begin Import**, as shown in the following screenshot:

Import Data		?
Import data from another EPiServer C	MS.	
Select an export file and upload	Choose File ExportedFile.episerverdata	
Select content destination		
Koot Koot		
Update existing content items with	matching ID	
Select language	All O English O svenska	
		Regin Import

If you do not clear the Update existing content items with matching ID check box, then the pages will appear not to import, because the it will just update existing items rather than add pages as new items!

4. After a few moments, the import should complete, as shown in the following screenshot:

Import Data	?
Import data from another EPIServer CMS.	
Import successful	
Number of content items: 30	
Number of content types: 0	

5. Navigate to **CMS** | **Edit**, and expand the **Pages** tree to see that **Alloy Plan** now has some new child pages, as shown in the following screenshot:

	\$	Ŧ
V Pages Sites Tasks Project Iter	ms	
Q Search		\supset
B Root		
🗉 🏫 Start		≡•
🖬 🕒 Alloy Plan		
News & Events		
🖬 🕒 Events		
🖪 🕒 Press Releases		
🖪 🗋 Download Alloy Plan		
Alloy Track		
🖬 🕒 Alloy Meet		
🖬 🕒 About us		



Exercise F2 – Implementing FAQs with content APIs

In this exercise, you will implement FAQs by implementing data pages and automating their creation using the content repository API.

You will create two page types; one to be used as the visible **FAQListPage** containing both a form for visitors to enter a question and a listing of answered questions, and one named **FAQItemPage** that will act as a data page for a single question and its answer.

Here's an example of an FAQ page with one question answered:

AIIOY	Start Alloy Track Alloy Plan	Alloy Meet Alloy Cool About Us	
ALLOY FAQ PAG	ε		
Enter A Question!		Sub	mit Question
FAQ LIST Why is the sky blue? A clear cloudless day-time sky is	, blue because molecules in the air sca	atter blue light from the sun more than th	iey scatter red light.

Prerequisites: complete Exercise A1.

Create an FAQ item data page type and an FAQ list page type

The **FAQItemPage** will not have a template because it will just store data, but we want it to be created as a child of a **FAQListPage** that will have a page template that shows a list of its children.

- 1. Open the Training solution with the AlloyDemo project.
- 2. In Solution Explorer, in AlloyDemo, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 3. In Add New Item AlloyDemo, navigate to Installed | Visual C# | Episerver, choose Page Type, enter FAQItemPage.cs for the Name, and click Add.
- 4. Change the DisplayName to FAQ Item.
- 5. Add a Description of "A data page for an FAQ item (cannot be created by editors)."
- 6. Add the AvailableInEditMode parameter and set it to false.
- 7. Delete the MainBody property.
- 8. Add the following properties with appropriate attributes:
 - a. Question: XhtmlString
 - b. Answer: XhtmlString

FAQItemPage does not need an icon because content editors cannot see it in edit view. It does not need to inherit from SitePageData because it will not render as an HTML page with META elements.

Your code should look something like the following:

```
using EPiServer.Core;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
```



using System.ComponentModel.DataAnnotations;

```
namespace AlloyDemo.Models.Pages
{
    [ContentType(DisplayName = "FAQ Item",
        Description = "A data page for an FAQ item (cannot be created by editors).",
        AvailableInEditMode = false)]
    public class FAQItemPage : PageData
    {
        [Display(Name = "Question", Order = 10)]
        public virtual XhtmlString Question { get; set; }
        [Display(Name = "Answer", Order = 20)]
        public virtual XhtmlString Answer { get; set; }
    }
}
```

- 9. In AlloyDemo, expand Models, right-click Pages, and click Add | New Item..., or press Ctrl + Shift + A.
- 10. In Add New Item AlloyDemo, navigate to Installed | Visual C# | Episerver, choose Page Type, enter FAQListPage.cs for the name, and click Add.
- 11. Change the DisplayName to FAQ List.
- 12. Group the page type under Specialized by using a constant in Global.GroupNames.
- 13. Add a **Description** of "Use this page for a list of FAQs entered by visitors, answered by editors."
- 14. Apply the SiteImageUrl attribute to the class.
- 15. Apply an attribute to restrict this page types children to only be FAQItemPage instances.
- 16. Inherit from SitePageData.
- 17. Delete the MainBody property.
- 18. Add a property named **FAQItems** that can contain the child FAQ item pages and apply the **Ignore** attribute to it.

Good practice would be to define a view model to hold this property but marking a content type property with [Ignore] means the property will not be registered with the CMS so it can be used for a similar purpose as a view model. Just beware of object caching!

Your code should look something like the following:

```
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.Collections.Generic;
namespace AlloyDemo.Models.Pages
{
    [ContentType(DisplayName = "FAQ List",
        GroupName = Global.GroupNames.Specialized,
        Description = "Use this page for a list of FAQs entered by visitors,
answered by editors.")]
    [SiteImageUrl]
    [AvailableContentTypes(Include = new[] { typeof(FAQItemPage) },
        IncludeOn = new[] { typeof(StartPage) })]
    public class FAQListPage : SitePageData
    {
        // having an ignored property avoids needing a view model
        // this property will not be stored in CMS so it does not
        // need to be virtual
        [Ignore]
        public IEnumerable<FAQItemPage> FAQItems { get; set; }
    }
```



}

Creating an FAQ list page template

- 1. In AlloyDemo, expand Controllers, and right-click and choose Add | New Item..., or press Ctrl + Shift + A.
- 2. In Add New Item AlloyDemo, navigate to Installed | Visual C# | Episerver, choose Page Controller (MVC), enter FAQListPageController.cs for the Name, and click Add.
- Fix the compilation error by clicking the light bulb, and choose the option to import the AlloyDemo.Models.Pages namespace.
- 4. Modify the class to derive from PageControllerBase<T>.
- 5. Modify the **Index** action method to use a view model and set the **FAQItems** property using the content loader service.
- 6. Add a CreateFAQItem method, with two parameters: currentPage and question.
- 7. Use the content repository service to add a new FAQ item page under the current page, setting its **Question** property to the **question** parameter, and giving it a **Name**.
- 8. Save and check in the new FAQ item page if the current user has Read access rights.

Your controller should look something like the following code:

```
using AlloyDemo.Models.Pages;
using AlloyDemo.Models.ViewModels;
using EPiServer;
using EPiServer.Core;
using System.Web.Mvc;
namespace AlloyDemo.Controllers
{
    public class FAQListPageController : PageControllerBase<FAQListPage>
    {
        private readonly IContentRepository repo;
        public FAQListPageController(IContentRepository repo)
        {
            this.repo = repo;
        }
        public ActionResult Index(FAQListPage currentPage)
        {
            var viewmodel = PageViewModel.Create(currentPage);
            var faqs = repo.GetChildren<FAQItemPage>(currentPage.ContentLink);
            viewmodel.CurrentPage.FAQItems = faqs;
            return View(viewmodel);
        }
        public ActionResult CreateFAQItem(FAQListPage currentPage, string question)
        {
            var faqItem = repo.GetDefault<FAQItemPage>(currentPage.ContentLink);
            // if someone is logged in then CreatedBy and ChangedBy will be set,
            // otherwise they will be empty string which is shown as "installer"
            if (string.IsNullOrEmpty(faqItem.CreatedBy))
                 faqItem.CreatedBy = "Anonymous";
            if (string.IsNullOrEmpty(faqItem.ChangedBy))
                faqItem.ChangedBy = "Anonymous";
            faqItem.Question = new XhtmlString(question);
            faqItem.Name = "Q. " + question;
            repo.Save(faqItem,
```

Copyright © Episerver AB. All rights reserved.

```
EPiServer.DataAccess.SaveAction.CheckOut,
EPiServer.Security.AccessLevel.Read);
return RedirectToAction("Index");
}
}
```

