Sitecore Training

Sitecore® Experience Platform™ 7.2
Course Manual

Website Development for .NET Developers

sitecore
Own the experience™

Welcome

Thank you for attending this 4 day course which teaches participants how to work as an effective .NET Developer on a Sitecore project.

This course is intended for .NET web developers with advanced C# skills who are new to the Sitecore API and wish to develop websites using Sitecore Recommended Practices.

Foreword

The original goal of this course was not only to teach developers the fundamentals of building a Sitecore site, but also to do so in accordance with recommended practices. We wanted to give developers the tools to support both visitor and business user needs, by building solutions that take full advantage of the platform's features and can scale to meet the demands of the business. In this release, we have streamlined the fundamentals and reemphasized the importance of understanding what the framework has to offer, and how to get the most out Sitecore as an Experience Platform.

Martina Welander, September 2014, Bristol UK

Martina Welander

Led Development Project

Martina Helene Welander is a Technical Consulting Engineer at Sitecore and (strangely) a Historical Studies graduate from Bristol University. At Sitecore, she has worked tirelessly to create and complete dozens of projects that have helped define the way users see Sitecore and ultimately, how our clients see their own businesses and marketing strategies.

Martina is a technical chameleon, with vast and varied knowledge of Sitecore, C#, front-end development, and user experience. She blogs for Sitecore at **http://www.sitecore.net/Learn/Blogs/Technical-Blogs/Martina-Welander-Sitecore-Blog.aspx**, and initiated a Twitter account for Sitecore Education Services at **https://twitter.com/mastersitecore**. Martina also updates her own Twitter feed with her trademark humour at **https://twitter.com/mhwelander**.

In her spare time, Martina slays zombies in Minecraft, makes timely use of Ætherize in Magic the Gathering, bakes, runs, and plays play-by-post RPGs.

Michelle White

Provided Domain Expertise
Greatly Influenced Course Restructuring

Michelle White is a Technical Architect and Instructor who joined the Sitecore USA team in early 2012. She enjoys sharing Sitecore knowledge with many different audience groups. Michelle's experiences as an IT consultant, and a passionate instructor since the early 90's, extend across multiple platforms and applications, including Microsoft, resulting in multiple areas of expertise, MCTs, and Master Microsoft Instructor achievements.

Michelle enjoys ridding Middle Earth (LOTRO) of orcs, football, gardening, reading, and is an accomplished artist.

Sen Gupta

Provided Domain Expertise
Greatly Influenced Course Restructuring

Sen Gupta has been innovating at Sitecore since 2011. A certified Microsoft trainer, he has been developing software and training on Microsoft technologies for 10+ years. Despite Sen's sincere belief that WND truly enables developers with all that it offers today, his obsession to continually improve it will not rest. When he is not working hard for Sitecore Sen loves to cook and race his latest prized possession a KTM Duke 390.

Atousa Memarpouri

Provided Creative Direction
Drove Overall Look and Feel

Atousa comes from a creative background and has years of hands on experience with Content Management Systems. Her favourite hobbies include reading, cooking and keeping fit by running about after her three beloved pets.

Dan Zhang

Built Course Application

Dan Zhang is a techie with a passion for enabling others. Dan has over five years of experience as a web developer, with a number of specialities including HTML, SQL, Javascript, C# etc. She is also an accomplished academic with a Bachelor's degree in Software Engineering, and a Master's degree in IT with Security from University of Bristol.

She loves travel, design and painting in her spare time.

Wendy Lawn

Tested Labs

Wendy has over 10 years of experience in the field, including 8 years developing computer based training for military contracts. She is driven by a passion for good design, technology and the future of learning technologies.

Susan Marenco

Copy Edited Course
Supported Formatting Efforts
Provided Final Polish

For the last 15 years, Susan Marenco has worked as a writer and editor for Microsoft Development Centre Copenhagen, Egmont Publishing, UNICEF and many other companies.

Course Participants Prerequisites

In addition to IT professional skills, learners should also have experience in:

- Sitecore user tools comprising Content Editor, Page Editor and Desktop and Sitecore user tasks including Publishing and Workflows (Sitecore Foundations Pre-Learning Video).
- Programming using .Net WebForms and C#.
- Working with Microsoft Visual Studio 2012 or higher.
- Working and interacting with Microsoft SQL Server.

Sitecore Certified Instructors

Your instructor is a Sitecore technical and instructional expert who has met rigorous standards to be able to deliver your course.

Sitecore Certification

Sitecore certification is a valuable credential that demonstrates your skills in the Sitecore technology. To earn certification you must complete and pass the exam at the end of your training. You will need to register on our https://sdn.sitecore.net, the Sitecore SDN portal, to complete the assessment.

Post Course Assessment

Sitecore values your opinion so we would appreciate it if you would fill in an assessment questionnaire at the end of your training.

Icons used in this manual

Important

Important information or warnings. Read carefully.

Recommended / Best Practices

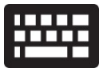
Recommended practice notice. Adhere whenever possible

Tip / Note

A key feature that is highlighted.

Demo / Walkthrough

Provide links and references to further material relating to exercise in the module.



Apply

Provide exercises relating to the material in the module.

Knowledge Check

Master the covered concepts and terminology.

Contents

Module 1
Sitecore Overview

Contents:

- Course Overview
- What Is Sitecore?
- Sitecore from an Author's Perspective
- Sitecore from a Developer's Perspective

Topic 1.1 Course Overview

Introduction

Objectives

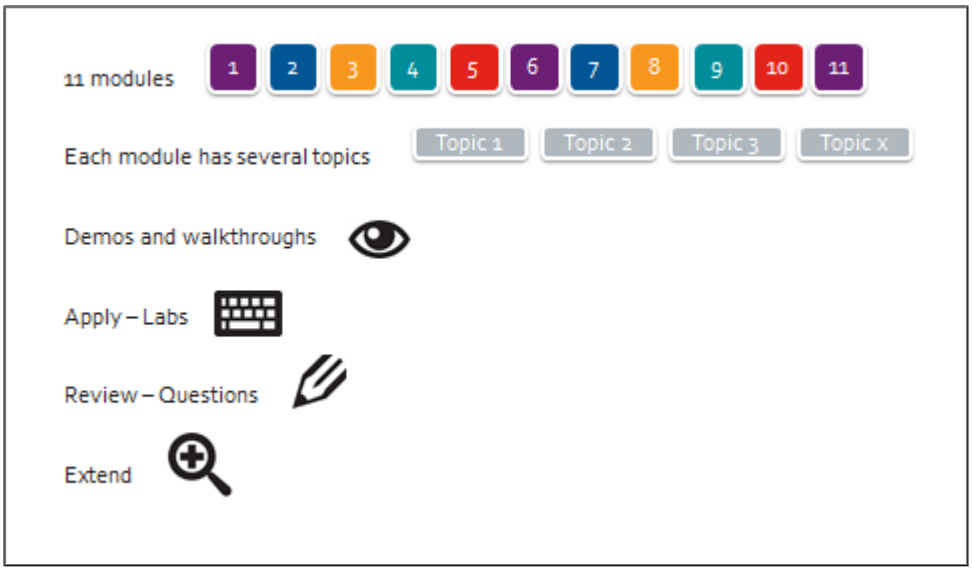
By the end of this topic you will be able to:

- Describe the objectives of this course.

Content

Objectives for This Course

Course Format



Module Overviews

Module 1 Sitecore Overview

Course overview

What is Sitecore?

Sitecore from an author's perspective

Sitecore from a developer's perspective

Module 2 Defining Data

Creating data templates

Data template inheritance

Standard values

Insert options

Summary and optional lab

Module 3 Presentation

Presentation is dynamic and modular

Setting up a Visual Studio Project

Creating a layout

Creating components

Dynamic binding

Rendering content

Module 4 Sitecore API

Basic API concepts & retrieving items

Item links

Creating, deleting, and modifying items

Working with complex fields

Module 5 Advanced Presentation Concepts

Reusable content

Layout deltas

Module 6 Real World Solutions

Familiar concepts

Dealing with larger sites

Sitecore query

Module 7 Configuring the Page Editor

Datasource restrictions

Parameters and parameter templates

Placeholder settings

Advanced Page Editor configuration

Module 8 Dealing With Your Data

Item buckets

Search

Module 9 Recommended Practices

Working with media

Caching & performance

Publishing

Installing & scaling

Team development

Dealing with deployment

Basic security

Workflow

Module 10 Marketing Functionality

Introduction

Engagement values & goals

Profiling & personalization

Module 11 Optional Topics

Branch templates

Other item and template properties

Pipelines and events

Rules

Placeholder overrides

Exam and Certification

Theory exam with 40 questions

Topic 1.2 What Is Sitecore?

Introduction

Objectives

By the end of this topic you will be able to:

- List key features of Sitecore Customer Experience Platform (XP).
- Describe a typical scenario for a visitor.
- Find the information that you need.
- Contact support.

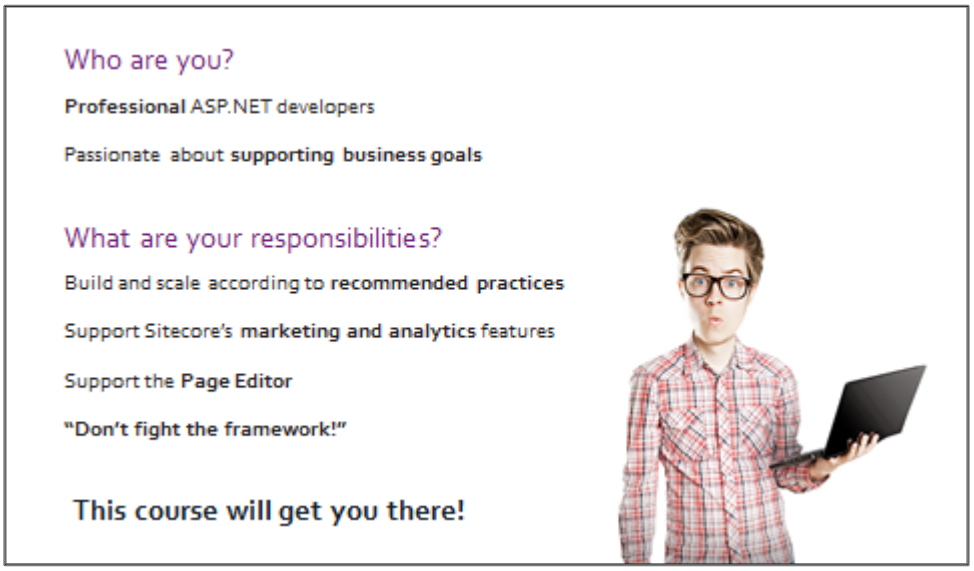
Content

Sitecore Is a Customer Experience Platform

Cross-Channel Communication

A Typical Scenario

The Developer's Role



What's in it for you?

A world leader in customer experience management

The company delivers highly relevant content and **personalized digital experiences.**

You **will** be sought after for your Sitecore skills.

MVP program
Technical previews, MVP summit, closed product feedback circuit

Active community
SDN, Twitter, Stack Overflow, blogging

Clients include:

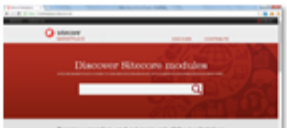
ASOS, Nestle, Brother, Cadbury, Dyson
easyJet, Canon, Heineken, Kia Motors
Play.com, Virgin Active, Thomas Cook

Where Can I Find More Information?

Knowledgebase *(support articles)*

kb.sitecore.net

Marketplace *(modules)*



marketplace.sitecore.net

Developer Network *(docs, forum, downloads)*

sdn.sitecore.net

Useful Guides

- Scaling Guide
- CMS Diagnostics Guide
- CMS Performance Tuning Guide
- Page Editor Recommended Practices for Developers

Documentation

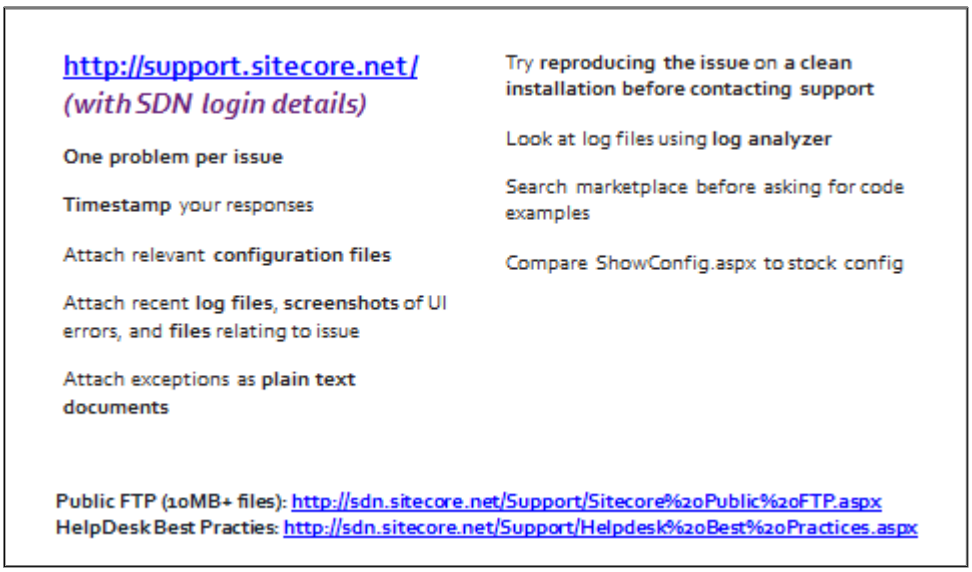
- https://doc.sitecore.net/
- https://sitecore-community.github.io/docs/

Modules

- Email Experience Manager
- Web Forms for Marketers
- Social Connected

All modules: http://sdn.sitecore.net/Resources/Downloads%20Overview.aspx

How Do I Contact Support?



Task – Sign Up for the SDN

- Browse to http://sdn.sitecore.net/.
- **Register,** making sure that your **name** is written as you want it to appear on the course certificate.

Topic 1.3 Sitecore from an Author's Perspective

Introduction

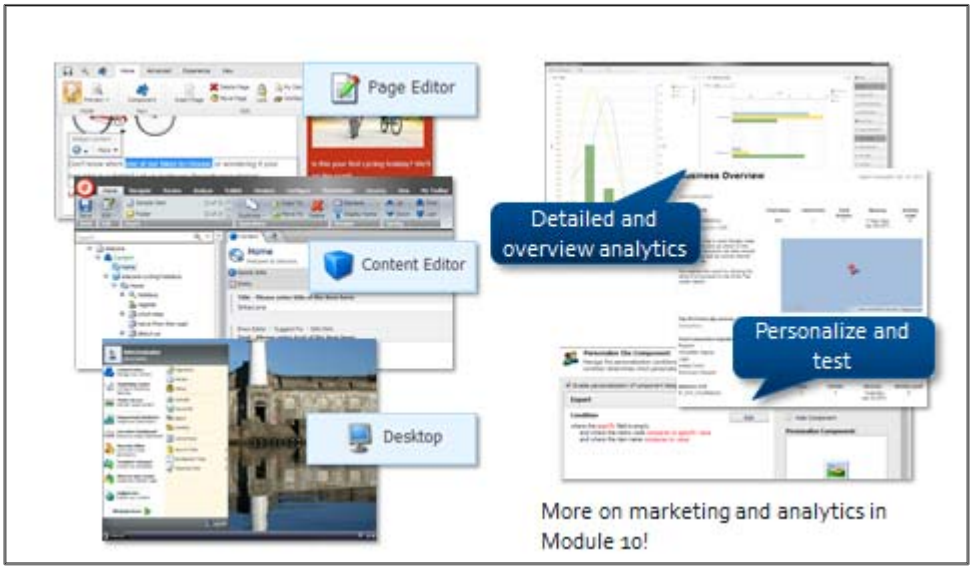
Objectives

By the end of this topic you will be able to:

- Name three interfaces that Sitecore business users have access to.
- Determine which interface is suitable for which task.
- Explain how Sitecore data is represented to authors.
- Name the process by which content is pushed to a live environment.

Content

Sitecore as a Content Management System (CMS) and Marketing Platform



Demo – Sitecore Interfaces

In the following demo, we will:

- Log into the various Sitecore interfaces.

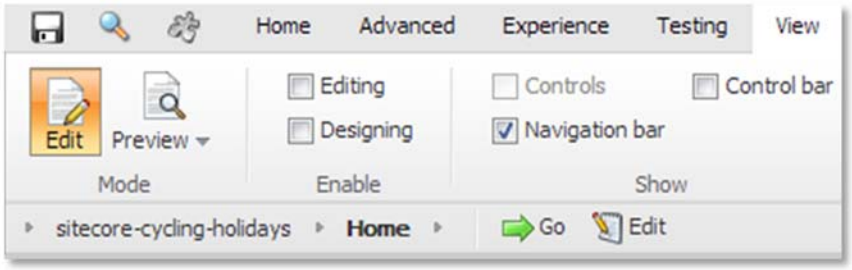
1. Open the training site in a browser and append ***/sitecore*** to the URL (for example, *http://BasicSitecore/sitecore*).
2. Log in as the administrator (password: **b**).

Note
There are three interfaces used for editing: *Page Editor*, *Content Editor* and *Desktop*.

3. Double-click the **Page Editor** icon to log into that interface.
4. **Expand** the **ribbon** using the arrow in the top right-hand corner.

5. Explore some of the tabs:
   Home – for common add/delete/move tasks
   Experience – for changing languages
6. Click the **View** tab. Notice that both the **Enable** and **Show** checkboxes are not selected.
7. Select the **Navigation bar** checkbox.

Note
The navigation bar is appended to the ribbon. This allows authors to navigate through the site structure by selecting the appropriate item and clicking Go.



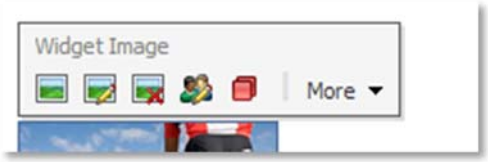
8. Navigate to the **/home/which-bike** item.

Demo – Editing in Page Editor

In the following demo, we will:

- Use the Page Editor to edit content inline.

1. On the **View** tab, select the **Editing** checkbox and hover over the various areas of the page such as an image, some text and so on. The **Edit Frame** toolbar appears. The editing options available on the toolbar depend upon the type of content (image or text) that you have selected.



2. Click the **Which bike?** title and change it to ***Which bike should I choose?***
3. Click the top-left corner to **save** the edited title.
4. On the **Home** tab, click **Insert** and select a new Standard Content template.
5. Give the template the name ***Bikes for beginners***. You have now created a new page.

Note
The breadcrumbs automatically know what to display.

6. Navigate back to the **Which bike?** Item. The page that you just inserted has been automatically added to the listing.

Demo – Designing in Page Editor

In the following demo, we will:

- Use the Page Editor to design a page.

1. Click the **View** tab.
2. Clear the **Editing** checkbox and select the **Designing** checkbox instead.
3. Click various components on the **Which Bike** page.

Note
You can no longer edit content, but you can now select areas of functionality on the page. You still get an edit frame, but the options are commands like *Move* or *Delete*.

4. On the **Home** tab, click **Add Component** (or the corresponding icon next to the tab). A number of **Add to here** buttons appear where you can insert a component.
5. Click the button at the top of the left-hand column and select the **Subnavigation** component and then click **Select.**
6. If the list on this page is long, you might want to present a quick overview by adding a side navigation component. To do that, click **Add to here** at the top of the left-hand column and choose a **Side Navigation** component.
7. Verify that a component that lists all the page titles, including the one that you just created, appears on the page.

Demo – Desktop and Content Editor

In the following demo, we will:

- Use Content Editor to edit and publish content.

1. Click the **Log off** button and go back to the **/sitecore** interfaces menu.
2. Click **Options.**
3. Log in as the administrator and double-click the **Desktop** icon.
4. Click the **Sitecore** button and open the **Content Editor.**

Note
There are a variety of interfaces that can be accessed via the Desktop icon.

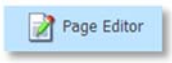
5. Type *Which bike* in the search bar above the Sitecore tree. The item that was edited earlier is displayed in the result set.
6. Open the item. It is made up of several different types of fields.

Note
The Heading field now reads *Which bike should I choose?* because it was changed using the Page Editor interface.

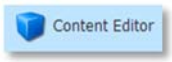
7. Change the **Heading** field to *Choices of bike* and then, on the ribbon, click **Publish > Preview** to see the changes.

8. If you are happy with the changes, click the **arrow** next to **Publish > Publish site,** and select the **Incremental Publish** option to publish the changed items.
9. **Log off** and view the **Which bike***?* page as a regular visitor. Because we have published the changes, the items have been pushed from the sandbox environment to the web environment.

You can access the **Login** screen by navigating to the **Home** page with **/sitecore** appended to the site's URL. Click the **Options** button to choose one of three editor environments:



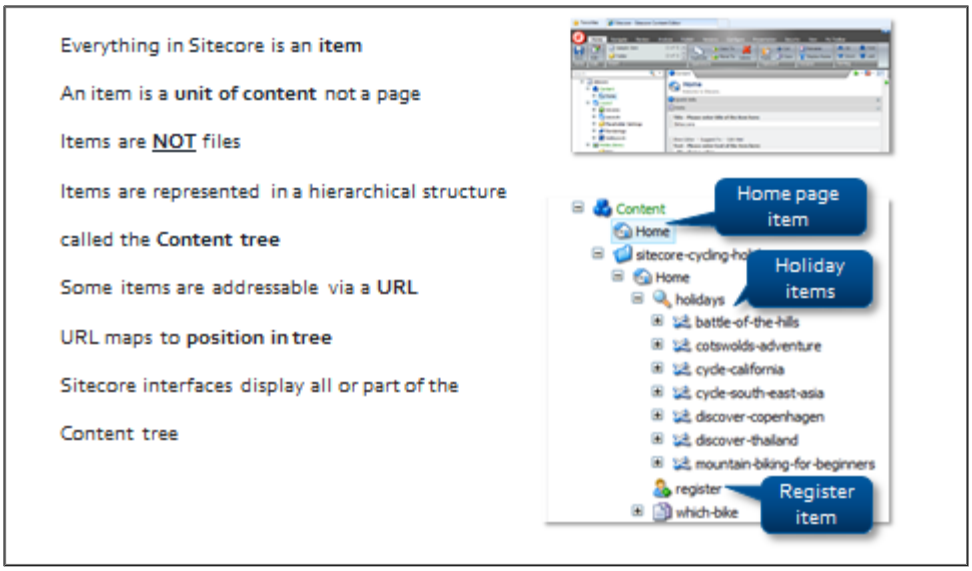
The **Page Editor** displays individual pages with an editing ribbon that allows business users to either edit content inline (changing text or adding media like images) or modify the page layout (adding or removing areas on the page).



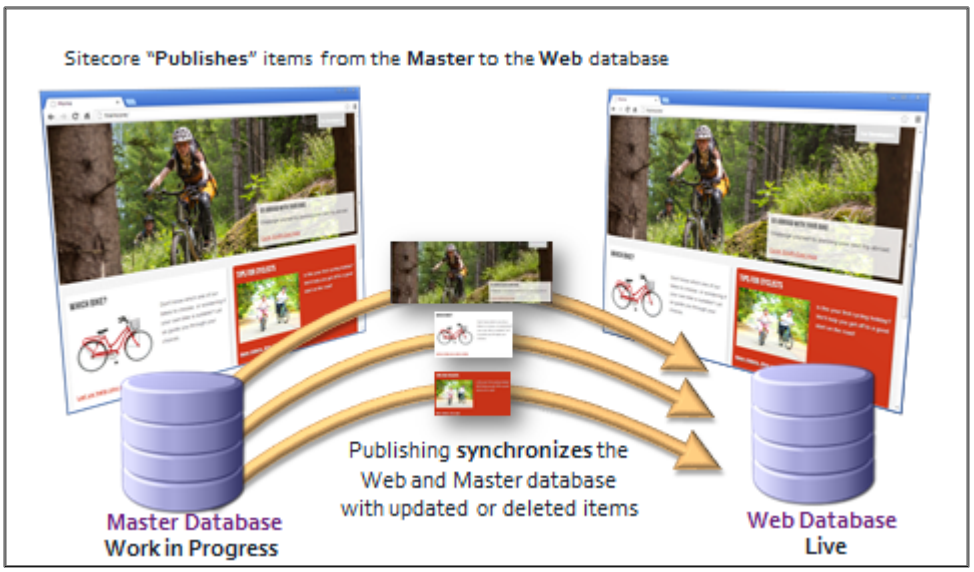
The **Content Editor** allows a business user to work directly with the all the data stored in Sitecore. Unlike in the **Page Editor,** authors in this editor environment are editing the actual data excluding any presentation.

The **Sitecore Desktop** uses the familiar "Windows" metaphor with a **Start** button and menu to access all Sitecore interfaces, including the **Page Editor** and **Content Editor**. The actual tools authors see depend on their security context; administrators see everything.

Items and the Content Tree



Publishing and the Sitecore Databases



Topic 1.4 Sitecore from a Developer's Perspective

Introduction

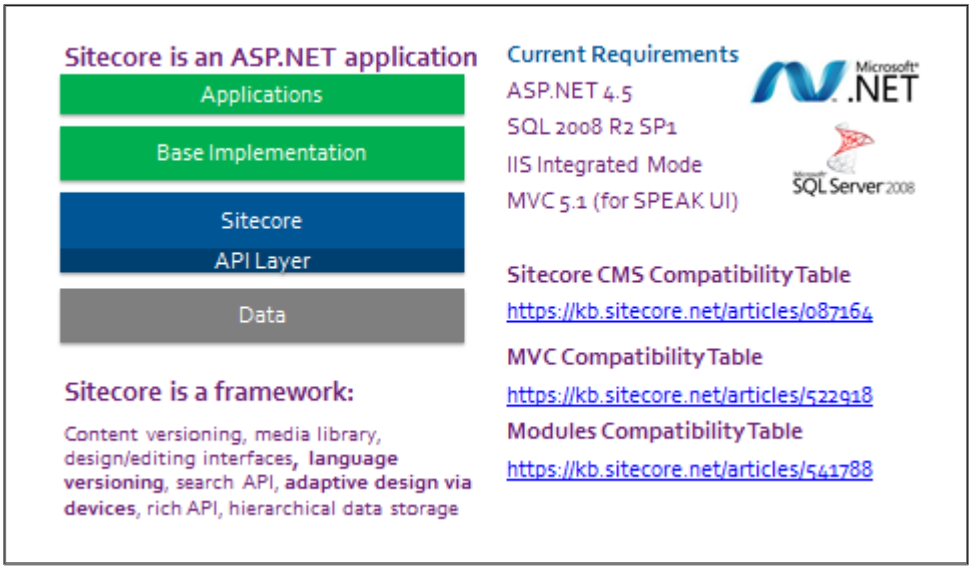
Objectives

By the end of this topic you will be able to:

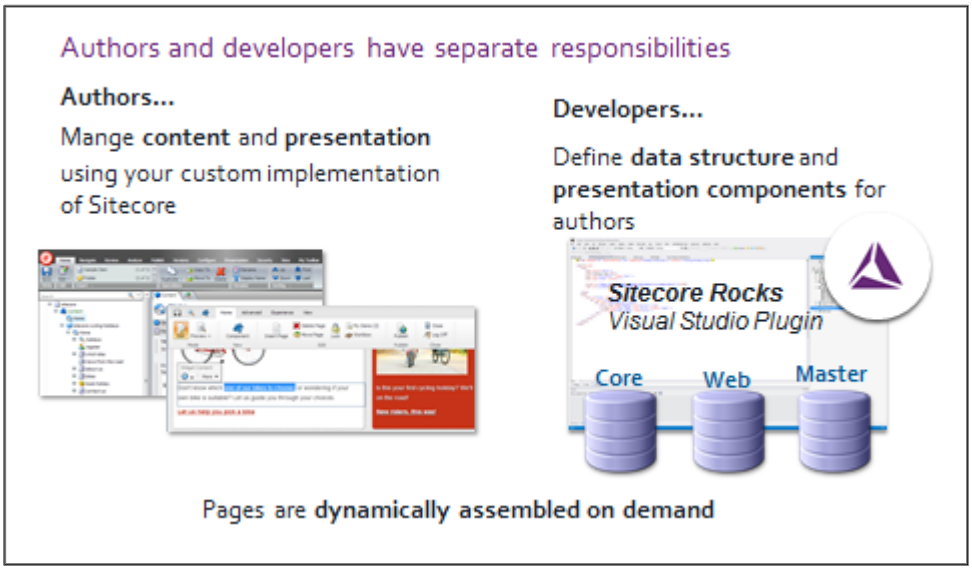
- Explain what gets installed on your computer when you install Sitecore.
- Name three Sitecore databases.
- Name four foundation features that Sitecore provides out-of-the-box.
- Draw out the minimum requirements for a production environment.