- 9. In AlloyDemo, right-click Views, and add a new folder named FAQListPage.
- 10. Right-click **FAQListPage**, and choose **Add** | **New Item**..., or press *Ctrl* + *Shift* + *A*.
- 11. In Add New Item AlloyDemo, navigate to Installed | Visual C# | Episerver, choose Page Partial View (MVC Razor), enter Index.cshtml for the Name, and click Add.
- 12. Modify the view, as shown in the following markup, and note the following:
 - a. The question form submits back to the CreateFAQItem action method.
 - b. Content editors see a warning message.
 - c. Each FAQ item outputs its: question and when it was asked, and the answer and who and when it was answered.

```
@model AlloyAdvanced.Models.ViewModels.PageViewModel<FAQListPage>
<h2>Alloy FAQs</h2>
<div class="navbar">
    <h3>Enter your question</h3>
    @using (Html.BeginForm(actionName: "CreateFAQItem", controllerName: null))
    {
        <input type="text" tabindex="1" name="question"</pre>
               placeholder="Enter your question" />
        <input type="submit" tabindex="2" class="btn" value="Submit" />
    }
</div>
@if (EPiServer.Editor.PageEditing.PageIsInEditMode)
ł
    <div class="alert alert-info">Questions do not appear until they have been
answered by a content editor and published.</div>
}
<div class="nav-stacked">
    <h2>Ouestions with answers</h2>
    @foreach (var faqItem in Model.CurrentPage.FAQItems)
    {
    <div class="alert alert-success">
        <small class="label label-inverse">Asked at
@faqItem.Created.ToShortTimeString() on
@faqItem.Created.ToShortDateString()</small>
        @Html.DisplayFor(m => faqItem.Question)
        <small class="label label-inverse">Answered by @faqItem.ChangedBy at
@faqItem.StartPublish.Value.ToShortTimeString() on
@faqItem.StartPublish.Value.ToShortDateString()</small>
        @Html.DisplayFor(m => faqItem.Answer)
    </div>
    }
</div>
```

Test the website FAQ functionality

1. Start the AlloyDemo website, and log in as Admin.



- 2. Create an FAQ list page named **FAQs** under the **Start** page, as shown in the screenshot:
- 3. Publish the page.
- 4. Switch to Live view to experience the website as a visitor.
- 5. Navigate to the **FAQs** page.
- 6. Enter and submit a question, for example, "Why is the sky blue?", and note that the FAQ does NOT appear, as shown in the following screenshot:

New Page:	
Start	
Name FAQs	
Other Page Type	25
	FAQ List
Α	Use this page for a list of FAQs entered by visitors, answered by editors.

	AIIOY	Start Alloy Plan	Alloy Track	Alloy Meet	About us	Alloy Go FAQ	S		
	ALLOY FAQ PA Enter A Question!	GE							
	Why is the sky blue?						Submit		
	FAQ LIST								
	Products Alloy Plan Alloy Track	The About t News &	Company	1		News & Ev Events Press Releases	vents		
7.	Log in as Admin , v	og in as Admin , view the Pages tree, and note the FAQs page			B	Search			
	has a child with a	pencil icon to	icon to indicate that it has a dra		raft that	B	Alloy Go		
	has not been published yet, as shown in the s			the scr	creenshot:		FAQs	≡•	
lf +h	ne browser was already running, press F5 to refresh the page to see newly created FAQ item page.					🗋 Q. Why is the sky blu	. 🖉		
the					🖬 📄 Cu	istomer Zone			

- 8. Edit the question, answer it, and publish it.
- 9. Switch to Live view, and note the question and its answer now appear, as shown in the following screenshot:

Alloy FAQs

Enter your quest	on and a second s	
Enter your question		
Submit		

Questions with answers

Asked at 9:18 AM on 3/21/2018
What c olor is the sky?
Answered by Admin at 9:19 AM on 3/21/2018
blue

10. Close the browser.



Exercise F3 – Listening for events and customizing services with initialization modules

In this exercise, you will create initialization modules that (1) prevents creating a page if the parent page already has more than a maximum number of children, and (2) removes the service that suggests most recent content types.

Prerequisites: complete Exercise A1.

Creating an initialization module to listen for events

- 1. In ~\Business\Initialization, add an Episerver | Initialization Module named PreventMoreThanMaxChildrenInitializationModule.cs.
- 2. Modify the statements, as shown in the following code:

```
using AlloyDemo.Models.Pages;
using EPiServer;
using EPiServer.Core;
using EPiServer.Framework;
using EPiServer.Framework.Initialization;
using System.Linq;
namespace AlloyDemo.Business.Initialization
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
    public class PreventMoreThanMaxChildrenInitializationModule : IInitializableModule
    ł
        private bool initialized = false;
        private IContentEvents events;
        private IContentLoader loader;
        private const int maxChildren = 8;
        public void Initialize(InitializationEngine context)
        {
            if (!initialized)
            {
                loader = context.Locate.ContentLoader();
                events = context.Locate.ContentEvents();
                events.CreatingContent += Events_CreatingContent;
                initialized = true;
            }
        }
        private void Events_CreatingContent(object sender, ContentEventArgs e)
            var sitepage = e.Content as SitePageData;
            if (sitepage != null)
            {
                var children = loader.GetChildren<IContent>(sitepage.ParentLink);
                if (children.Count() >= maxChildren)
                {
                    e.CancelAction = true;
                    e.CancelReason =
        $"Cannot create a new page if the parent has {maxChildren} or more children.";
                }
            }
        }
        public void Uninitialize(InitializationEngine context)
        {
```



```
if (initialized)
{
    events.CreatingContent -= Events_CreatingContent;
    }
}
```

- 3. Start the site, and log in as Admin.
- 4. Navigate to CMS | Edit.
- 5. Add a **Standard** page underneath the **Start** page named **Alpha**. You will be able to create the page, as shown in the following screenshot:



- 6. **Publish** the Alpha page.
- 7. Add a **Standard** page underneath the **Start** page named **Beta**. This time, you will not be able to create the page, because Start now has eight children, as shown in the following screenshot:

	E +			A 🗖		
Pages Sites Tasks Project Items						
Q Search	New Page: Stan	dard Page				
B 🗐 Root	Start			1 Cancel		
🖬 🏫 Start						
🖬 🗋 Alloy Plan	Name Beta		Something went w	rong 🕕		
🖬 🗋 Alloy Track			Cannot create a new page if the parent has 8 or more			
🖬 🗋 Alloy Meet	Suggested Page	a Types	children.			
🖬 🗋 About us						
🖬 🚍 How to buy		Product		Standard Page		
🖪 📄 Campaigns		Used to present a specific		Used mainly for default		
🗋 Search		product		editorial content such as text		
🗅 Alpha 🗧	÷			and mayes		
🗉 🚍 Customer Zone						

- 8. Click Cancel.
- 9. Close the browser.

Limiting the number of children to eight will cause you annoyance later, so change the limit in the initialization module to something like 100, as shown in the following code:

private const int maxChildren = 100;

Creating an initialization module to remove a service

1. In the previous screenshot, note the **Suggested Page Types** group. This group is automatically created based on the most recently used page types.