Content

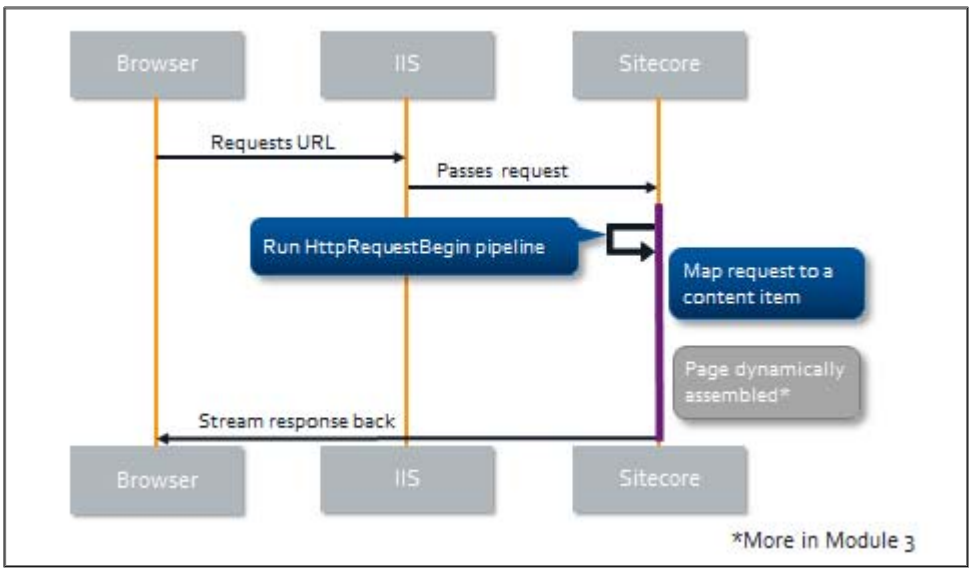
Architecture and Requirements



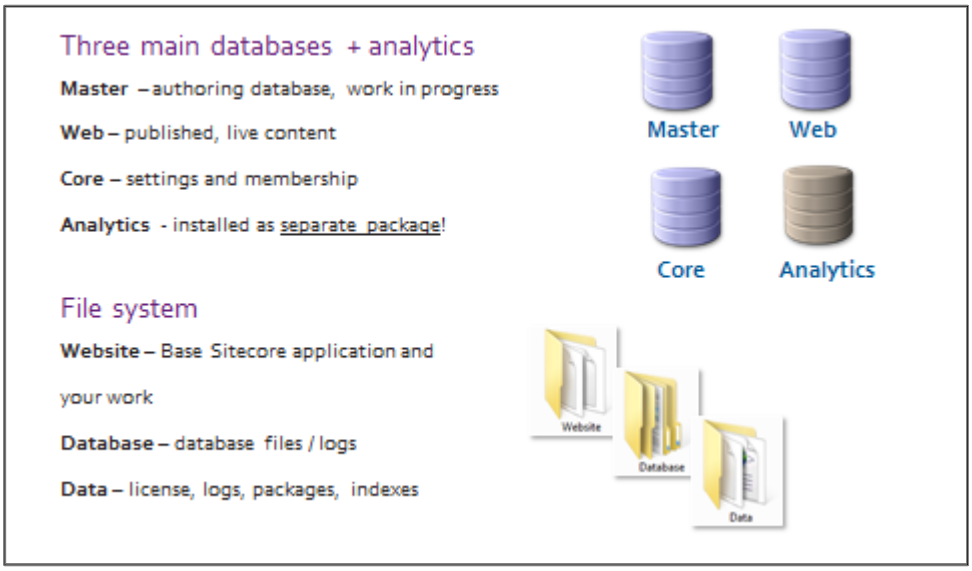
Separation of Concerns



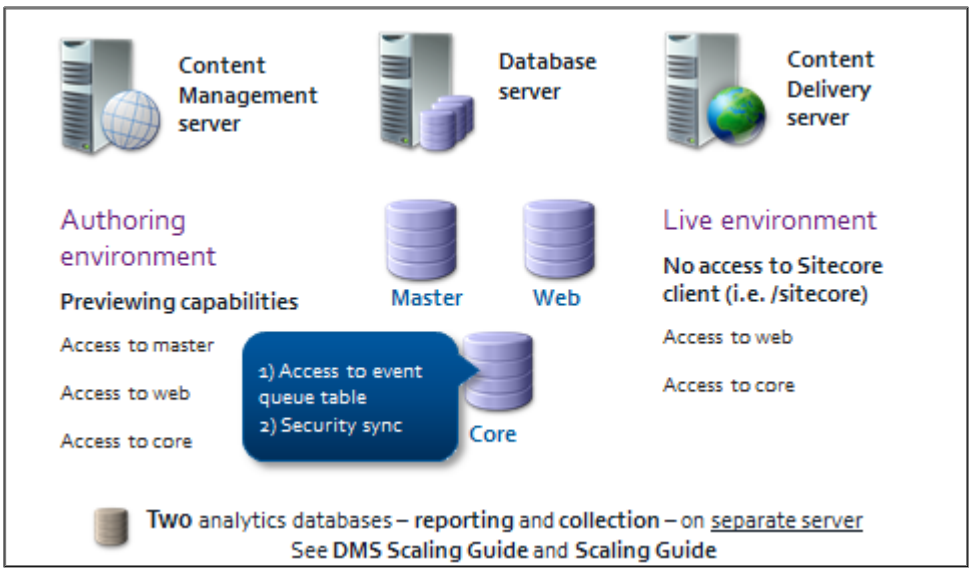
How Sitecore Handles a Request



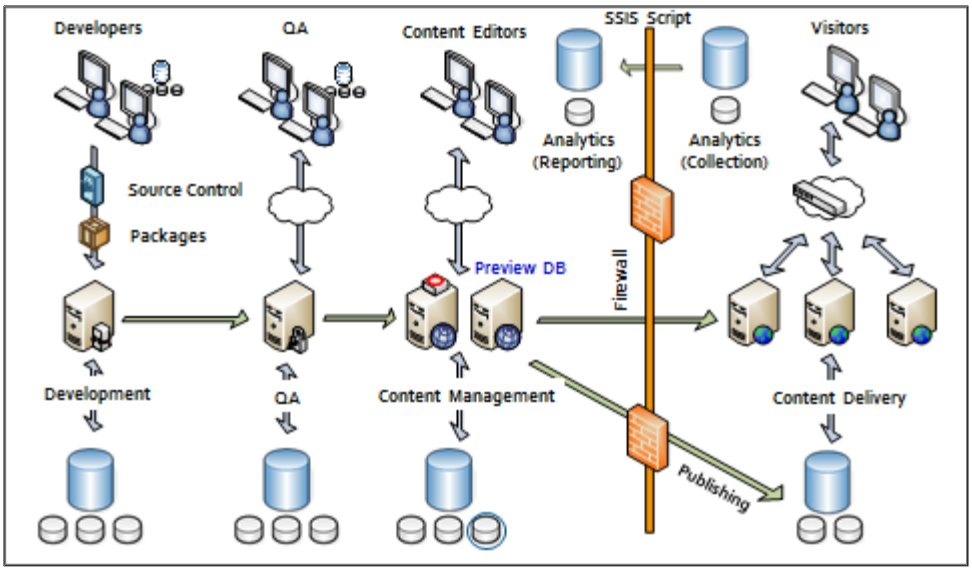
What Gets Installed?



Installation Scenarios - Recommended Minimum



A Realistic Example



Module 2
Defining Data

Contents

- Creating Data Templates
- Data Template Inheritance
- Standard Values
- Insert Options
- Summary and Optional Lab

Topic 2.1 Creating Data Templates

Introduction

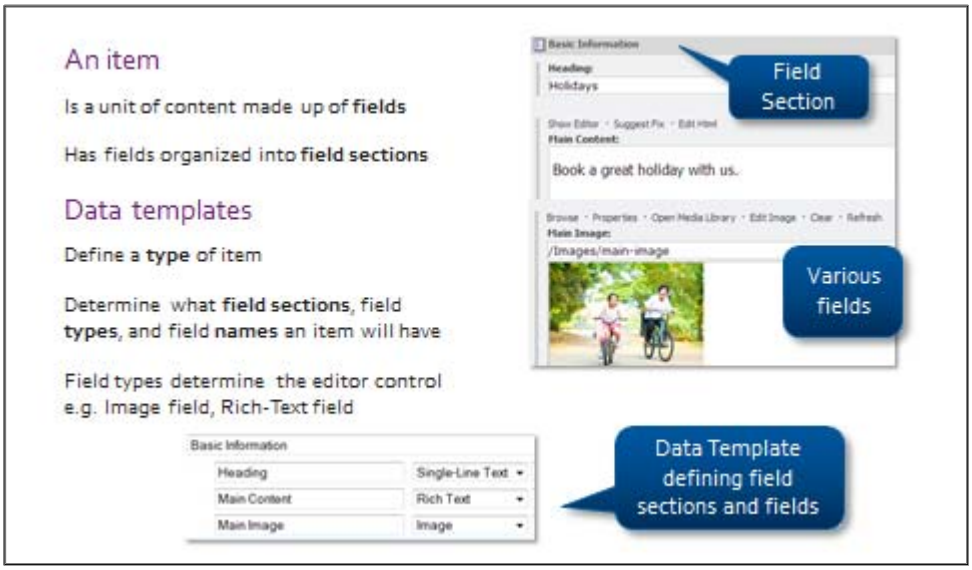
Objectives

By the end of this topic you will be able to:

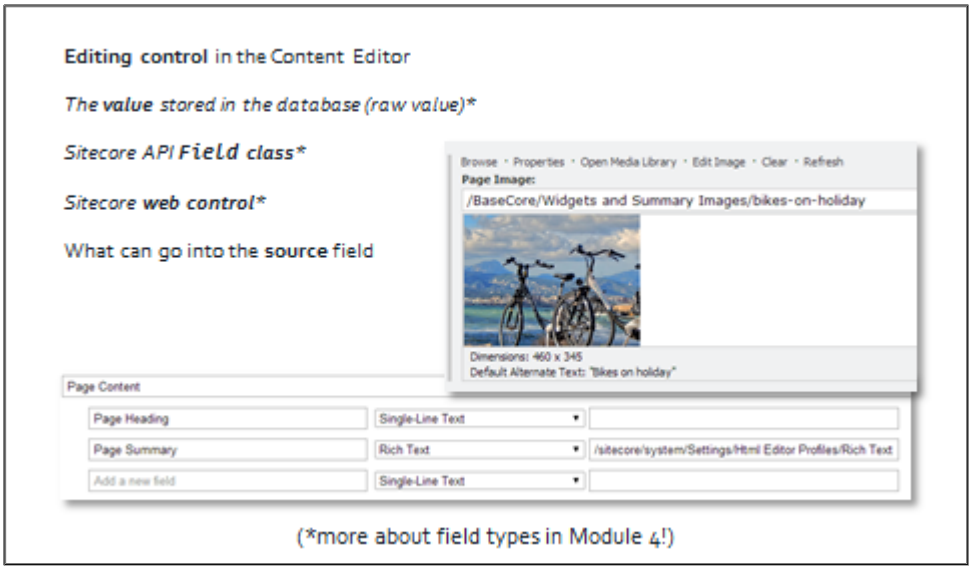
- Create a data template using Sitecore Rocks.

Content

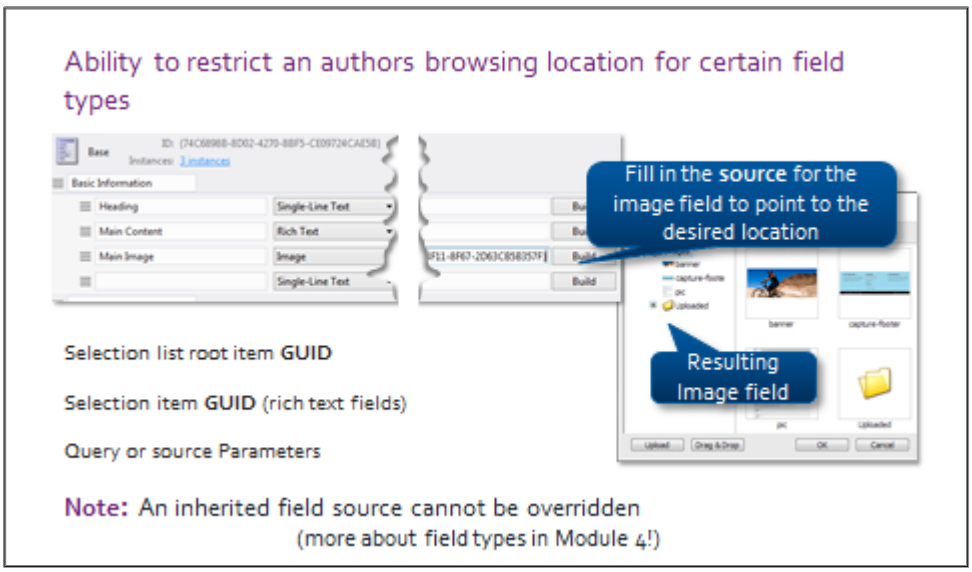
Items, Fields, and Data Templates



Field Type Determines…

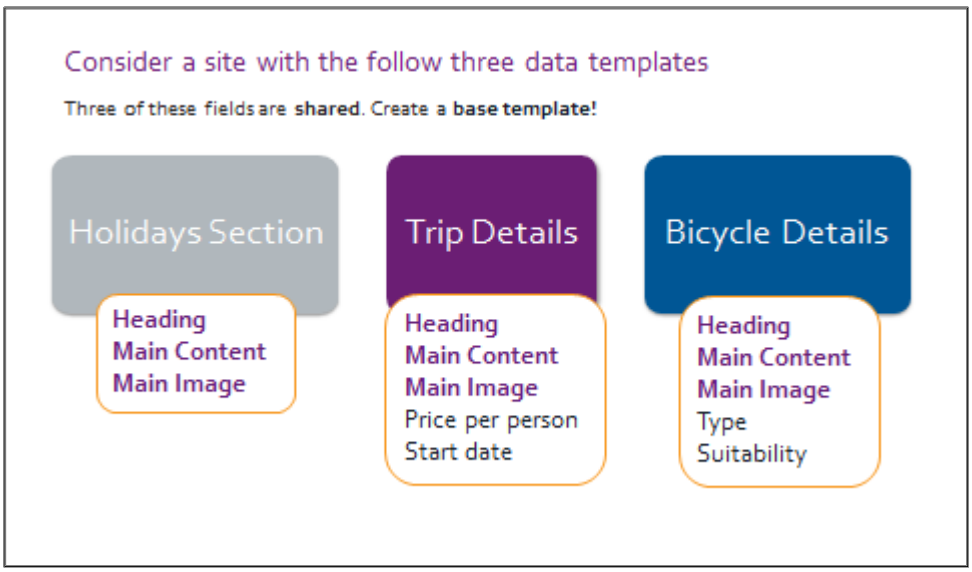


Field Source

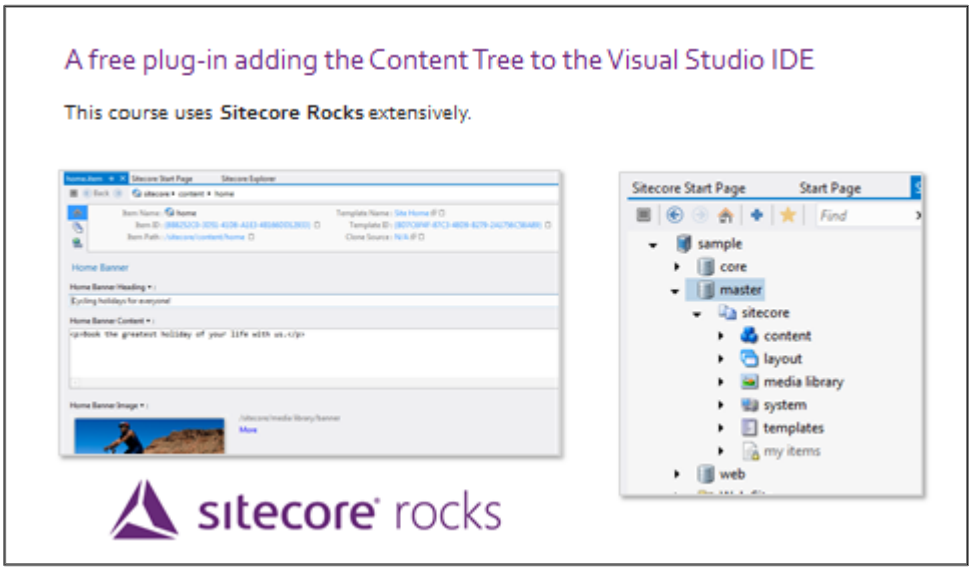


Who Decides What the Data Structure Will Be?

Are Fields Shared?



We Use Sitecore Rocks



Tip
For more information about Sitecore Rocks, see:
http://www.sitecore.net/Learn/Blogs/Technical-Blogs/Trevor-Campbell/Posts/2013/02/28-Days-of-Sitecore-Rocks-Intro.aspx

Demo – Creating a Data Template

In this demo, we will:

- Create a new data template named *Base*.
- Add three fields to the template: *Heading*, *Main Image*, and *Main Content*.

Note: *You will perform these tasks in the upcoming labs.*

Apply – Topic 2.1

Creating a Base Data Template

You can find a blank installation of Sitecore on your student computer. Your instructor will give you the URL. Throughout this manual, the hostname for the blank Sitecore instance will be referred to as http://basicsitecore/. If your hostname is different, then please adjust the URLs accordingly.

Lab A. Create a Data Template Named *Base*

Overview Steps

Optional detailed steps are available below to help you.

Using **Sitecore Rocks,** connect to the *BasicSitecore* instance, create a data template called *Base* under *master/sitecore/Templates/User Defined.*

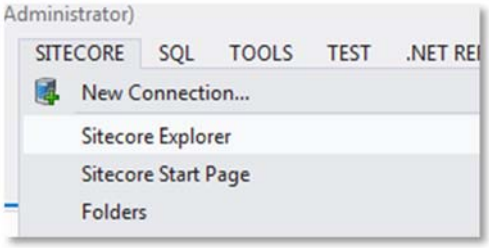
Create **field sections** and **fields** as described in the following table:

Field section	
Basic Information	
Field name	**Field type**
Heading	Single-line text
Main Content	Rich Text
Main Image	Image

Detailed Steps

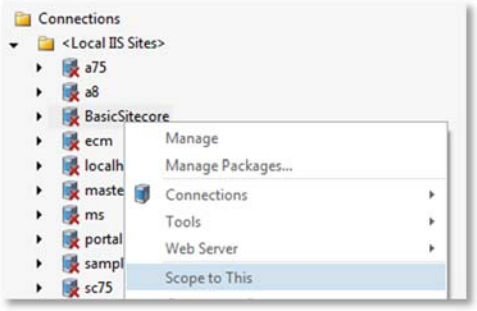
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Open **Visual Studio**.
2. To open **Sitecore Explorer,** select **Sitecore>Sitecore Explorer** in the Visual Studio toolbar.

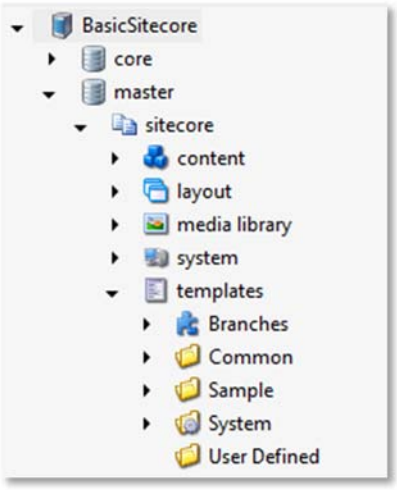


3. Locate your instance in the Sitecore Explorer window under **Connections > <Local IIS Sites>**. The instance name is *BasicSitecore* unless your instructor tells you otherwise.

4. Right-click *BasicSitecore* and select **Scope to This**.



5. Expand in the *BasicSitecore* by clicking the arrow to the left of the server icon.
6. Expand the **master** database.



7. Continue to expand items until you can see the **User Defined** folder under: */BasicSitecore/master/sitecore/templates/User Defined*.
8. Right-click **User Defined** and select **New Template…**

Keyboard Shortcut

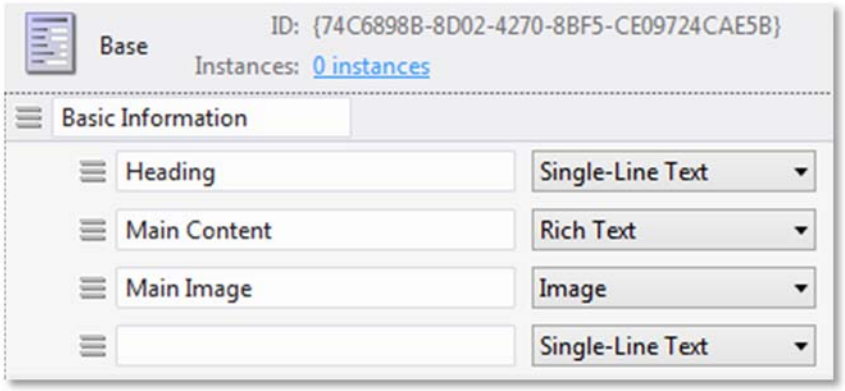
CTRL + Shift + Space = Commandy

Commandy is similar to "Navigate To…" function in Visual Studio; it's a utility for finding functions in Sitecore Rocks. Select an item and use the command to bring up a searchable list of available tasks.

9. Name the data template *Base.*
10. Create **field sections** and **fields** as described in the following table:

Field section	
Basic Information	
Field name	**Field type**
Heading	Single-line text
Main Content	Rich Text
Main Image	Image

Your final screen should look like this:



11. Click **Save**.

Extend

- **Data Definition Reference**
  http://sdn.sitecore.net/Reference/Sitecore%207/Data%20Definition%20Reference.aspx
- **Content Author's Cookbook**
  http://sdn.sitecore.net/Reference/Sitecore%207/Content%20Authors%20Cookbook.aspx

Topic 2.2 Data Template Inheritance

Introduction

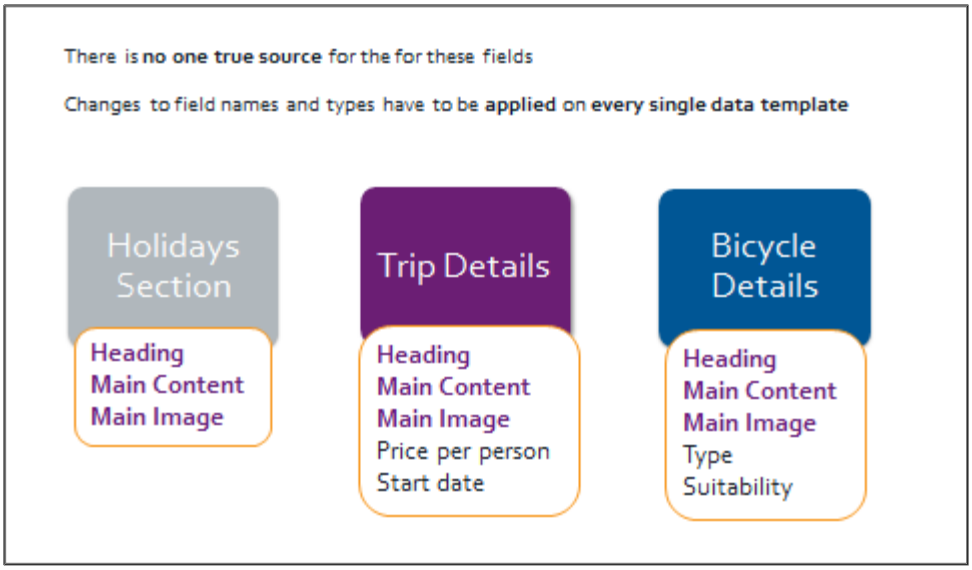
Objectives

By the end of this topic you will be able to:

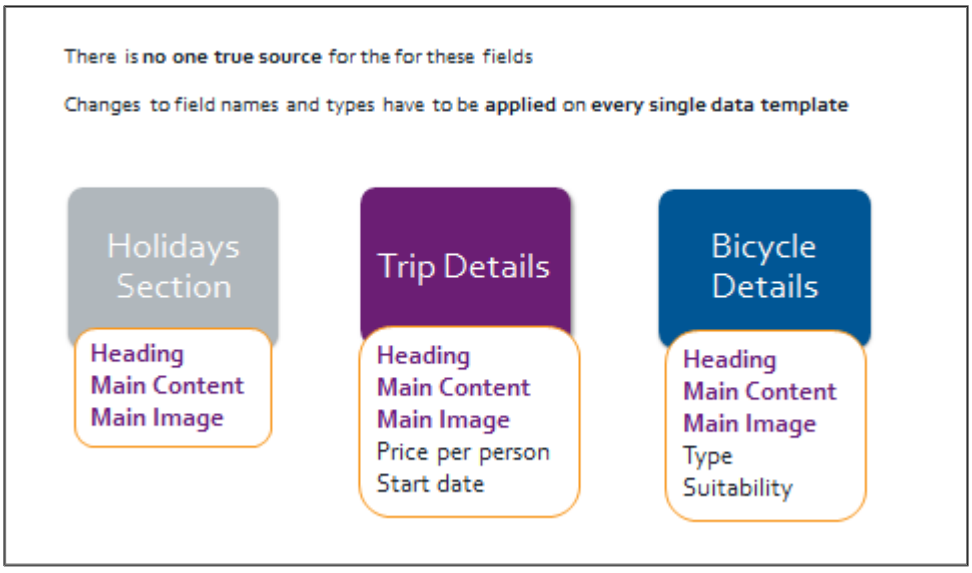
- Specify the base templates a data template inherits from.
- Describe the benefits of data template inheritance.
- Name the data template that all data templates inherit from.

Content

RECAP: Challenges with Field Duplication



Data Template Inheritance



Tip

A data template can inherit from any number of base templates.

You can use a data template multiple times in the inheritance chain (for example, if *Base* and *Holiday* both inherit from *Standard Values*, the fields will not appear twice).

Best Practice

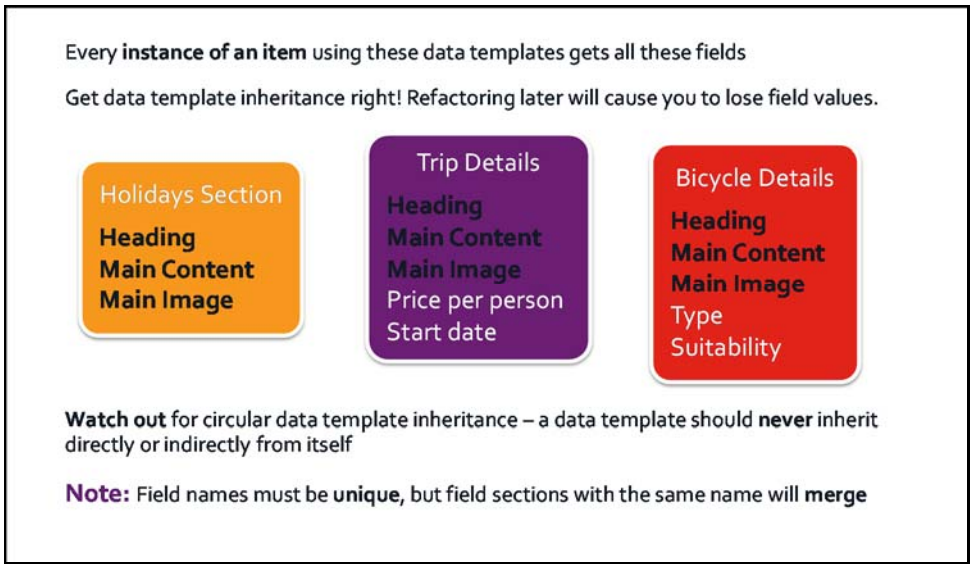
Although you can use a data template multiple times in the inheritance chain, you should avoid creating a circular template inheritance (A → B → A).

Similarly, field names must be unique. You cannot define *Title* field in both *Base* and *Holiday* templates and expect to be able to target them as separate fields in the latter's superset of fields. They do not merge like template sections do.

Never inherit a field that an item is not going to use (for example, if an author can populate a field that is never rendered or used, consider revising your inheritance chain).

Overusing data template inheritance can quickly become difficult to manage beyond 3 or 4 levels.

Resulting Data Templates



Important

Data template inheritance should be considered at the start of your project. Moving fields into a base template after items have been created using that data template will result in a loss of data.

Demo – Setting Up and Viewing Data Template Inheritance

In this demo we will be:

- Creating a **Holidays Section** data template.
- Choosing an **icon** for the **Holidays Section** data template.
- Configuring the **Holidays Section** data template to inherit from *Base*.
- Creating a **Family Holidays** item based on the **Holidays Section** data template.
- Create a **Trip Details** data template and configure it to inherit from **Base**
- Creating a **Cycle the Cotswolds** item based on the **Trip Detail** data template under the **Family Holidays** item.
- Viewing the list of data templates that the **Cycle the Cotswolds** item inherits.

Note: *You will perform this task in the upcoming lab.*

Best Practice

Encapsulate all fields required for a specific feature in a data template, which other templates can then inherit from. For example:

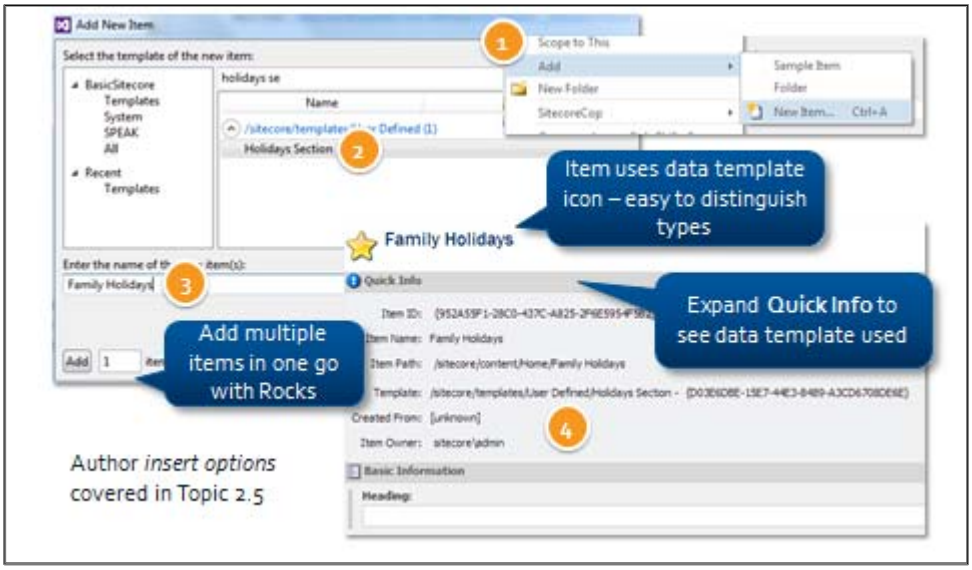
- *Page Meta Data* [Title, Description, Keywords]

  or
- Banner [Banner Image, Banner Text]

Tip

By enabling **standard** fields in the **View** group in the Content Editor, you can see all the additional fields derived from the **Standard Template**. Right-click in the Editor Pane in Sitecore Rocks to see the standard fields as well.

Creating Items



Apply – Topic 2.2

Data Template Inheritance

In the following labs, you will use Sitecore Rocks to:

- Create **Holidays Section** and **Trip Details** data templates.
- Configure both data templates to inherit from **Base**.
- Create a **site tree** using the **Holidays** Section and **Trip Detail** data templates.

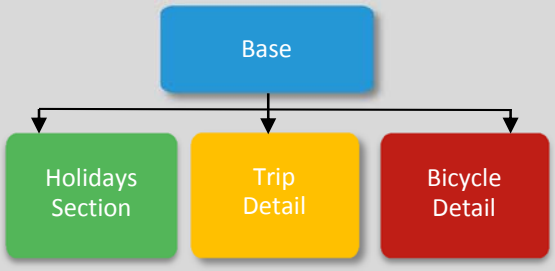


Figure 2-1 Data template inheritance structure

Lab A. Create One Data Template Named *Holidays Section* and Another Named *Trip Detail*

Overview Steps

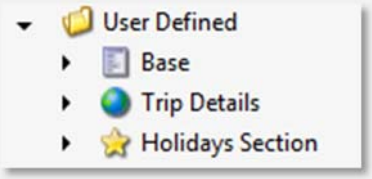
Optional detailed steps are available below to help you.

Using **Sitecore Rocks,** create a data template named **Holidays Section** under :*/sitecore/Templates/User Defined* and assign a **star** icon to it. This data template does not require any additional fields.

Create a second data template named **Trip Details** under */sitecore/Templates/User Defined* and assign a **globe** icon to it. Create field sections and field as described in the following table:

Field section	
Trip Information	
Field name	**Field type**
Price per person	Single-line text
Start date	Date

*At the end of this lab, you should have three data templates under the **User Defined** folder:*

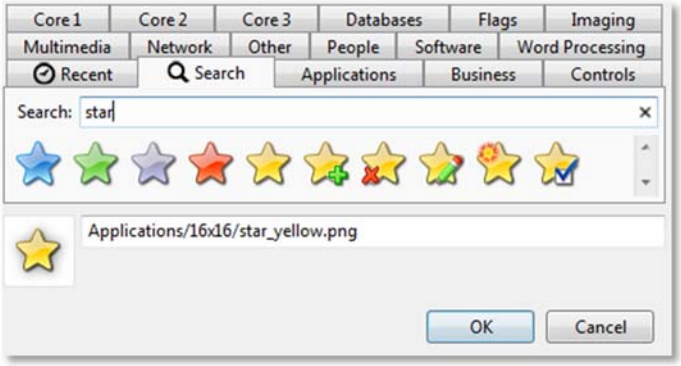


Detailed Steps

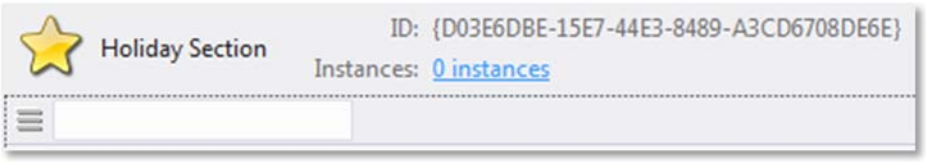
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In **Visual Studio,** open the **Sitecore Explorer** toolbar.
2. Expand the **BasicSitecore** solution until you can see: */BasicSitecore/master/sitecore/Templates/User Defined*.
3. Right-click the **User Defined** folder and select **New Template…** to create a new template.
4. Name the template *Holidays Section*.
5. Double-click the **Holidays Section** template to open it. Click the icon in the top left-hand corner to open the **icon selection dialog**.
6. Click the **Search tab** and search for "*star*".

7. Select the **yellow star** icon and click **OK**.



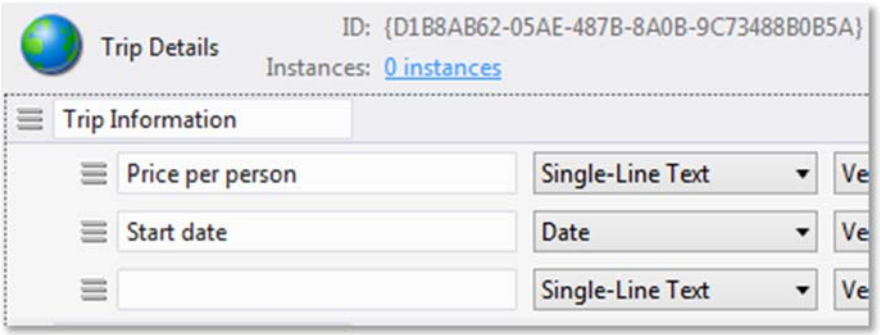
8. **Save**. Your data template will look like this:



9. Right-click the **User Defined** folder and select **New Template…** to create a new template.
10. Name the template *Trip Details.*
11. Change the **Trip Details** data template's icon to an **earth icon**.
12. Create **field sections** and **fields** as described by the following table:

Field section	
Trip Information	
Field name	**Field type**
Price per person	Single-line text
Start date	Date

13. **Save**. Your data template will look like this:



Lab B. Configure *Holidays Section* and *Trip Details* to Inherit from *Base*

Overview Steps

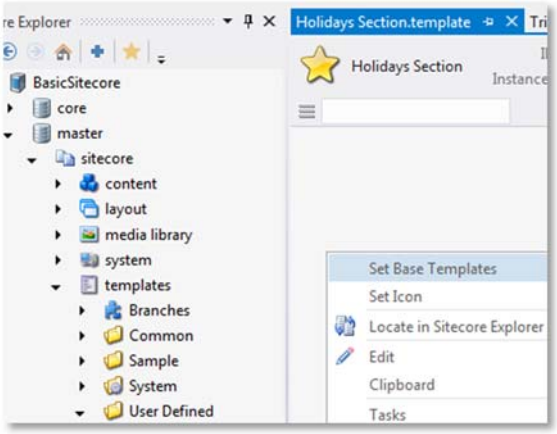
Optional detailed steps are available below to help you.