- 2. In ~\Business\Initialization, add an Episerver | Initialization Module named RemoveSuggestedPageTypesInitializationModule.cs.
- 3. Modify the statements, as shown in the following code:

Note the class implements IConfigurableModule (that itself implements IInitializationModule), and that the method we implement to remove the service is named ConfigureContainer. Both the methods Initialize and Uninitialize must exist, but in this scenario, they do nothing.

```
using EPiServer.Cms.Shell.UI.Rest;
using EPiServer.Framework;
using EPiServer.Framework.Initialization;
using EPiServer.ServiceLocation;
namespace AlloyDemo.Business.Initialization
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
    public class RemoveSuggestedPageTypesInitializationModule : IConfigurableModule
    ł
        public void ConfigureContainer(ServiceConfigurationContext context)
        {
            context.StructureMap().EjectAllInstancesOf<IContentTypeAdvisor>();
        }
        public void Initialize(InitializationEngine context) { }
        public void Uninitialize(InitializationEngine context) { }
    }
}
```

- 4. Start the site, and log in as Admin.
- 5. Navigate to CMS | Edit.
- 6. Add a **Standard** page underneath the **Start** page, and note the **Suggested Page Types** group has gone, as shown in the following screenshot:





Install Episerver Developer Tools to monitor initialization modules

- 1. Navigate to Tools | NuGet Package Manager | Package Manager Console.
- 2. Enter the following command:

Install-Package -ProjectName AlloyDemo EPiServer.DeveloperTools

- 3. Start the website, and log in as Admin.
- 4. Navigate to **Developer | Startup Perf**, as shown in the following screenshot:

III localhost:60954/EPiServe ×			Merik	— C	: נ	×
\leftrightarrow \rightarrow C \triangle \bigcirc localhost:6	0954/EPiServer/EPiServer.DeveloperTo	ols/TimeMeters			$\dot{\mathbf{x}}$:
Dashboard CMS Deve	loper		ерг 🜏 ?	1 Alice	Q	
Container Content Type Analyz	er Loaded Assemblies Log Viewer	Memory Dump Remote Event	Revert Content Types	Routes		
Startup Perf Templates View	w Locations					
Startup Performance Displays timing measurements from the initialization process, can be used to find modules causing slow startups. Number of time meters: 96. Total time: 6.78 s.						
		Action			(ms) 🔻	_
EPiServer.Data	DataInitialization	Initialize			2298	
EPIServer.Forms.UI		Initialize			1085	
EPiServer.Marketing.Testing.web	NedelSynchriticalization	Initialize			204	
EPiSenver Shell	ShellInitialization	Initialize			304	
EPiServer Framework	ServiceContainerInitialization	ConfigureContainer			328	
EPiServer.Enterprise	EnterpriseInitialization	Initialize			306	
EPiServer	PlugInInitialization	Initialize			202	
EPiServer.Framework	InitializationEngine	OnInitComplete EPiServer.Personalization.Visito <>cDisplayClass7_0	orGroups.VisitorGroupInitiali	zation+	183	



Exercise F4 – Implementing scheduled jobs

In this exercise, you will implement a scheduled job to modify page names, titles, and descriptions that contain bad words.

Prerequisites: complete Exercise A1.

Create a scheduled job

- 1. Open the solution that includes the AlloyDemo project.
- 2. In Solution Explorer, right-click ~/Business, add a folder named ScheduledJobs, and add an Episerver Scheduled Job named BadWordScannerScheduledJob.
- 3. Modify its code as shown below:

```
using AlloyDemo.Models.Pages;
using EPiServer;
using EPiServer.Core;
using EPiServer.PlugIn;
using EPiServer.Scheduler;
using EPiServer.ServiceLocation;
namespace AlloyDemo.Business.ScheduledJobs
{
    [ScheduledPlugIn(DisplayName = "Bad Word Scanner",
       Description = "Scan for bad words in pages and censor them.")]
    public class BadWordScannerScheduledJob : ScheduledJobBase
    {
        private bool _stopSignaled;
        // you could load this from a file or service
        private string[] badWords = new[] { "frak" };
        public BadWordScannerScheduledJob()
        {
            IsStoppable = true;
        }
        public override void Stop()
        {
            _stopSignaled = true;
        }
        public override string Execute()
        {
            OnStatusChanged(string.Format(
                "Starting execution of {0}", this.GetType()));
            var repo = ServiceLocator.Current.GetInstance<IContentRepository>();
            var finder = ServiceLocator.Current
                .GetInstance<IPageCriteriaQueryService>();
            int pageCount = 0;
            foreach (string word in badWords)
            {
                var criteria = new PropertyCriteriaCollection();
                criteria.Add(new PropertyCriteria
                {
                    Type = PropertyDataType.LongString,
```


```
Name = "PageName",
                    Condition = EPiServer.Filters.CompareCondition.Contained,
                    Value = word
                });
                criteria.Add(new PropertyCriteria
                {
                    Type = PropertyDataType.LongString,
                    Name = "MetaTitle",
                    Condition = EPiServer.Filters.CompareCondition.Contained,
                    Value = word
                });
                criteria.Add(new PropertyCriteria
                {
                    Type = PropertyDataType.LongString,
                    Name = "MetaDescription",
                    Condition = EPiServer.Filters.CompareCondition.Contained,
                    Value = word
                });
                PageDataCollection results = finder.FindPagesWithCriteria(
                    ContentReference.RootPage as PageReference, criteria);
                foreach (SitePageData page in results)
                {
                    var clone = page.CreateWritableClone() as SitePageData;
                    clone.Name = page.Name.Replace(word, "[censored]");
                    clone.MetaTitle = page.MetaTitle.Replace(word, "[censored]");
                    clone.MetaDescription =
                        page.MetaDescription.Replace(word, "[censored]");
                    repo.Save(clone,
                        EPiServer.DataAccess.SaveAction.CheckIn,
                        EPiServer.Security.AccessLevel.NoAccess);
                    pageCount++;
                }
                //For long running jobs periodically check if stop is signaled and if
so stop execution
                if ( stopSignaled)
                {
                    return "Stop of job was called";
                }
            }
            return $"{pageCount} pages containing one of the following bad words:
'{string.Join("' or '", badWords)}' have been censored.";
        }
    }
```

Testing the scheduled job

}

- 1. Start the AlloyDemo site, and log in as Admin.
- 2. Edit Start, add frak into its Name, and publish the change.
- 3. Edit About us, add frak into its Title (MetaTitle), and publish the change.
- 4. Edit Alloy Track, add frak into its Page description (MetaDescription), and publish the change.



- 5. Navigate to CMS | Admin | Admin | Scheduled Jobs | Bad Word Scanner.
- 6. Set the job to be active, and set it to run every hour, and change the next schedule date to two minutes in the future, and click **Save**, as shown in the following screenshot:

Bad Word Scanner			
Scan for bad words in pages and	censor them.		
Settings have been saved.			
Settings History			
	 Active 		
Scheduled job interval	1	hour v	
Next scheduled date	2017-06-24 10:16		
			🛃 Save 🚨 Start Manually 🔀 Stop Job

- 7. Navigate to CMS | Edit, and wait a few minutes for the time you set to pass.
- 8. Navigate to CMS | Admin | Admin | Scheduled Jobs | Bad Word Scanner.
- 9. Click **History**, and note the job ran succesfully and censored three pages, as shown in the following screenshot:

ad word Scanner							
n for bad words in pag	es and cens	sor them.					
Settings History							
Date	Duration	Status	Server	Message			

10. Navigate to **CMS** | **Edit**, and note the Start page has been censored and marked as eing ready to publish, as shown in the following screenshot:

C Start[censored]					×	Ready to publish	Publish? 🗸
Name	×	Start[censored]	Visible to	Everyone			

- 11. Publish the censored change.
- 12. Check the About us and Alloy Track pages are similiarly censored and ready to publish.



Exercise F5 – Implementing soft and hard deletes

In this exercise, you will create a page type with template to allow anonymous visitors to enter a content reference of an item of content to delete.

Prerequisites: complete Exercise A1.

Adding the delete content feature

- 1. If you haven't done so already, extract the folders and files in cmsdevfun_exercisefiles.zip.
- 2. Drag and drop the \cmsdevfun-exercisefiles\Module F\F5\Features\ folder into the AlloyDemo project.
- 3. Expand the **Features** folder and review the files included, as shown in the following screenshot:
 - a. **DeleteContentPage.cs**: a page type class to enable the feature.
 - b. **DeleteContentPage.cshtml:** a Razor file for the user interface of the feature.
- - DeleteContent
 - C# DeleteContentPage.cs
 - [@] DeleteContentPage.cshtml
 - C* DeleteContentPageController.cs
 - [@]_ViewStart.cshtml
 - 🔁 Web.config
- c. **DeleteContentPageController.cs**: a controller that performs the work of the feature.
- 4. Open DeleteContentPageController.cs and note the following:
 - a. The **content repository service** set using constructor parameter injection. It will be used to either delete (if hard delete is "on") or move the content reference to the wastebasket.
 - b. The Delete action method that responds to HTTP POSTs, with three parameters: currentPage, contentReference, and hardDelete. If hardDelete is "on" then it uses the content repository's Delete() method to permanently delete the content identified with the contentReference, else it uses the Move() method to move the content to the Wastebasket. It sets a message in ViewData to tell the visitor what happened.
- 5. Open DeleteContentPage.cshtml and note the following:
 - a. If a message is passed using ViewData, it is shown to the visitor.
 - b. A form that allows a visitor to enter a content reference, select a check box for hard delete, and click a **Delete** button.