Using **Sitecore Rocks,** open the **Holidays Section** data template and right-click inside the grey space. Select **Set Base Templates** and select **Base**. Repeat the process for **Trip Details**.

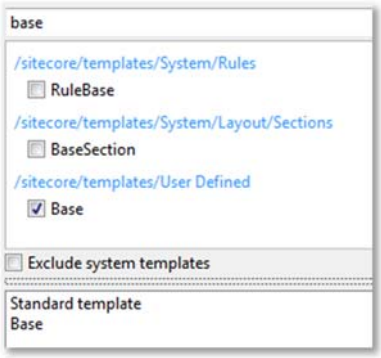
Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Ensure that you are in the Sitecore Explorer toolbar in Visual Studio. Locate the **Holidays Section** data template under: */BasicSitecore/master/Templates/User Defined/Holidays Section*.
2. Double-click the **Holidays Section** item to open it.
3. Right-click inside the grey space and select **Set Base Templates**:



4. Type *base* in the **Search** field and select the **Base** checkbox:



5. Click **OK** and **Save**.
6. Repeat the same process for the **Trip Details** data template. This data template should also inherit from **Base**.

Lab C. Create Content Items Using the Holiday Listing and Trip Details Data Templates

Overview Steps

Optional detailed steps are available below to help you.

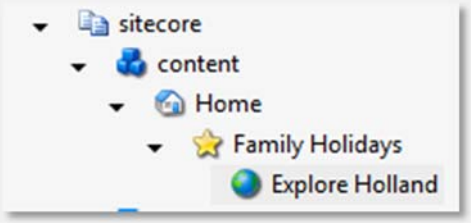
Using **Sitecore Rocks**, right-click on the **Home** item under: */BasicSitecore/master/sitecore/Content* and create a new item based on the **Holidays Section** item. Name it *Family Holidays*. Populate it with the following sample content, or write your own:

Family Holidays	
Basic Information	
Heading	Holidays for the Entire Family
Main Content	Take the kids on a cycling adventure this summer!
Main Image	*Any from the media library*

As a **child** of the Family Holidays item, create a new item based on the **Trip Details** data template. Name this item ***Explore Holland***. Use the following sample content or write your own:

Explore Holland	
Basic Information	
Heading	Explore Holland
Main Content	Explore Holland
Main Image	*Any from the media library*
Trip Information	
Price per person	$600 (or use any value)
Start date	4/24/2013 (or enter any date)

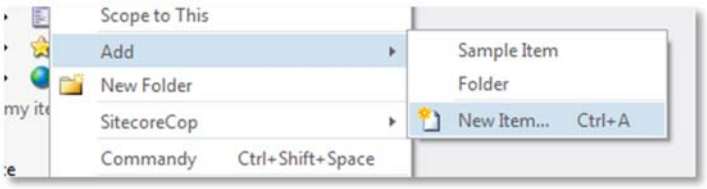
You will end up with a content tree that looks like this:



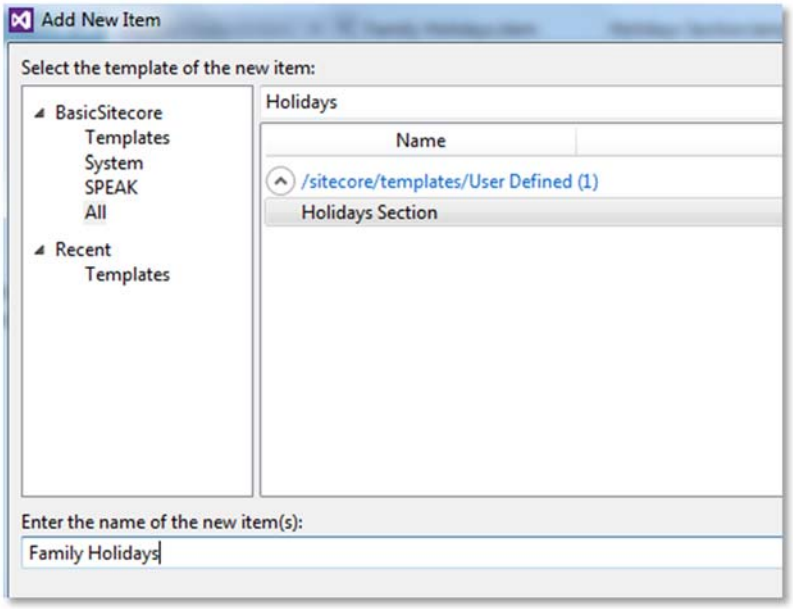
Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio, open the **Sitecore Explorer**.
2. Locate the **Home** item under /BasicSitecore/master/sitecore/Content/Home.
3. Right-click on the **Home** item and select **Add>New Item…**



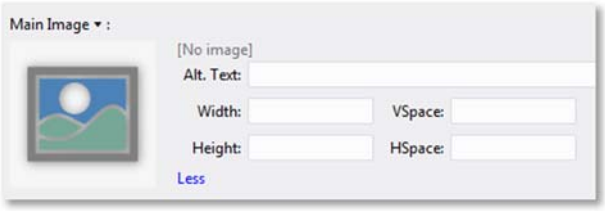
4. Search for **Holidays Section** in the **Add New Item** dialog. Make sure that **All** is selected in the list to the left:



5. Name the item **Family Holidays** and click **OK**.
6. Populate the item with the following sample content, or write your own:

Family Holidays	
Basic Information	
Heading	Holidays for the Entire Family
Main Content	Take the kids on a cycling adventure this summer!
Main Image	*Any from the media library*

7. To insert an image, **drag and drop** an image file from the file system *or* the Sitecore Media Library onto the **image placeholder** icon under the **Main Image** field. There are sample images available in the **Student Resources Folder** on the file system under: *Student Resources Folder* > *WND Labs* > *Images*.

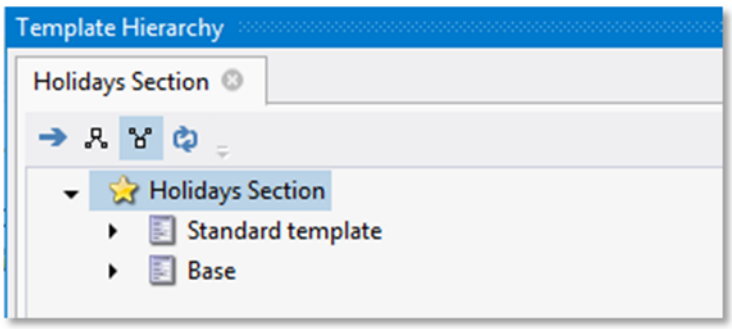


8. Right-click the **Family Holidays** item and insert a new **child item** based on the **Trip Details** data template. Name this item *Explore Holland*.
9. Populate the item with the following content, or write your own:

Explore Holland	
Basic Information	
Heading	Explore Holland
Main Content	Explore Holland
Main Image	*Any from the media library*
Holiday Information	
Price per person	$600 (or use any value)
Start date	4/24/2013 (or enter any date)

Tip

To view a list of data templates that an item or data template inherits from, right-click the item and select ***Navigate>Template Hierarchy***. Two icons allow you to see **inherited** and **inheriting** data templates. The following image shows which data templates the **Family Holidays** item is inherited from:



Topic 2.3 Standard Values

Introduction

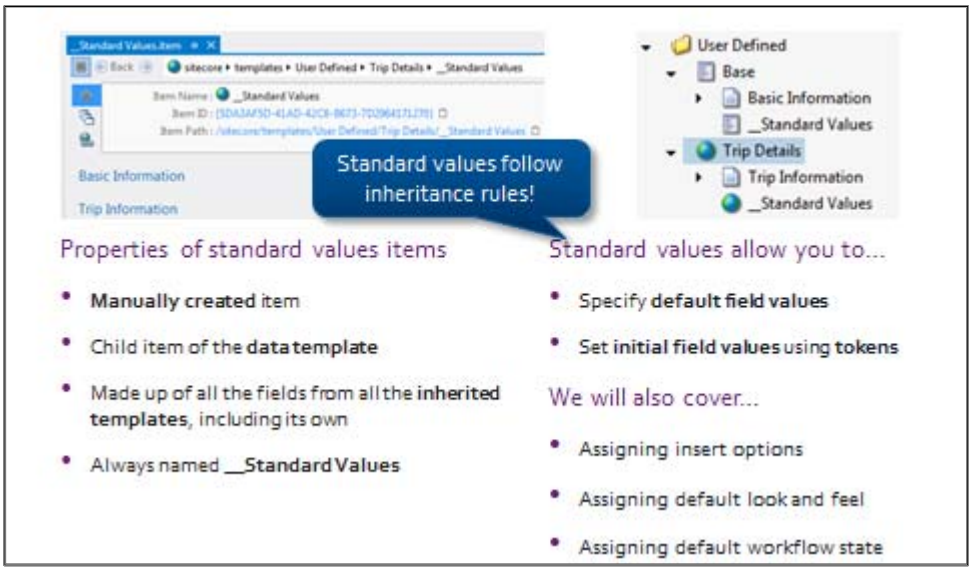
Objectives

By the end of this topic you will be able to:

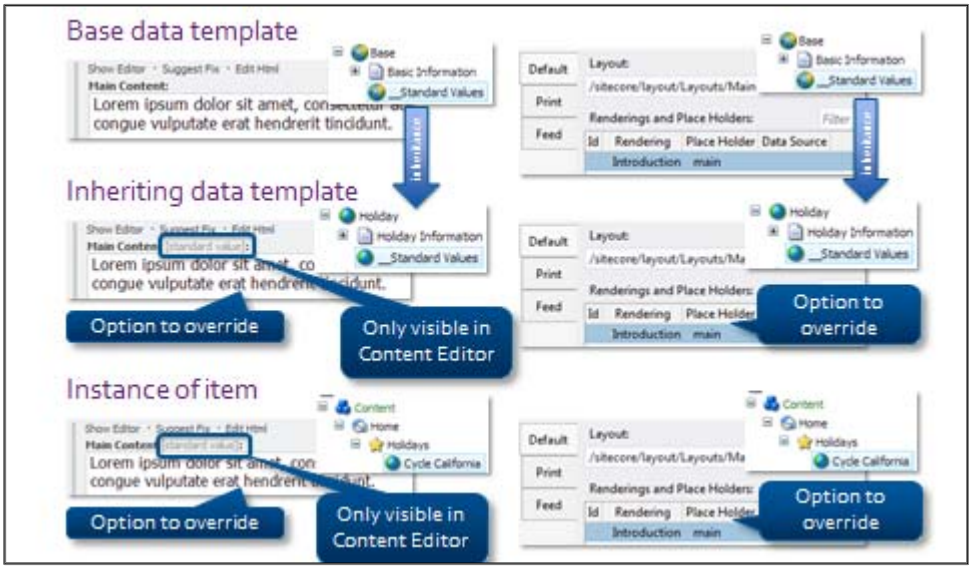
- Create an item's standard values using Sitecore Rocks.
- Use standard values to set the default content of a data template.
- User tokens to insert dynamic field values.
- Reset an item's field values back to the standard values of its data template.
- List the properties of an item that are set on its standard values as recommended practice.

Content

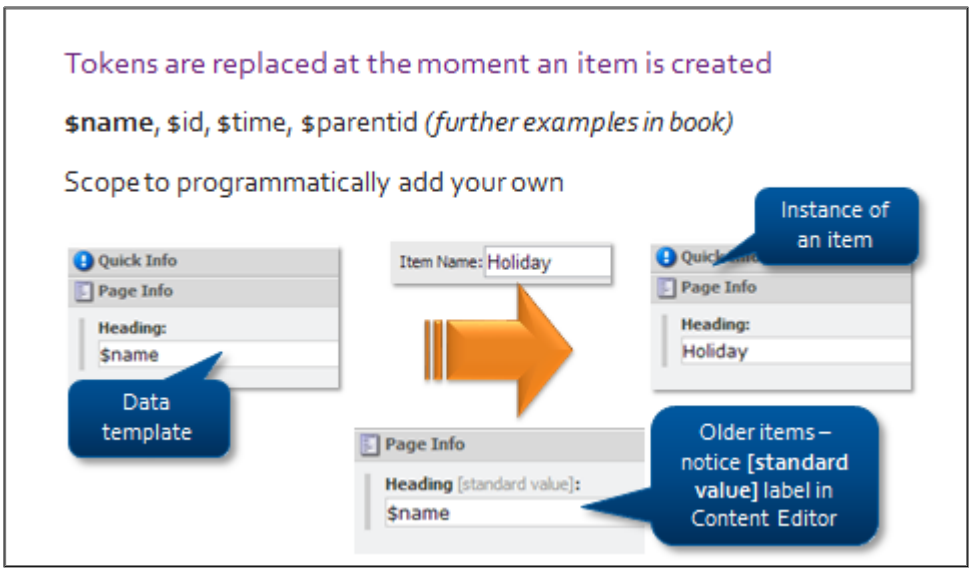
Standard Values Allow You to Set Up Defaults



Standard Values and Inheritance



Tokens

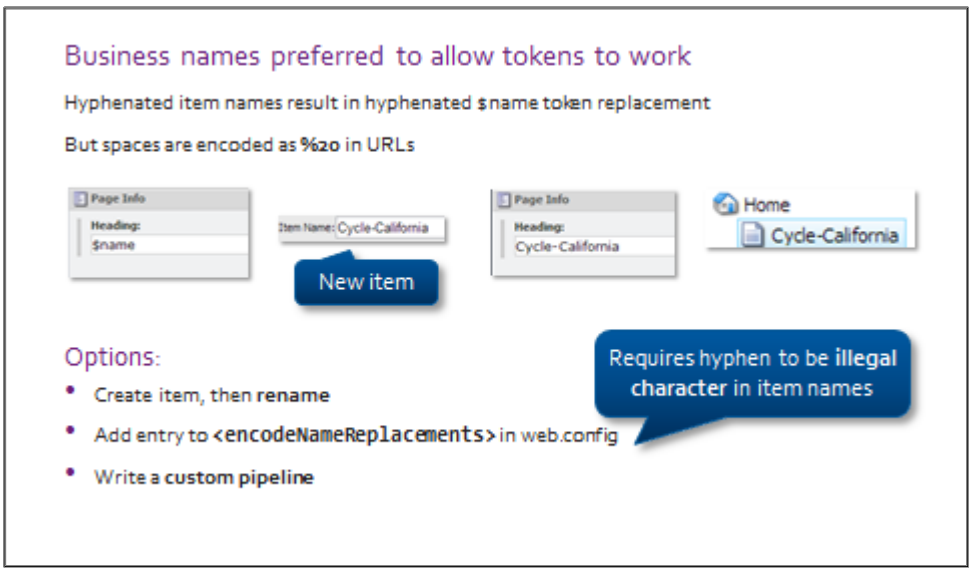


Token Name	Description
$name	Name of the item
$time	ID of the item
$now	ID of the parent of the item
$date	Name of the parent of the item
$time	Current date (ISO format)
$now	Current time (ISO format)
$date	Current date and time (ISO format)

Important

These tokens are only expanded when you are creating an item. This means that if you add a *$name* token to a field in template's standard values, items that have already been created will literally have *$name* in that field.

Item Naming Conventions



Note – URL Duplication

Sitecore will warn you about duplicate item names on the same level, but it will not stop you from creating them. The solutions are to either create a pipeline to rename the item at item creation time, or create a validation rule to intercept and deny saving the item.

Demo – Standard Values, Tokens, and Default Field Values

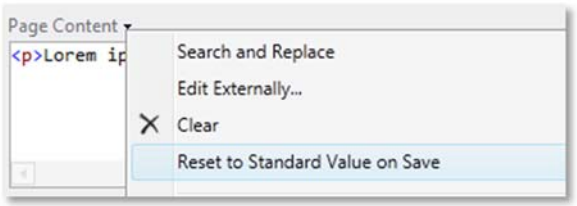
In this demo, we will:

- *Create standard values for the **Base** data template.*
- *Add a **$name** token to the **Heading** field.*
- *Add **Lorem ipsum** content to the **Main Content** field.*
- *Create a new item based on the **Trip Details** data template.*
- *Change **Main Content** and reset to standard values.*

You will be performing these tasks in the upcoming lab.

Tip

To reset an item's field to the value in standard values, right-click the field name and select **Reset to Standard Value on Save**, then save changes.



Apply – Topic 2.3

Create and Use Standard Values

Lab A. Create Standard Values and Populate with Tokens and Sample Content

Overview Steps

Optional detailed steps are available below to help you.

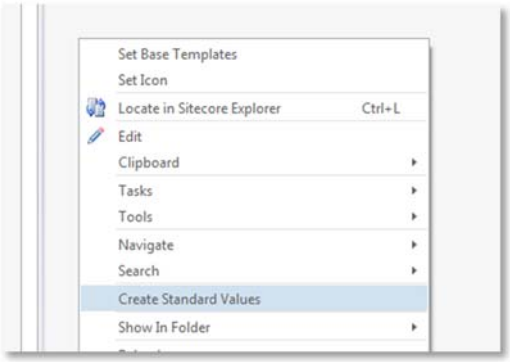
Using **Sitecore Rocks,** create the **Standard Values** for the **Base** data template.

Add a *$name* token to the **Heading** field, and sample content, *Lorem Ipsum* to the **Main Content** field.

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio, open the **Sitecore Explorer**.
2. Locate the **Base** data template under:*/BasicSitecore/master/sitecore/Templates/User Defined Base.*
3. Double-click the data template to open it.
4. Right-click inside the grey space in the right-hand panel and select **Create Standard Values**. This will create a __Standard Values item as a child of **Base**. Right-click **Base** and click **Refresh** to see it.



5. Ensure that the newly created **__Standard Values** item is selected.
6. Type ***$name*** into the **Heading** field.
7. Right-click the **Main Content** field and select **Add 'Lorem Ipsum…',** or type your own sample content.
8. Click **Save**.

Lab B. Create a New Trip Details Item

Overview Steps

Optional detailed steps are available below to help you.

Using **Sitecore Rocks**, create a new **Trip Details** item under: */Home/Family Holidays*. Name it ***Discover the Cotswolds***.

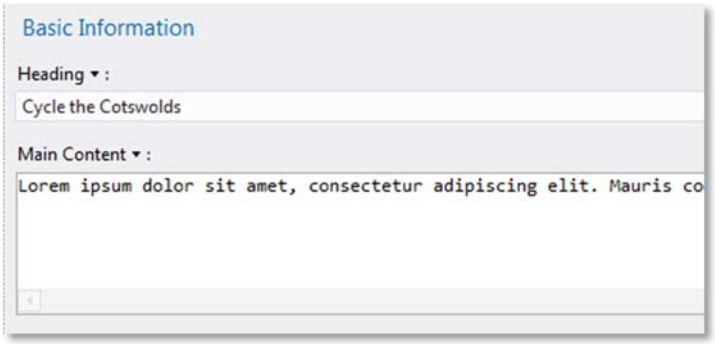
Note that the **Heading** field is replaced by the item's name, and the **Main Content** field contains sample content, *"Lorem ipsum..."*. Populate the remaining fields with sample content for the following table, or write your own.

Discover the Cotswolds	
Basic Information	
Main Image	*Any from the media library*
Trip Information	
Price per person	$800 (or use any value)
Start date	5/1/2013 (or enter any date)

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio, open the **Sitecore Explorer**.
2. Locate the **Family Holidays** item under: */BasicSitecore/master/sitecore/Content/Home/Family Holidays*.
3. Right-click the **Family Holidays** item and select **Add**>**New Item...**
4. Insert a new item based on the **Trip Details** data template. Name this item *Discover the Cotswolds*. Because we have set up default values and tokens, your item will be pre-populated with content:



5. Populate the remaining fields with sample content from the following table, or write your own:

Discover the Cotswolds	
Basic Information	
Main Image	*Any from the media library*
Trip Information	
Price per person	$800 (or use any value)
Start date	5/1/2013 (or enter any date)

6. Click **Save**.

Topic 2.4 Insert Options

Introduction

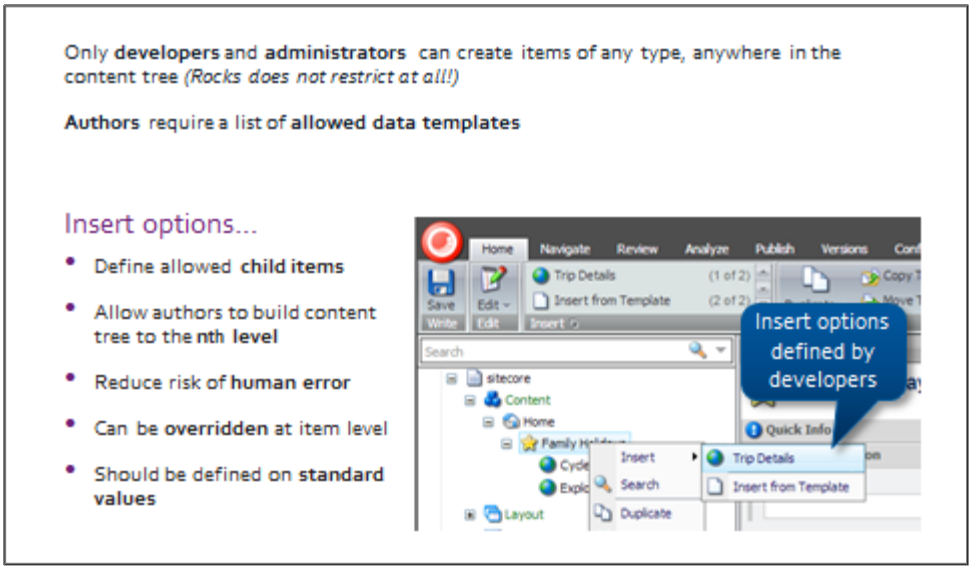
Objectives

By the end of this topic you will be able to:

- Assign insert options to a data template.
- Explain why recommended practice is to assign insert options to standard values.
- Explain why authors require insert options.

Content

What Are Insert Options?



Demo – Assign Insert Options

In this demo, we will:

- *Assign insert options to the **Holidays Section** data template.*
- *Create a new **Trip Details** item under the **Family Holidays** item.*

***Note:** You will be performing these tasks in the upcoming lab.*

Apply – Topic 2.4

Assign Insert Options

Lab A. Add Trip Details as an Insert Option on the Holidays Section Data Template

Overview Steps

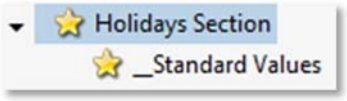
Optional detailed steps are available below to help you.

Using **Sitecore Rocks,** add **Trip Details** as an insert option to the **Holidays Section** data template. Remember to assign insert options on the data template's **standard values**. Test that the data template now appears as an insert option

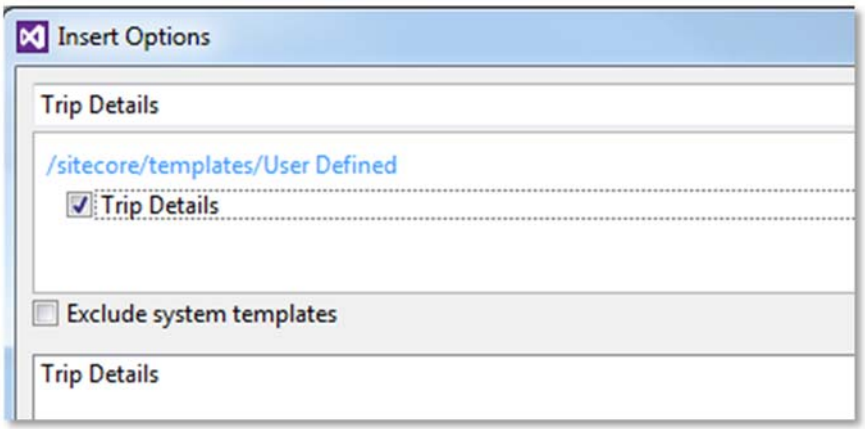
Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio, open the **Sitecore Explorer**.
2. Locate the **Holidays Section** data template under: */BasicSitecore/master/sitecore/Templates/User Defined/Holidays Section.*
3. Double-click the data template to open it.
4. Right-click in the grey space inside the left-hand panel and select **Create Standard Values**.
5. Right-click the **Holidays Section** item in the tree and click **Refresh**. The standard values item will appear as a child of the data template:



6. Right-click the **__Standard Values** item. Select **Tasks>Set Insert Options**.
7. Search for **Trip Details** and select the **Trip Details** checkbox:



8. Click **OK** and **Save**.
9. To test that **Trip Details** now appear as an insert option on the **Holiday Section** data template, right-click on the **Family Holidays** item and ensure that **Trip Details** appears.

Topic 2.5 Summary and Optional Lab

Introduction

Objectives

This topic covers:

- A summary of key vocabulary
- A summary of standard values versus data templates
- A summary of key steps
- An optional lab

Content

Key Vocabulary

Data Template

Defines an item's type. Contains field sections and fields. Should always have a unique icon. Used to create items.

Field

Holds content. Can be different types – single-line text, rich text, image, link. Type determines the interface that the author sees.

Data Template Inheritance

Data templates can inherit fields and field sections from other data templates. Reduces duplication.

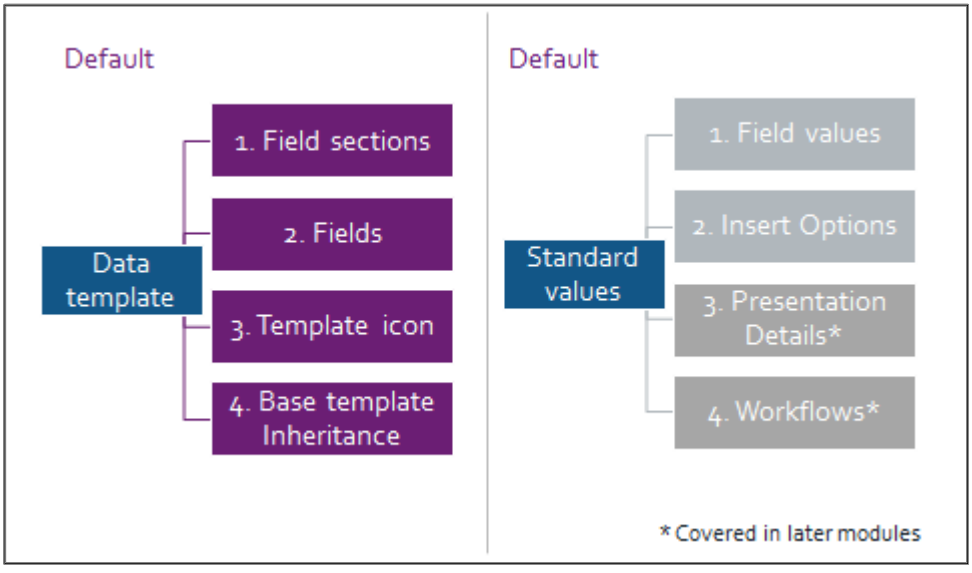
Standard Values

__Standard Values child item of a data template. Vehicle for defaults.

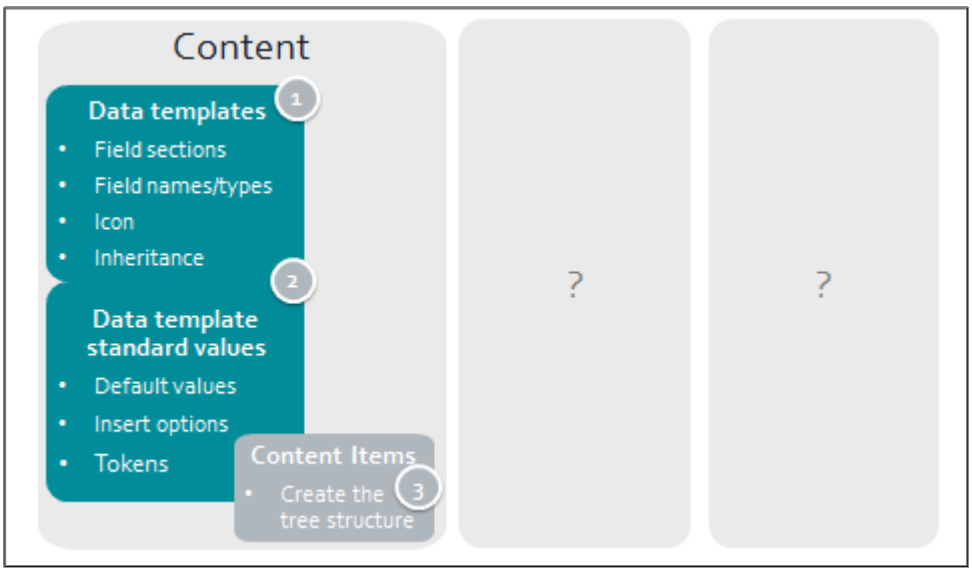
Insert Options

List of item types that can be inserted as a child of a particular item.

Data Template Versus Standard Values



Data Definition Steps



Apply – Topic 2.5 – Optional

Create a Bicycle Details Data Template

Overview Steps

*This is a **summary lab** based on everything you have learned in this module. Refer to previous labs for detailed instructions.*

Using **Sitecore Rocks,** create a **Bicycle Details** data template using the following information:

- The data template must inherit from **Base**.
- The data template must contain the following fields:

Bicycle Details	
Bicycle Information	
Field name	**Field type**
Suitability	Single-line text
Type	Single-line text

- It must be possible for authors to create **Bicycle Details** items under **Trip Details** items.
- The **Suitability** field should contain **"All"** by default.

Module 3
Presentation

Contents

Topic 3.1 Presentation Is Dynamic and Modular

Introduction

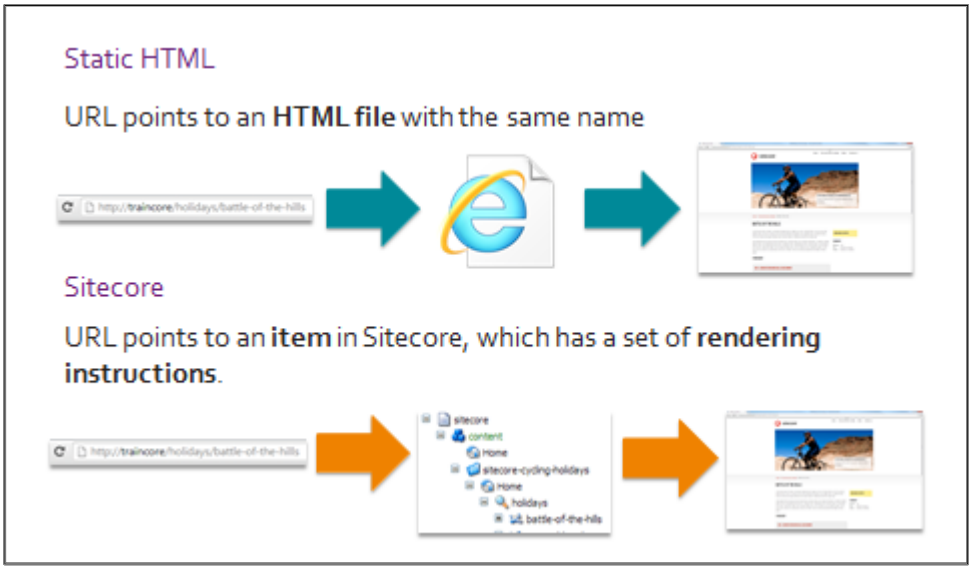
Objectives

By the end of this topic you will be able to:

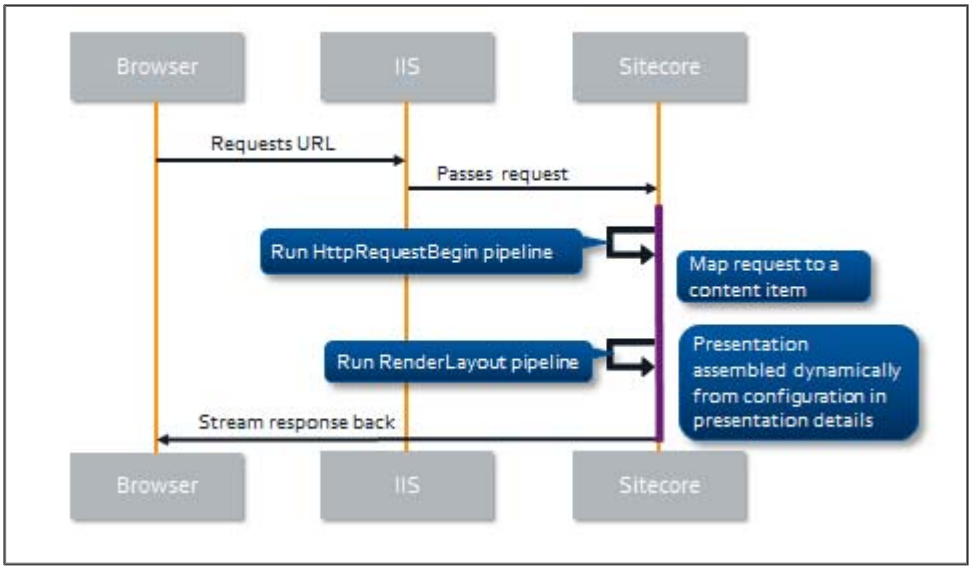
- Explain how Sitecore resolves a page compared to a static site.
- Define presentation details in general terms.