Test the website delete content functionality

- 1. Start the AlloyDemo website, and log in as Admin.
- 2. Navigate to CMS | Admin | Admin | Access Rights | Set Access Rights.
- 3. In the Set Access Rights content tree, select Recycle Bin.

By default, only members of the Windows group Administrators can work with the Recycle Bin! If you have not already fixed this issue, then you must give CmsAdmins full access rights.

- 4. Click Add Users/Groups.
- 5. Search for Groups, add the CmsAdmins group, and click OK.



6. Clear all access rights from **Administrators**, and assign all access rights to **CmsAdmins**, as shown in the following screenshot:

Set Access	Righ	ts for	"Recy	cle Bi	in"		(
Restore access rightems.	nts in EPi	iServer Cl	MS for iten	ns that yo	ou have, fo	r example, completely removed access to. You can chan	ge all rights on all
Root Start Recycle Bin For All Site: Customer Z	s Sone						
ୡ Add Users/Gr	oups						
	Read	Create	Change	Delete	Publish	Administer	
Administrators							
Everyone	\$						
CmsAdmins	*	•	◄	4	•	8	
	rom pare	nt item					
Innerit settings fr							

- 7. Click Save.
- 8. Create and publish a **Delete Content** page named **Deleter** under the **Start** page.
- 9. Create and publish a Standard page named Alloy Ahoy! under the Start page.
- 10. Make a note of the ID for Alloy Ahoy!, for example 115, as shown in the following screenshot:

Alloy Ahoy!			
Name	Alloy Ahoy!	Visible to	Everyone Manage
Name in URL	alloy-ahoy Change	Languages	en
Simple address	<u>Change</u>	ID, Type	115, Product
	✓Display in navigation		Tools 🗸

- 11. Switch to Live view and log out so that you are anonymous.
- 12. Navigate to the **Deleter** page.
- 13. Enter the ID of the Alloy Ahoy! page, click **Delete**, and note the message, as shown in the following screenshot:



14. Log in as Admin, and view the Trash, as shown in the following screenshot:

Trash			Empty Trash
Q Search			
Name	Removed	Ву	
🕒 Alloy Ahoy!		installer	Restore

- 15. Restore the Alloy Ahoy! page.
- 16. Switch to Live view and log out so that you are anonymous.
- 17. Navigate to the Deleter page.



18. Enter the ID of the Alloy Ahoy! page, select the **Hard delete** check box, click **Delete**, and note the message, as shown in the following screenshot:



- 19. Log in as Admin and confirm that the page is not in the Trash, or the Pages tree, and has been permanently deleted.
- 20. Close the browser.



Exercise F6 – Learning from Episerver's assemblies

In this exercise, you will use ILSpy to decompile and view Episerver assembly code to learn from it. **Prerequisites:** complete Exercise A1.

Decompiling Episerver assemblies to understand our APIs

- 1. Download and install ILSpy from: http://ilspy.net/
- 2. Choose File | Open, browse to AlloyDemo site's bin folder, select EPiServer.dll, and click Open, as shown in the following screenshot:

🞾 Open					\times
← → • ↑ <mark> </mark> « 1	Fraining → AlloyDemo → bin	ٽ ~	Search bin		9
Organize 🔻 New fol	der				?
This PC	Name	Date modified	Туре	Size	^
	EPiServer.Data.dll	09/06/2017 12:08	Application extens	543 KB	
	EPiServer.dll	09/06/2017 12:08	Application extens	2,916 KB	
Documents	EPiServer.Enterprise.dll	09/06/2017 12:08	Application extens	236 KB	
🕂 Downloads	EPiServer.Events.dll	09/06/2017 12:08	Application extens	170 KB	
🎝 Music 🗸	EPiServer.Framework.dll	09/06/2017 12:08	Application extens	515 KB	\sim
File	name: EPiServer.dll		 NET assemblies 		\sim
			<u>O</u> pen	Cancel	
			<u>O</u> pen	Cancel	

- 3. Expand EPiServer (11.n.n.0).
- 4. Expand EPiServer.Core.
- 5. Expand PageData.
- 6. Select PageName, and note that property is decompiled, as shown in the following screenshot:





7. In the decompiled source code, click **this.Name**, and note that property is decompiled, as shown in the following screenshot:



Reviewing Episerver initialization

- 1. In the left list, collapse EPiServer.Core.
- 2. Scroll down the tree view, expand EPiServer.Initialization.Internal, expand CmsCoreInitialization, and click Initialize(InitializationEngine) : void, and note some of the things that Episerver CMS does when it starts, as shown in the following screenshot:



You have now learned how to use tools like ILSpy to see how Episerver's own developer implement functionality. Follow their good practice, for example, in an initialization module use **context.Locate.Advanced.GetInstance<T>** to retrieve services using a dependency resolver.



Module G – Optimizing, Securing, and Deploying

Goal

The overall goal of the exercises in this module is to learn how to prepare an Episerver CMS website before deployment. You will:

- 1. Control the caching of responses in CDN and browser.
- 2. Implement logging.
- 3. Secure the AlloyDemo website.

Exercise G1 – Controlling the caching of responses

In this exercise, you will set cache-control headers and confirm that they were sent in the HTTP response to be read by a CDN and browser.

Prerequisites: complete Exercise A1.

 ${}^{m
u}$ The following instructions assume you are using Google Chrome. Other browsers have similar features.

- 1. Open the AlloyDemo project.
- 2. In Visual Studio, ensure that Google Chrome is set as the default browser for running your website.



- 3. Start the site by pressing Ctrl + F5 or navigate to Debug | Start Without Debugging.
- 4. In Google Chrome, press F12 to show the developer tools pane.
- 5. Click the Network tab.
- 6. Check the **Disable cache** check box to ensure requests will not be read from the browser's local cache, as shown in the following screenshot:



G)



7. Press *F5* to refresh the site's Start page. You will see all the HTTP requests recorded in the developer tools pane, as shown in the following screenshot:

🔴 🔿 🔳 🔽	View:	= 🔨 🗆	Preserve la	a 🖉 Di	sable ca	che 🗍 🗍 C)ffline Not	hrottling
• • •			, meserve ie	.g 5.				motanig
Filter	R	egex 💷 Hid	e data URLs					
All XHR JS CSS Im	ng Medi	a Font Doc	WS Mani	fest Othe	er			
200 ms	=	400 ms		600 ms		800	ms	1000 n
Name	Status	Туре	Initiator	Size	Time	Timeline – St	art Time	1.00 s 📥
localhost	200	document	Other	2.5 KB	58 ms			
bootstrap.css	200	stylesheet	(index):10	22.4 KB	24 ms			
bootstrap-responsi	200	stylesheet	(index):11	4.4 KB	20 ms			
editmode.css	200	stylesheet	(index):14	884 B	42 ms			
jquery.js	200	script	(index):16	95.8 KB	55 ms			
bootstrap.js	200	script	(index):17	12.9 KB	58 ms			
media.css	200	stylesheet	(index):12	1.3 KB	11 ms	1		
logotype.png	200	png	(index):31	3.0 KB	27 ms			
style.css	200	stylesheet	(index):13	3.7 KB	12 ms			
🗄 alloymeetbanner.p	200	png	<u>(index):80</u>	578 KB	63 ms			
alloyplan.png	200	png	<u>(index):94</u>	32.0 KB	31 ms			
 alloytrack.png 	200	png	(index):101	13.0 KB	51 ms			
 alloymeet.png 	200	png	(index):108	32.4 KB	51 ms			
glyphicons-halfling	200	png	<u>(index):199</u>	13.9 KB	39 ms	•		
searchbuttonsmall	200	png	(index):199	2.3 KB	37 ms			

Note that the Status code returned is **200** OK for all requests. Note the load time was 275 ms and 819 KB was transferred over 15 requests for resources.

8. Uncheck the **Disable cache** check box to allow requests to be read from the browser's local cache.



9. Press *F5* to make the same request, and note the differences in the developer tools pane, as shown in the following screenshot:

Elements	Conso	le Sources	Network	Timeline	Profiles	Appli	cation	>>	:	×
● ◎ ■ 7	View:	i 🔨 🛛	Preserve	log 🔲 Dis	able cache		Offline	No th	rottling	J
Filter	R	egex 🔲 Hic	de data URLs	;						
All XHR JS CSS In	ng Med	ia Font Do	c WS Man	ifest Othe	r					
200 ms		400 ms		600 ms		800) ms		10	000 ms
=										
Name	Status	Туре	Initiator	Size		Time	Timelin	e – Star	t Time	
localhost	200	document	Other		2.5 KB	63 ms				
bootstrap.css	200	stylesheet	(index):10	(from	disk cache)	7 ms				
style.css	200	stylesheet	(index):13	(from	disk cache)	5 ms	1			
bootstrap-responsi	200	stylesheet	(index):11	(from	disk cache)	4 ms	1			
media.css	200	stylesheet	(index):12	(from	disk cache)	5 ms				
editmode.css	200	stylesheet	(index):14	(from	disk cache)	4 ms	- L			
jquery.js	200	script	(index):16	(from	disk cache)	9 ms	- E			
bootstrap.js	200	script	(index):17	(from	disk cache)	6 ms	- L			
logotype.png	200	png	(index):31	(from men	nory cache)	0 ms	1			
alloymeetbanner.p	200	png	(index):80	(from men	nory cache)	0 ms	1			
- alloyplan.png	200	png	(index):94	(from men	nory cache)	1 ms	1			
- alloytrack.png	200	png	(index):101	(from men	nory cache)	0 ms	1			
 alloymeet.png 	200	png	(index):108	(from men	nory cache)	0 ms	1			
glyphicons-halfling	200	png	(index):199	(from men	nory cache)	0 ms				
searchbuttonsmall	200	png	(index):199	(from men	nory cache)	0 ms				
-										

15 requests | 2.5 KB transferred | Finish: 199 ms | DOMContentLoaded: 221 ms | Load: 220 ms

Note that the Status code returned is **200** OK for all requests. Note the load time was 220 ms and 2.5 KB was transferred over 15 requests for resources. Only the request for **localhost** was sent from the web server, all other requests were handled from either disk cache (for stylesheets and scripts) or memory cache (for images) in the browser.

Viewing the HTTP headers for three types of response

The 15 resources used on the Start page can be divided into three types:

- Dynamically-generated content: localhost
- **Static application files**: bootstrap.css, style.css, bootstrap-responsive.css, media.css, editmode.css, jquery.js, bootstrap.js, glyphicons-halflings.png, searchbuttonssmall.png
- Static asset content: logotype.png, alloymeetbanner.png, alloyplan.png, alloymeet.png, alloytrack.png

By default, each of the three types is treated differently for caching purposes. This is something you will likely want to configure to optimize for your site.

 In the list of requests, click the first one, named localhost, and ensure that the Headers tab is selected. This response is dynamically generated at runtime from an MVC view. Note the Cache-Control header is set to private, meaning that only the browser could store it in its cache, but CDNs



would not cache it, and there is no **Expires** value after the **Date** header, so the browser won't cache it either, as shown in the following screenshot:

Name	× Headers Preview Response Cookies Timing						
localhost	▼ General						
bootstrap.css	Request URL: http://localhost:60540/						
bootstrap-responsive.css	Request Method: GET						
media.css	Status Code: 200 OK Remote Address: [::11:60540						
style.css	Response Headers view source						
logotype.png	Cache-Control: private						
* alloymeetbanner.png	Content-Encoding: gzip						
editmode.css	Content-Length: 2391						
alloyplan.png	Content-Type: text/html; charset=utf-8						
jquery.js	Date: Sun, 27 Nov 2016 16:44:05 GMT Server: Microsoft-IIS/10.0						

Good practice would be to change the default from private to public and consider if and for how long to cache based on the frequency that the dynamically-generated content changes.

2. In the list of requests, click the one named style.css, and ensure that the Headers tab is selected. This response is static and loaded from the web server's file system. Note that the Cache-Control header is set to max-age=86400, meaning that the browser or any intermediaries like CDNs can store it for up to one day (the value is in seconds), as shown in the following screenshot:



Common max-age values are:

- One minute: max-age=60
- One hour: max-age=3600
- One day: max-age=86400
- One week: max-age=604800
- One month: max-age=2628000
- One year: max-age=31536000

Good practice would be to change the default from one day to one year.

3. In the list of requests, click the one named **alloyplan.png**, and ensure that the **Headers** tab is selected. Note the **Cache-Control** header is set to **public**, meaning that the browser and any



intermediaries can store it until it expires. Its **Expires** header is 12 hours after the **Date** header, as shown in the following screenshot:

Name	× Headers Preview Response Timing
localhost	▼ General
bootstrap.css	Request URL: http://localhost:51343/globalassets/alloy-p
bootstrap-responsive.css	lan/alloyplan.png
media.css	Status Code: 200 DK (from memory cache)
style.css	Remote Address: [::1]:51343
editmode.css	▼ Response Headers
logotype.png	Accept-Ranges: bytes
jquery.js	Cache-Control: public
✤ alloymeetbanner.png	Content-Length: 32335
bootstrap.js	Content-Type: 1mage/png
- alloyplan.png	ETag: "1027BA80C6A2210"
alloytrack.png	Expires: Wed, 01 Feb 2017 03:05:07 GMT
> allovmeet.ong	Last-Modified: Tue, 31 Jan 2017 09:54:44 GMT

4. Close the browser.

Controlling caching of dynamically-generated content with code

For total control of how dynamically-generated content is cached, use code.

- 1. Open ~\Controllers\StartPageController.cs.
- 2. Modify its Index method to add the following three statements before returning the view:

```
Response.Cache.SetCacheability(System.Web.HttpCacheability.Public);
Response.Cache.SetExpires(System.DateTime.Now.AddHours(1));
Response.Cache.SetSlidingExpiration(true);
```

- 3. Start the site.
- 4. View the recorded network requests in the developer tools pane. Note that the Cache-Control header is set to public, meaning that both the browser and CDNs can cache it, and there is an Expires value set one hour after the Date controlling how long the resource will be cached, as shown in the following screenshot:

Name	× Headers Preview Response Cookies Timing					
localhost	▼ General					
bootstrap.css	Request URL: http://localhost:51343/					
bootstrap-responsive.css	Request Method: GET Status Code:					
media.css						
style.css						
logotype.png						
editmode.css	Content-Encoding: gzip					
🔹 alloymeetbanner.png	Content-Length: 2214					
jquery.js	Content-Type: text/html; charset=utf-8					
bootstrap.js	Date: Tue, 31 Jan 2017 15:39:22 GMT					
- alloyplan.png	Server: Microsoft-IIS/10.0					

- 5. Close the browser.
- 6. To avoid confusion in later exercises, open ~\Controllers\StartPageController.cs, and comment out the three statements that enabled caching.

Controlling caching of dynamic content with attributes and configuration

A simpler, but less flexible and powerful, alternative to writing code is to use a combination of configuration and attributes.

1. In the ~\Controllers folder, open DefaultPageController.cs.

This controller is used for all page types in Alloy that do not have their own specific controller.