Content

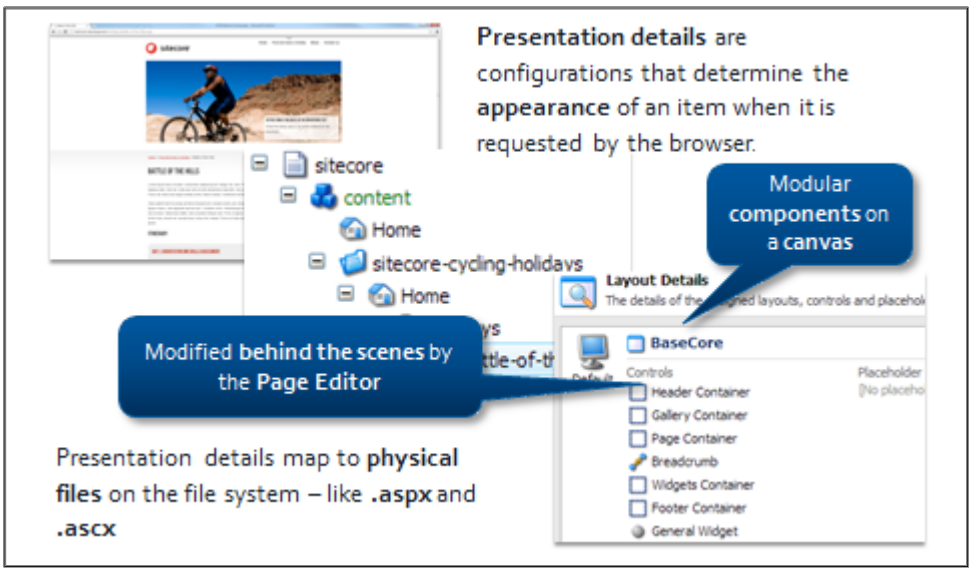
How Sitecore Resolves a Page



A Detailed Look at a Request



Presentation Details



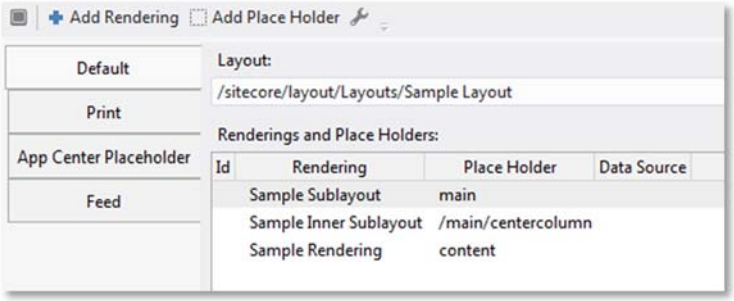
Demo – Modify a Presentation with Page Editor

In this demo we will:

- Create a new **Standard Content** item in **Traincore**.
- Use **Sitecore Rocks** to view the presentation.
- Add a **Banner** and **General Widget** component to the newly created page.
- Use **Sitecore Rocks** to view changes in presentation.

Tip

To view an item's presentation details in Sitecore Rocks, select the item and press CTRL+U:



Topic 3.2 Preparing to Build

Introduction

Objectives

By the end of this topic you will be able to:

- Break a design into components.
- Set up a Visual Studio project to work with Sitecore.

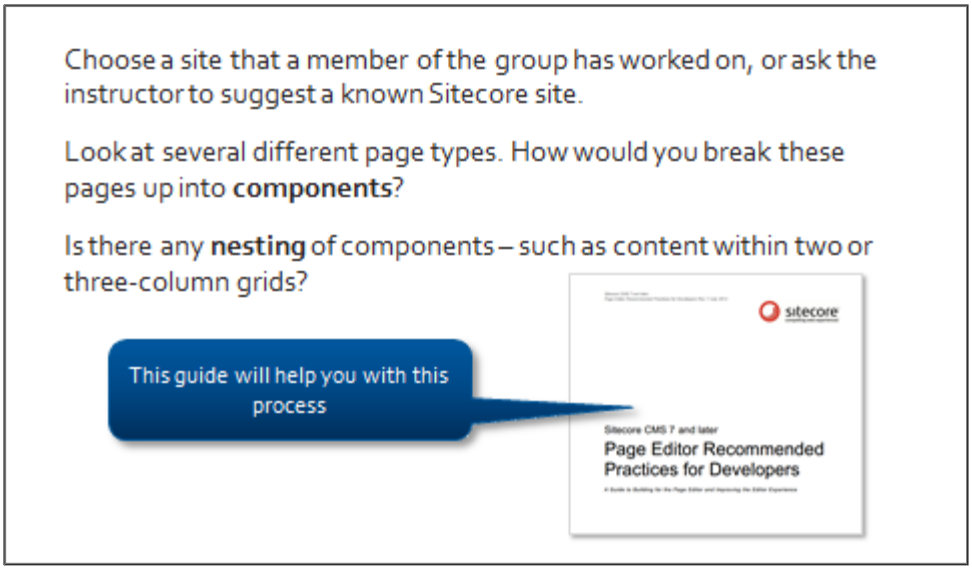
Content

Break Your Design into Components

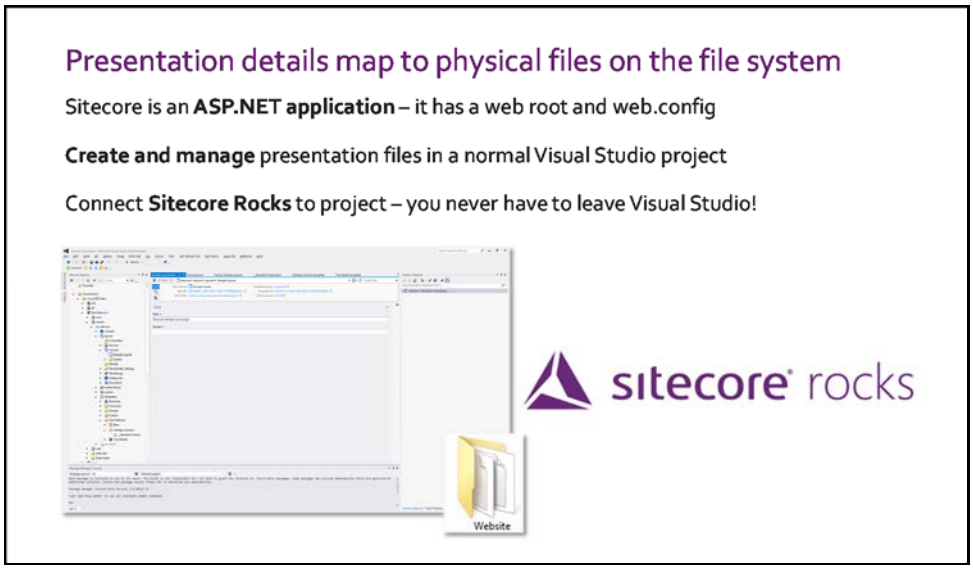
Tip

For more information on how to componentize a page, see the **Page Editor Recommended Practices Guide** for developers on the SDN: http://sdn.sitecore.net/reference/sitecore%207/page%20editor%20recommended%20practices%20for%20developers.aspx

How Would You Componentize Your own Sites?



Create a Visual Studio Project



Demo – Creating a Project

In this demo we will:

- Create a new **ASP.NET Empty Web Application (C#) project** in Visual Studio 2012 called *BasicSitecore* to correspond to our Sitecore instance.
- Move the **.sln** and **.csproj** files into the web root.
- Include the **web.config** and **Layouts folder**.
- Connect the project to **Sitecore Rocks**
- Add a reference to the **Sitecore.Kernell.dll**.

You will perform these tasks in the upcoming demo.

Important

In the development environment, we recommend that you work outside the web root and deploy your files on a fresh Sitecore instance. To do this, use *Publish Profiles* or a product like *Hedgehog's Team Development for Sitecore*.

Apply – Topic 3.2

Create a Visual Studio Project and Connect It to Sitecore Rocks

In the following labs, you will set up a **Visual Studio project** to work with Sitecore.

Lab A. Create an ASP.NET Empty Web Application Project in the Web Root

Overview Steps

Optional detailed steps are available below to help you.

In Visual Studio, create an **ASP.NET Empty Web Application** in the Sitecore website's **web root**. Name it *BasicSitecore.* You do **not** need to create an entirely new folder for *BasicSitecore*.

In the file system, move the following files and folders out of the newly created **project folder** and into the **Website root**:

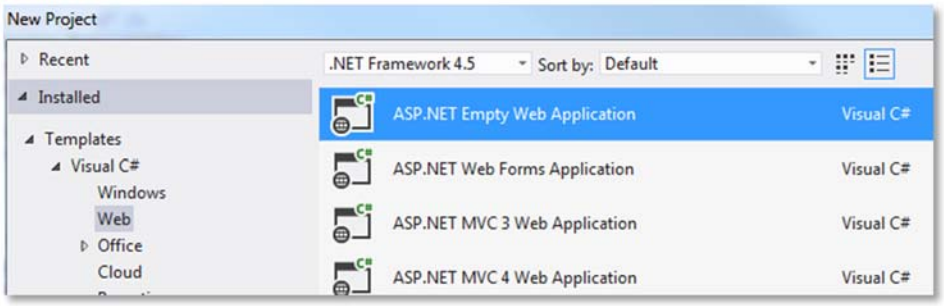
- .sln
- .csproj
- Properties folder
- obj folder

Delete what remains in the project folder. Open the **.sln** file in its new location and move the **layouts** folder into the project.

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

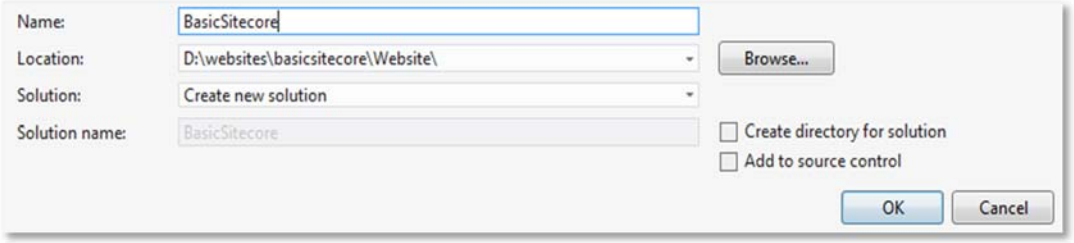
1. Open **Visual Studio**.
2. On the Visual Studio toolbar, select **File>New>Project…**
3. Select **ASP.NET Empty Web Application (C#)** from **Visual C# > Web** section:



4. Name the project *BasicSitecore*, and change the **Location** to the **web root of your Sitecore instance**. This differs depending on the computer you are using, but is likely to be: *C:\inetpub\wwwroot\BasicSitecore\Website.*

   The web root specifically refers to the **Website** folder, where the web.config lives.

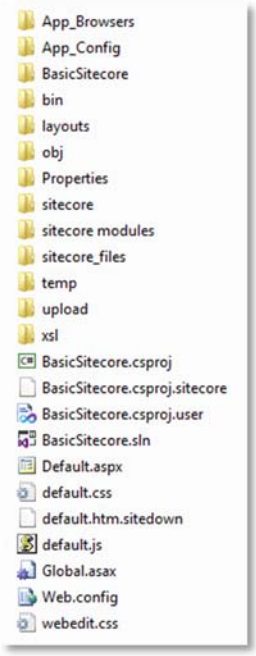
5. Clear the **Create directory for solution** checkbox. Your screen should look something like this:



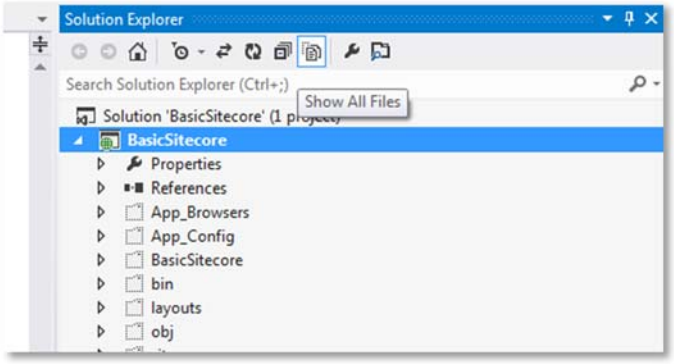
6. Click **OK**.
7. The solution will appear in a **subfolder** in the web root. Some of the contents of this subfolder need to be moved into the web root. **Close** the solution navigating to **File>Close Solution** on the **Visual Studio** toolbar.
8. In the **file system,** navigate to your Sitecore website's web root. For example: *C:\Inetpub\wwwroot\BasicSitecore\Website*
9. Locate the **BasicSitecore** folder that you just created:

10. Open this folder. Copy the following files and folders into the **web root** (one folder above). Your files should end up on the same level as Sitecore's own web.config:
    - BasicSitecore.sln
    - BasicSitecore.csproj
    - Properties
    - obj

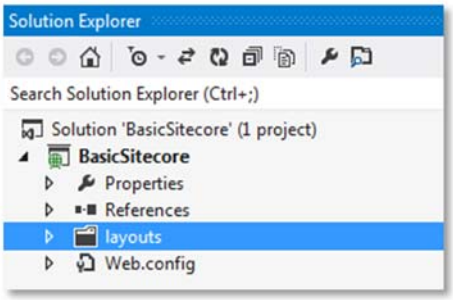
    Do not overwrite Sitecore's own web.config. Your project folders and files are now mixed in with the default Sitecore folders and files in the web root:



11. You can now **delete** the remaining **BasicSitecore** folder from the web root.
12. Go back to **Visual Studio** and open the **.sln** file from its new location inside the web root
13. Double-click the **web.config** file. Confirm that it is Sitecore's web.config file, which is much larger than your project's web.config file (you can also search for **sitecore**).
14. In **Solution Explorer,** select the **BasicSitecore project** and click **Show All Files**:



15. Right-click on the **layouts** folder and select **Include in project**. Click **Show All Files** again to hide all other hidden files. Your final project should look like this:



Lab B. Connect Your Project to Sitecore Rocks

Overview Steps

Optional detailed steps are available below to help you.

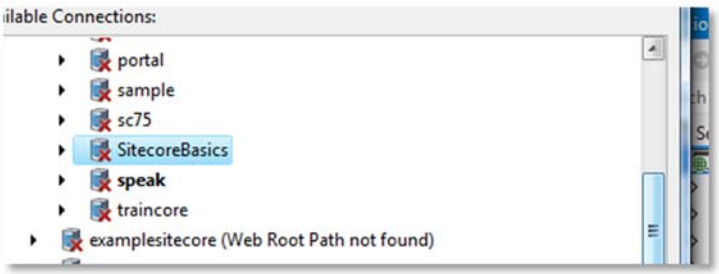
Right-click the **BasicSitecore** project and select **Sitecore>Connect to Sitecore…**

Locate your instance in the list and click **OK** to connect it to your project.