2. Apply the [ContentOutputCache] attribute *before* the **Index** method:

[EPiServer.Web.Mvc.ContentOutputCache]
public ViewResult Index(SitePageData currentPage)

- 3. Open ~\Web.config.
- 4. Find the <episerver> section. Inside the <applicationSettings> element, add the attribute httpCacheExpiration and set its value to a time span of two hours (02:00:00). Note the httpCacheability attribute value is already Public, but this is ignored unless the [ContentOutputCache] attribute is applied as an action filter. The defaults set in configuration can be overridden in a [ContentOutputCache] attribute if necessary.

```
<episerver>
   <applicationSettings
    httpCacheability="Public"
    httpCacheExpiration="02:00:00"
    ...</pre>
```

- 5. Start the site.
- 6. View the recorded network requests in the developer tools pane for the About us page and note the HTTP response headers have been affected, for example, Expires is two hours after Date, and maxage is 7200 seconds (2 hours), as shown in the following screenshot:

Name	× Headers Preview Response Cookies Timing					
about-us/	▼ General					
bootstrap.css	Request URL: http://localhost:51343/about-us, Request Method: GET Status Code: ● 200 OK Remote Address: [::1]:51343					
bootstrap-responsive.css						
media.css						
style.css	Perpansa Handers Jian source					
editmode.css	<pre>* Kesponse Headers view source Cache-Control: public, max-age=7200 Content-Encoding: gzip Content-Length: 2864 Content-Type: text/html; charset=utf-8 Date: Tue, 31 Jan 2017 15:49:36 GMT</pre>					
jquery.js						
bootstrap.js						
logotype.png						
polarbearonice.png						
children.png	Last-Modified: Tue, 31 Jan 2017 17:49:36 GMT					
• teaser_contactus.png	Server: Microsoft-IIS/10.0					

- 7. Close the browser.
- 8. To avoid confusion in later exercises, open ~\Controllers\DefaultPageController.cs, and comment out the [ContentOutputCache] that enabled caching.

Controlling caching of static content

- 1. Open ~\Web.config.
- At the top of the file, add the following <section> element inside the <configSections> element:

```
<section name="staticFile" allowLocation="true" type=
"EPiServer.Framework.Configuration.StaticFileSection, EPiServer.Framework.AspNet" />
```

```
Breaking change in CMS 11
Assembly name has changed to EPiServer.Framework.AspNet. For CMS 10, use EPiServer.Framework.
```

3. Add the following element after the </configSections> element. It will set the expiration to 3 hours of any static media assets managed by the CMS, such as images, videos, and so on.
<staticFile expirationTime="03:00:00" />

In this exercise, you set the expiration to three hours, but a year would be better practice in real life.



- 4. Start the site.
- 5. Show the developer tools by pressing *F12*.
- 6. Check the **Disable cache** check box, and then press *F5* to refresh the previously cached for 12 hours static content.
- 7. Uncheck the **Disable cache** check box, and then press *F*5.
- 8. View the recorded network requests in the developer tools pane for the **alloyplan.png** image on the Start page and note the HTTP response headers have been affected. The difference between the **Date** and **Expires** will now be 3 hours instead of 12 hours, as shown in the following screenshot:

Name	× Headers Preview Response Cookies Timing					
Iocalhost	▼ General					
bootstrap.css	Request URL: http://localhost:51343/globalasset s/alloy-plan/alloyplan.png Request Method: GET Status Code: @ 200 OK					
bootstrap-responsive.css						
media.css						
style.css	Remote Address: [::1]:51343					
editmode.css	▼ Response Headers view source					
jquery.js	Accept-Ranges: bytes Cache-Control: public					
bootstrap.js						
Iogotype.png	Content-Length: 32335					
🔹 alloymeetbanner.png	Content-Type: image/png					
– alioyplan.png	ETag: "1027BA80C6A2210"					
- alloytrack.png	Expires: Tue, 31 Jan 2017 18:57:00 GMT Last-Modified: Tue, 31 Jan 2017 09:54:44 GMT					
→ alloymeet.png						
aluphicons halflings and	Server: Microsoft-IIS/10.0					

9. Close the browser.

Controlling caching of static application files

- 1. Find the <system.webServer> element.
- Inside it, find the <staticContent> element that contains a <clientCache> element with cacheControlMode set to UseMaxAge and a cacheControlMaxAge set to a time span of 1 day, as shown in the following configuration:

```
<staticContent>
<clientCache cacheControlMode="UseMaxAge" cacheControlMaxAge="1.00:00:00" />
```

3. Change the number of days to **365**. This will set the expiration to one year for any static application files that are part of your site, such as images, JavaScript files, CSS stylesheets, and so on.

When you set a "far future" max-age value for static application resources, you should include a version number or date as part of the filename, e.g. change style.css to style-2017-01-31.css or jquery.js to jquery-3.1.1.js.

- 4. Start the site.
- 5. Show the developer tools by pressing *F12*.
- 6. Check the **Disable cache** check box, and then press *F5* to refresh the previously cached for 1 day static application files.
- 7. Uncheck the **Disable cache** check box, and then press F5.
- View the recorded network requests in the developer tools pane for the style.css file on the Start page and check that the HTTP response headers have been affected. The max-age is now 31536000 seconds (one year) instead of 86400 seconds (one day), as shown in the following screenshot:



Name	× Headers Preview Response Timing			
localhost	▼ General			
bootstrap.css	Request URL: http://localhost:51343/Static/css/style.css Request Method: GET Status Code: 200 OK (from disk cache) Remote Address: [::11:51343			
media.css				
bootstrap-responsive.css				
style.css	Response Headers			
editmode.css	Accept-Ranges: bytes			
jquery.js	Cache-Control: max-age=31536000			
bootstrap.js	Content-Encoding: gzip			
logotype.png	Content-Length: 3344			
alloymeetbanner.png	Content-Type: text/css			
- alloyplan.png	ETag: "42272baea67bd21:0"			
- alloytrack.png	Last-Modified: Tue, 31 Jan 2017 09:44:56 GMT			

9. Close the browser.

Bundling, minimization, and cache busting

In the screenshot above, note that the CSS files needed for the page are five separate HTTP requests.

- 1. Find the <system.web> element.
- 2. Inside it, find the <compilation> element that contains a **debug** attribute set to **true**, as shown in the following configuration:

- 3. Change the **debug** attribute to **false**.
- 4. Start the site.
- 5. Show the developer tools by pressing *F12*.
- 6. Press F5.
- 7. View the recorded network requests in the developer tools pane and note the CSS files (and JavaScript files) have been bundled into single requests, as shown in the following screenshot:

Name	Status	Туре	Initiator	Size
localhost	200	document	Other	3.1 KB
css?v=IOTLf70WKv4gQuk9DGPncjai2nEog8pxNJx9rWSknpA1	200	stylesheet	(index)	(from disk cache)
js?v=5x_bEnqiEO2lzoxxfpj_tyxcOY5c0JQS8e_OpUn4XQ1	200	script	(index)	(from disk cache)
logotype.png	200	png	(index)	(from memory cache)
1 alloymeetbanner.png	200	png	(index)	(from memory cache)
alloyplan.png	200	png	(index)	(from memory cache)
alloytrack.png	200	png	(index)	(from memory cache)
∞ alloymeet.png	200	png	(index)	(from memory cache)
find.js	200	script	(index)	(from disk cache)
glyphicons-halflings.png	200	png	(index)	(from memory cache)
searchbuttonsmall.png	200	png	(index)	(from memory cache)

8. Click the JavaScript bundle, **js?v=...**, and then click the **Response** tab, and note that as well as bundling multiple requests for JavaScript together, the files have been minimized, as shown in the following screenshot:

Name	×	He	eaders Preview	Response	Timing	
localhost	1	1	(function(n,t)	{function	n nu(n){var	<pre>i=yt[n]={},t,r;for(n=n.split(/\s+/),t=0,</pre>
css?v=IOTLf70WKv4gQuk9DGPncj						
js?v=5x_bEnqiEO2lzoxxfpjtyxcOY						



If any of the files in a bundle changes, ASP.NET automatically changes the hash value (v=...) so that the cached bundle is "busted" and the change downloaded. You could implement a similar "cache busting" technique in combination with "far future" cache control headers for static content.

- 9. Close the browser.
- Open ~\Business\Initialization\BundleConfig.cs, and note the RegisterBundles method that determines which static application files are included in the two bundles, as shown in the following code:

Creating a page type and view model for the object cache

You will create a page type with a page template that will show the contents of the Episerver object cache.

```
This page only exists for demonstrating the idea. In a real site, you would implement this as a plug-in or gadget.
```

- 1. In ~\Models\Pages, add an Episerver | Page Type named ObjectCachePage.cs.
- 2. Modify the contents, as shown in the following code:

```
namespace AlloyDemo.Models.Pages
{
    [SiteContentType(DisplayName = "Object Cache",
        GroupName = Global.GroupNames.Specialized,
        Description = "View the contents of the object cache.")]
    [SiteImageUrl]
    [AvailableContentTypes(IncludeOn = new[] { typeof(StartPage) })]
    public class ObjectCachePage : SitePageData
    {
     }
}
```

- 3. In ~\Models\ViewModels, add a class named ObjectCachePageViewModel.cs.
- 4. Modify the contents, as shown in the following code:

```
using AlloyDemo.Models.Pages;
using System.Collections;
using System.Collections.Generic;
namespace AlloyDemo.Models.ViewModels
{
    public class ObjectCachePageViewModel
        : PageViewModel<ObjectCachePage>
    {
        public IEnumerable<DictionaryEntry> CachedItems { get; set; }
        public string FilteredBy { get; set; }
        public ObjectCachePageViewModel(
            ObjectCachePageViewModel(
            ObjectCachePage currentPage) : base(currentPage)
```



{ } }

}

Creating a page template for the object cache

```
60
   When creating controllers in the Alloy site, you should inherit from the site-specific class named
   PageControllerBase<T>, instead of the usual Episerver class named PageController<T>.
    1. In ~\Controllers, add an Episerver | Page Controller (MVC) named ObjectCachePageController.cs.
    2. Modify the contents, as shown in the following code:
using AlloyDemo.Models.Pages;
using AlloyDemo.Models.ViewModels;
using EPiServer;
using EPiServer.Core;
using EPiServer.Web.Mvc;
using System.Collections;
using System.Linq;
using System.Web.Mvc;
namespace AlloyDemo.Controllers
{
    public class ObjectCachePageController
        : PageControllerBase<ObjectCachePage>
    {
        public ActionResult Index(ObjectCachePage currentPage, string filterBy)
         {
             var viewmodel = new ObjectCachePageViewModel(currentPage);
             var cachedEntries = HttpContext.Cache.Cast<DictionaryEntry>();
             switch (filterBy)
             {
                 case "pages":
                     viewmodel.CachedItems = cachedEntries
                          .Where(item => item.Value is PageData);
                     break;
                 case "content":
                      viewmodel.CachedItems = cachedEntries
                          .Where(item => item.Value is IContent);
                      break;
                 default:
                      viewmodel.CachedItems = cachedEntries;
                      break;
             }
             viewmodel.FilteredBy = filterBy;
             return View(viewmodel);
        }
    }
}

    In ~\Views, create a new folder named ObjectCachePage.

    4. In ~\Views\ObjectCachePage, create a file named Index.cshtml.
```

5. Modify its contents, as shown in the following markup:

```
@using System.Collections
@using EPiServer.Core
```



```
@using EPiServer.Web.Mvc.Html
@model AlloyDemo.Models.ViewModels.ObjectCachePageViewModel
<div>
   <div class="alert alert-info">Object Cache</div>
   <div class="well well-small">
       @Html.ContentLink("All Cached Objects", Model.CurrentPage.ContentLink, null,
         htmlAttributes: new { @class = string.IsNullOrWhiteSpace(Model.FilteredBy)
? "btn btn-warning" : "btn" })
       @Html.ContentLink("Any Content", Model.CurrentPage.ContentLink,
          routeValues: new { filterBy = "content" },
         htmlAttributes: new { @class = Model.FilteredBy == "content"
                              ? "btn btn-warning" : "btn" })
       @Html.ContentLink("Pages Only", Model.CurrentPage.ContentLink,
         routeValues: new { filterBy = "pages" },
         htmlAttributes: new { @class = Model.FilteredBy == "pages"
                              ? "btn btn-warning" : "btn" })
   </div>
   Key
           Type
           @(string.IsNullOrWhiteSpace(Model.FilteredBy)
                     ? "Value" : "Name (ID) Published")
       @foreach (DictionaryEntry item in Model.CachedItems)
       {
          >
              @item.Key
              @item.Value.GetType()
              @if (item.Value is IContent)
                  {
                     @((item.Value as IContent).Name)
                     <span class="badge badge-warning">
                         @((item.Value as IContent).ContentLink.ID)</span>
                  @if (item.Value is PageData)
                     @((item.Value as PageData).StartPublish)
              }
   </div>
```

Creating an object cache page in the page tree

- 1. Start the site, and log in as Admin.
- 2. Underneath the Start page, add an Object Cache page named Cache.
- 3. Publish the page.
- 4. View the site as a visitor.



5. Click **Any Content**, as shown in the following screenshot, and note the key values that Episerver uses for content, for example, the **Alloy Track teaser** block has a key of **EPPageData:55**:

Cache - NOT FOR COMM	×	Mark		×
	10st:54739/cache/?filterBy=content		☆	:
Alloy	Start Alloy Plan Alloy Track Alloy Meet About us	Cache Search	٩	
All Cached Objects	Object Cache			
Кеу	Туре	Name (ID) Published		
EPPageData:35	Castle.Proxies.StandardPageProxy	Management 35 8/22/2012 5:19:48 PM		
EPPageData:151	Castle.Proxies.ImageFileProxy	AlloyPlan.png 151		
EPPageData:152	Castle.Proxies.ImageFileProxy	AlloyTrack.png 152		
EPPageData:35:en	Castle.Proxies.StandardPageProxy	Management 35 8/22/2012 5:19:48 PM		
EPPageData:150	Castle.Proxies.ImageFileProxy	AlloyMeetBanner.png 150		
EPPageData:55	Castle.Proxies.TeaserBlockProxy	Alloy Track teaser 55		
EPPageData:49:en	Castle.Proxies.LandingPageProxy	Reseller extranet (49) 10/10/2012 8:53:43 AM		-



Exercise G2 – Implementing logging

In this exercise, you will implement logging an invalid property setting by using Log4net.

Prerequisites: complete Exercise A1.

Reviewing the default logging configuration

- 1. Open the Training solution with the AlloyDemo project.
- 2. Open the ~/EPiServerLog.config file.
- 3. Note the configuration of the first <appender> element:
 - It uses the **RollingFileAppender** type with a **rollingStyle** or **Date** and a pattern of **yyyyMMdd** so that each day a new log file will be created with the date as part of the filename.
 - The files will be stored in the App_Data folder.
 - Each entry will include the date, thread, level, and message.

4. Note the enabling of logging at Error level.

```
<root>
    <!-- setting this value to All, Debug or Info will affect performance.-->
    <level value="Error" />
    <!--Enabled file logging-->
        <appender-ref ref="errorFileLogAppender" />
```

Writing to the logger

- 1. Open the ~/Controllers/StartPageController.cs file.
- 2. Import namespaces for logging, by adding the following statements to the top of the file:

```
using EPiServer.Core;
using EPiServer.Logging;
```

3. Inside the class, declare a field to store a reference to the registered **ILogger** service, as shown in the following code:

```
private readonly ILogger logger = LogManager.GetLogger();
```

4. Inside the **Index** method, before creating the page view model, add a statement to write to the log if a search page link has not been set for the StartPage, as shown in the following code:

```
if (PageReference.IsNullOrEmpty(currentPage.SearchPageLink))
{
    logger.Error("No 'Search page' is specified in 'Site settings'.");
}
```



5. In the Solution Explorer, click Show All Files, and then delete the ~/App_Data/EPiServerErrors.log file to clear any existing logs for today.

Testing the logging

- 1. Start the site, and log in as Admin.
- 2. Edit the Start page, and switch to All Properties view.
- 3. Click the **Site settings** tab.
- 4. Clear the **Search page** property, as shown in the following screenshot:

Name	Start
Name in URL	start Change
Simple address	Change
	Display in navigation
SEO Site	settings Content Settings
Logotype	
Url	logotype.png 😵
Title	Alloy - collaboration, con
Search page	
Contact pages	Contacts 🛛 🚷

- 5. **Publish** the changes to the Start page.
- 6. View the Start page as a visitor.
- 7. Close the browser.

Reviewing the log

- 1. In Solution Explorer, refresh the ~\App_Data folder, and then open the EPiServerErrors.log file.
- 2. If the file contains lots of log entries, press *Ctrl* + *F* to find, type **StartPageController** and press ENTER to find the log entry.
- 3. Note the log entry, as shown in the following screenshot:

```
2017-01-05 12:37:45,586 [54] ERROR
AlloyDemo.Controllers.StartPageController: No 'Search page' is specified in
'Site settings'.
```

```
      EPiServerErrors.log + X
      StartPageController.cs

      1399
      at System.Web.HttpApplication.txecuteStep(ltxecutionStep step, Boolean& completedSynchronou)

      1400
      2017-01-05
      12:37:40,853
      [39] ERROR EPiServer.Search.Internal.RequestHandler: Update batch coul

      1401
      at EPiServer.Search.Internal.RequestHandler.MakeHttpRequest(String url, NamedIndexingServic)
      Aa A H * CurrentDocument

      1402
      at EPiServer.Search.Internal.RequestHandler.SendRequest(SyndicationFeed feed, String namedI

      1403
      2017-01-05
      12:37:40,888
      [39] ERROR EPiServer.Search.Internal.RequestHandler: SendRequestGueueHandler: Send batch for named index 'serviceName' failed. Iter

      1404
      2017-01-05
      12:37:40,586
      [54] ERROR EPiServer.Gontrollers.StartPageController: No 'Search page' is specified in 'Site settings'.

      1405
      2017-01-05
      12:38:10,962
      [60] ERROR EPiServer.Global:
      1.2.5

      1405
      2017-01-05
      12:38:10,962
      [60] ERROR EPiServer.Global:
      1.2.5

      1405
      2017-01-05
      12:38:10,962
      [60] ERROR EPiServer.Global:
      1.2.5
```

Restore the Search page

- 1. Start the site, and log in as Admin.
- 2. Edit the **Start** page, and switch to **All Properties** view.



- 3. Click the Site settings tab.
- 4. Set the Search page property to the Search page, as shown in the following screenshot:

Select Page		×
Q Search		
🖬 🗋 Root		
🗉 🗋 Start		
🖪 🗋 Alloy Plan		
🖪 🗋 Alloy Track		
🖬 🗋 Alloy Meet		
🖪 🗋 About us		
🖬 🚍 How to buy		
🖪 🚍 Campaigns		
🕒 Search		
🖪 📄 Customer Zone		
	Current Page OK C	ancel

- 5. **Publish** the changes to the Start page.
- 6. Close the browser.

You've now added basic logging to one page type used by the site.



Exercise G3 – Securing an Episerver site

In this exercise, you will change the Episerver URL path to implement security through obscurity. Prerequisites: complete Exercise A1.

Reviewing the default Episerver URL path

- 1. Open the AlloyDemo project.
- 2. Start the site.
- In the browser's address bar, enter: EPiServer/ at the end of the URL, as shown in the following screenshot:



4. You will be redirected to the log in page, as shown in the following screenshot:



5. Enter a name of Admin and a password of **Pa\$\$wOrd**, click **Log** in, and you will see the **Global** menu and **Dashboard**, as shown in the following screenshot:



6. Close the browser.

You will now modify the site's configuration to change the URL path.



Changing the Episerver URL path

- 1. Open ~/Web.config.
- 2. Find the <applicationSettings> element, and modify the **uiUrl** attribute to ~/Secret/CMS, as shown in the following partial configuration:

```
<episerver>
  <applicationSettings
    ...
    uiSafeHtmlTags="b,i,u,br,em,strong,p,a,img,ol,ul,li"
    httpCacheability="Public"
    uiEditorCssPaths="~/Static/css/Editor.css"
    uiShowGlobalizationUserInterface="true"
    uiMaxVersions="20"
    uiUrl="~/Secret/CMS/" />
```

3. Find the <location path="EPiServer"> element, and modify the **path** attribute to **Secret**, as shown in the following configuration:

```
<location path="Secret">
```

4. Find the <location path="EPiServer/CMS/admin"> element, and modify the path attribute to Secret/CMS/Admin, as shown in the following partial configuration:

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

- 5. Use Visual Studio's Find feature (press *Ctrl* + *F*) to search for ~/EPiServer.
- 6. Modify the two ~/EPiServer entries to ~/Secret, as shown in the following configuration:

Breaking change in CMS 11

Assembly name has changed to EPiServer.Framework.AspNet. For CMS 10, use EPiServer.Framework.

7. Start the site, and log in as Admin.



8. All the menus in the **Global** menu should now use **Secret** as the URL path, as shown in the following screenshot:



This is the end of the exercises book.



Summary of Attributes in Episerver

Attributes for Content Types

Name	Parameters	Description
ContentType	DisplayName Description GroupName Order GUID AvailableInEditMode	Controls the registration of a content type in the CMS database.
Access	Roles Users others	Controls who can create an instance of this type.
ImageUrl	Path	Sets a 120 x 90 preview icon.
AvailableContentTypes	Availability Include IncludeOn Exclude ExcludeOn	Controls which content types can be this content type's parent and children.
MediaDescriptor	Extensions ExtensionString	Controls which file extensions are associated with this media type.

Attributes for Properties

Name	Parameters	Description
Display	Name Description GroupName Order	Controls the registration of the property in the CMS database.
AllowedTypes	AllowedTypes RestrictedTypes others	Controls which content types can be referenced by the property.
CultureSpecific		Allows the property to have language branches.
Seachable		Includes the property in the search index.
Editable	true	
Ignore		Prevents property from being stored in CMS database.
ScaffoldColumn	false	Hides the property from Editors.
UIHint	uiHint presentationLayer	For visitors: selects a DisplayTemplate. For editors: selects a custom property editor.
SelectOne	SelectionFactory	Editors can select one value from a drop-down listbox.
SelectMany	SelectionFactory	Editors can select many values from a list of check boxes.
Required, Range, StringLength, RegularExpression, and other ValidationAttribute- derived attributes	various	Validation rules are applied when saving and publishing property.



Attributes for Controllers

Name	Parameters	Description
TemplateDescriptor	Name Description Inherited Default Tags, TagString AvailableWithoutTag ModelType Path TemplateTypeCategory	Controls the registration of a content template in the CMS database.
ContentOutputCache	Duration (seconds) Location others	Controls how long and where the response is cached. Can also be applied to action method(s).

Attributes for Groups and Tabs

Name	Parameters	Description
GroupDefinitions	n/a	Apply to a static class with string constants to define Tabs in code.
Display	Name Order	Controls the registration of the tab names in the CMS database.
RequiredAccess	AccessLevel	Controls what access rights an Editor must have to see a tab and its properties.

Attributes for Initialization Modules

Name	Parameters	Description
InitializableModule	UninitializeOnShutdown	Apply to a class to register it as an initialization module. It must implement IInitializableModule or IConfigurableModule.
ModuleDependency	One or more types.	Controls the order in which initialization modules execute by defining dependencies.



Attributes for Extensions

Name	Parameters	Description
EditorDescriptorRegistration	TargetType UIHint EditorDescriptorBehavior	Registers a class as a custom editor for a data type.
UIDescriptorRegistration	n/a	Registers a class to customize UI elements for a content type.
Component	Title Description PlugInAreas SortOrder AllowedRoles Categories	Registers a class as a gadget for an Editor or Admin to add to their Dashboard or Edit view Navigation and Assets panes.
GuiPlugIn	Area Category SortIndex DisplayName Description DefaultEnabled RequiredAccess	Registers a class as a plug-in. Category is only used by Visitor Group Criteria
ScheduledPlugIn	DisplayName Description InitialTime IntervalLength IntervalType Restartable SortIndex GUID	Registers a class as a scheduled job. It can optionally inherit from ScheduleJobBase or have a static method named Execute that returns a string.