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio, right-click the **BasicSitecore** project.
2. Select **Sitecore>Connect to Sitecore…**
3. Locate the **BasicSitecore** instance in the list, select it, and click **OK**:



Lab C. Add a Reference to the Sitecore.Kernel.dll

Overview Steps

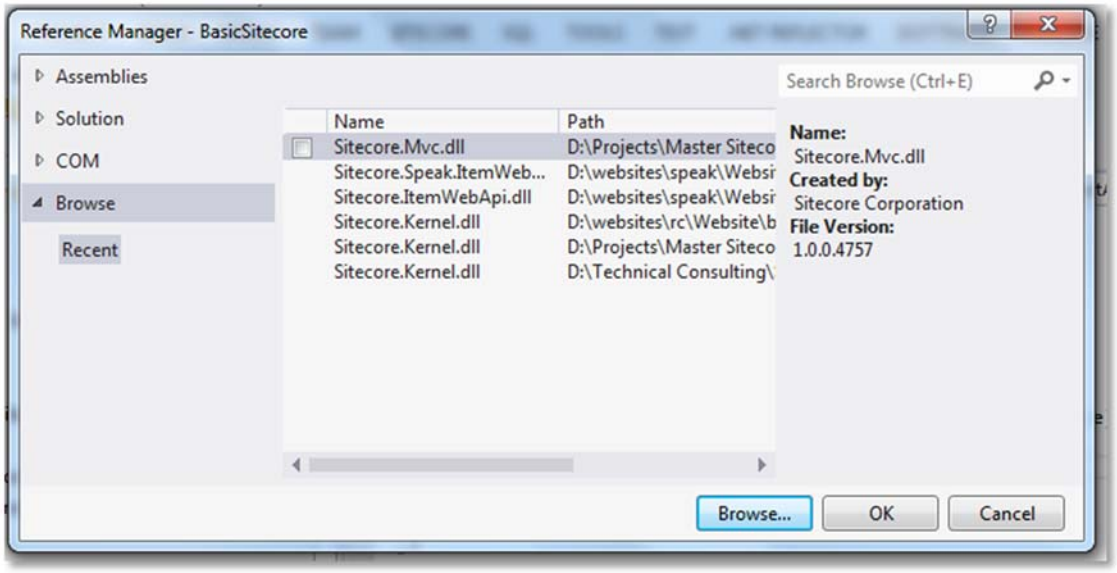
Optional detailed steps are available below to help you.

Add a reference to **libraries\Sitecore.Kernel.dll** to the project.

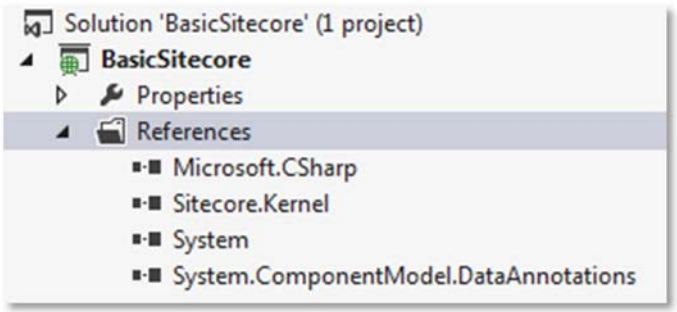
Detailed steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In the web root, create a **libraries** folder.
2. In **Visual Studio,** expand the **BasicSitecore** project.
3. Right-click **References** and select **Add Reference…**
4. Click **Browse** in the bottom right-hand corner:



5. Browse to BasicSitecore **web root** and open the **bin** folder. Your path may be: *c:\inetpub\wwwroot\BasicSitecore\Website\bin*
6. Put a copy of **bin\Sitecore.Kernel.dll** in the **libraries** folder.
7. Select **libraries\Sitecore.Kernel.dll** and click **Add**. Confirm that the Sitecore.Kernel appears in the References list:



Topic 3.3 Creating a Layout

Introduction

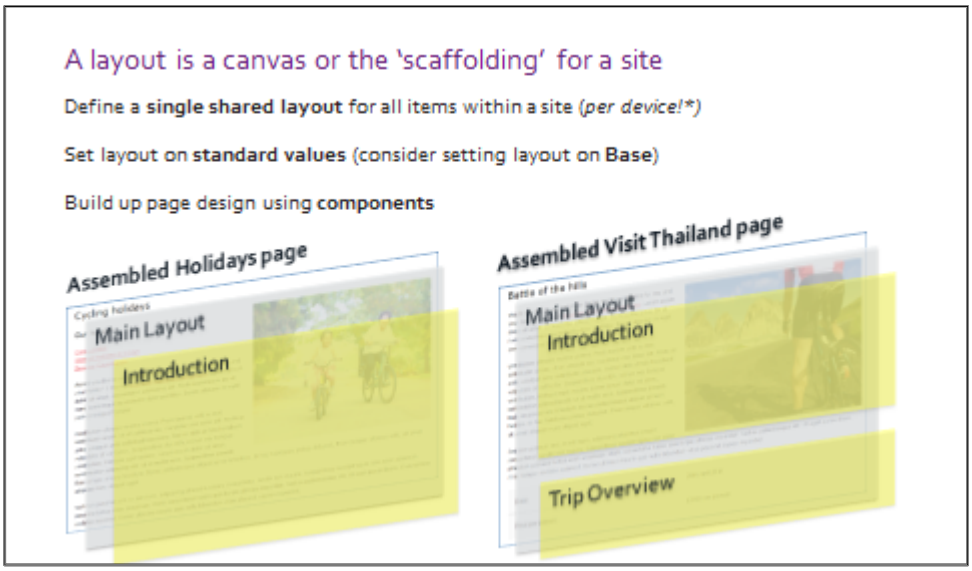
Objectives

By the end of this topic you will be able to:

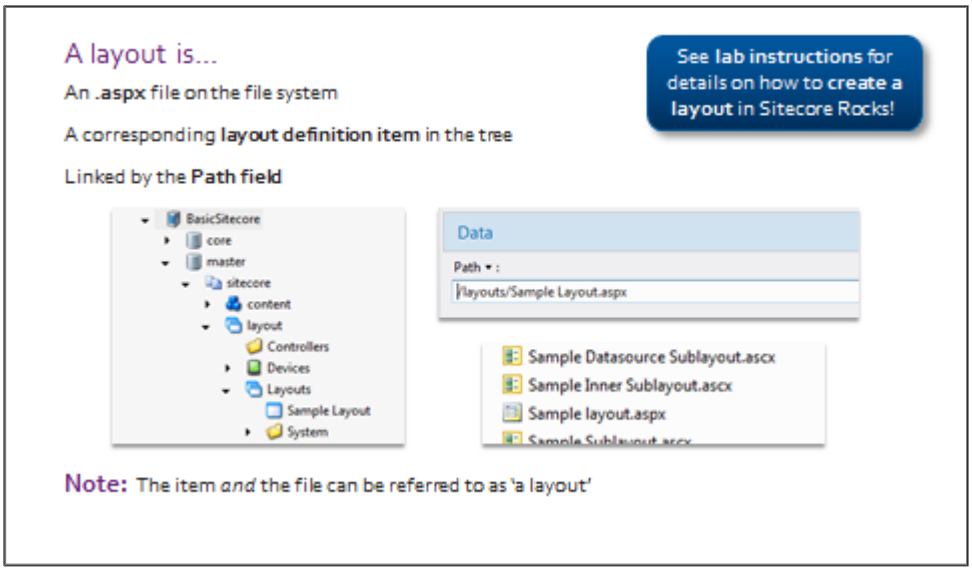
- Create a layout using Sitecore Rocks.
- Assign a layout to a data template's standard values.

Content

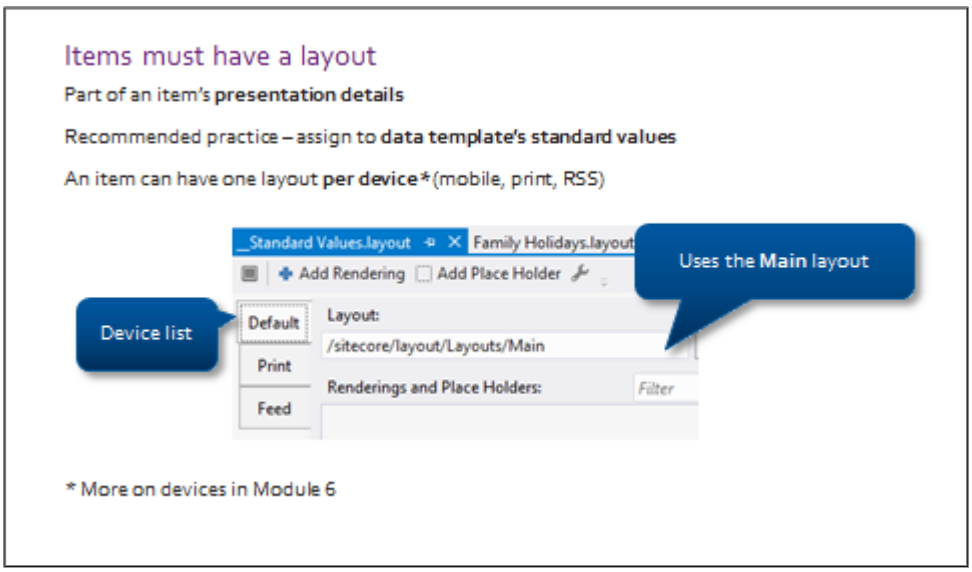
One Layout for All Items Within a Site



What Is a Layout?



Assigning a Layout



Creating and Assigning a Layout

In this demo we will:

- Create a layout using Sitecore Rocks – this creates both the **file** and the **item**.
- Paste **sample HTML** into the layout **.aspx file**.
- Assign layout to Base data template's **standard values**.
- Preview an item that inherits from **Base**.

Apply – Topic 3.3

Creating and Assigning a Layout

In the following labs you will use Sitecore Rocks to:

- Create a layout called **Main** and populate the .aspx file with some sample HTML.
- Assign the layout to the **Base** data template's standard values.
- Preview an existing item.

Lab A. **Create a Layout Called *Main***

Overview Steps

Optional detailed steps are available below to help you.

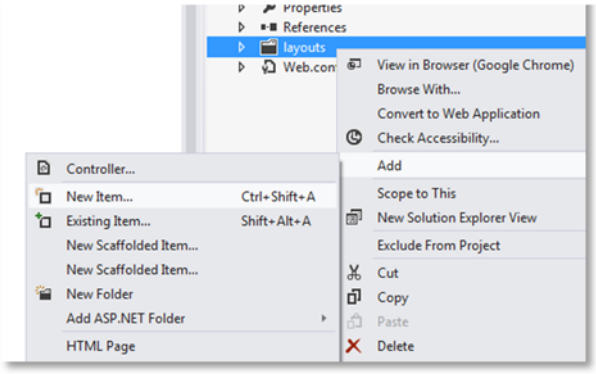
Using **Visual Studio's Solution Explorer,** create a new **Sitecore layout** in the **layouts** folder. When you are prompted by Sitecore Rocks to choose a location for the corresponding item, navigate to: */BasicSitecore/master/sitecore/Layout/Layouts.*

Detailed Steps

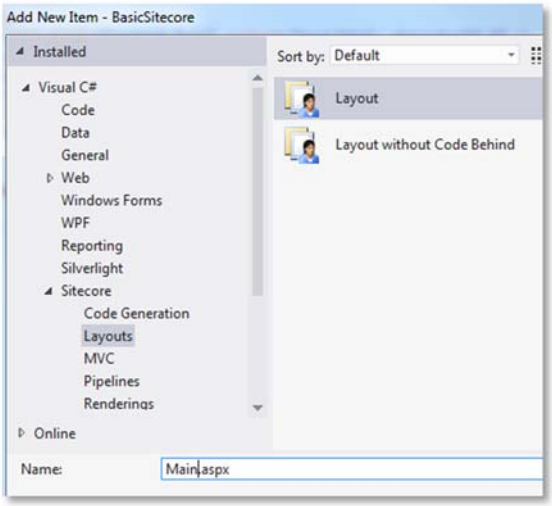
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Open the **Visual Studio Solution Explorer** (*not* the Sitecore Explorer)

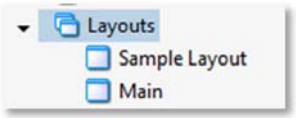
2. Right-click the **Layouts** folder and select **Add>New item...**



3. In the **Add New Item** dialog, locate the **Sitecore** templates in the left-hand column. Expand the tree until you see **Layouts**, and select it.
4. Select **Layout** in the right-hand panel, and call it **Main.aspx**:



5. Click **Add**, and wait for Sitecore Rocks to prompt you to create an item.
6. When prompted by Sitecore Rocks to **create a layout item**, navigate to: */BasicSitecore/master/sitecore/Layouts and select it.*
7. Click **OK**.
8. Using Sitecore Explorer, right-click and **refresh** the /BasicSitecore/master/sitecore/Layouts item.
9. Confirm that that the Main layout item appears:



Lab B. Paste Sample HTML into the Main.aspx File

Overview Steps

Optional detailed steps are available below to help you.

Using Visual Studio's **Solution Explorer ,** paste the following sample HTML into your **Main.aspx** file. Ensure that you do not accidentally delete the page directive at the top.

Drag and drop the **css** and **img** folders from **Student Resources Folder > WND Labs > Module 3 > Topic 3.3 > campaign-page.html** into your Visual Studio solution.

```
<!DOCTYPE html>
<!--[if (gte IE 9)|!(IE)]><!--><html> <!--<![endif]-->
<head>
<meta charset="utf-8" />
<title>Campaign Page</title>
<meta name="description" content="Test" />
<meta name="author" content="" />

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.min.js"></script>
<link rel="stylesheet" href="/css/campaigns.css" />

</head>
<body>
<form id="Form1" runat="server" method="post">
<div class="container">
<div class="indentedSection content">
<!-- HEADING -->
<h1>Cycling holidays</h1>
<!-- MAIN IMAGE -->
<img src="/img/main-image.png" class="right" />
<!-- MAIN CONTENT -->
<ul>
<li><a href="campaign-page-holiday-detail.html">Cycle London</a></li>
<li><a href="campaign-page-holiday-detail.html">Discover Helsinki in 5 Days</a></li>
<li><a href="campaign-page-holiday-detail.html">Discover Copenhagen</a></li>
</ul>
<p>Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of <a
href="#">mountain biking and adventures</a>.</p>
</div>
</div>
</form>
</body>
</html>
```

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Open the Visual Studio's **Solution Explorer** (*not* the Sitecore Explorer).
2. Open **Main.aspx**, located in the **layouts** folder.
3. Delete the sample HTML, taking care **not** to delete the top three lines:

Code Sample

```
<%@ Page language="c#" Codepage="65001" AutoEventWireup="true"
Inherits="BasicSitecore.layouts.Main" CodeBehind="Main.aspx.cs" %>
<%@ Register TagPrefix="sc" Namespace="Sitecore.Web.UI.WebControls" Assembly="Sitecore.Kernel"
%>
<%@ OutputCache Location="None" VaryByParam="none" %>
```

4. Paste the following sample HTML from the **Student Resources Folder > WND Labs > Module 3 > Topic 3.3 > Lab 3.3B-Code-Sample.html** file from the Student Resources folder_v7.2.0.1, into the file below the top three lines:

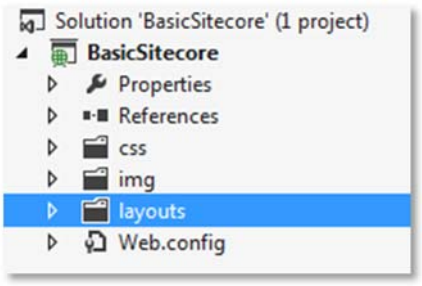
Code Sample

```
<!DOCTYPE html>
<!--[if (gte IE 9)|!(IE)]><!--><html> <!--<![endif]-->
<head>
<meta charset="utf-8" />
<title>Campaign Page</title>
<meta name="description" content="Test" />
<meta name="author" content="" />

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.min.js"></script>
<link rel="stylesheet" href="/css/campaigns.css" />

</head>
<body>
<form id="Form1" runat="server" method="post">
<div class="container">
<div class="indentedSection content">
<!-- HEADING -->
<h1>Cycling holidays</h1>
<!-- MAIN IMAGE -->
<img src="img/main-image.png" class="right" />
<!-- MAIN CONTENT -->
<ul>
<li><a href="campaign-page-holiday-detail.html">Cycle London</a></li>
<li><a href="campaign-page-holiday-detail.html">Discover Helsinki in 5 Days</a></li>
<li><a href="campaign-page-holiday-detail.html">Discover Copenhagen</a></li>
</ul>
<p>Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of <a
href="#">mountain biking and adventures</a>.</p>
</div>
</div>
</form>
</body>
</html>
```

5. **Save** the file.
6. On the **file system,** browse to **Student Resources Folder > WND Labs > Module 3 >Topic 3.3**.
7. **Drag and drop** the **img** and **css** folders into the Visual Studio's **Solution Explorer.** The folders should end up on the same level as the **layouts** folder:



8. **Build** the solution.

Lab C. Assign Layout to Base Standard Values and Preview Family Holidays Item

Overview Steps

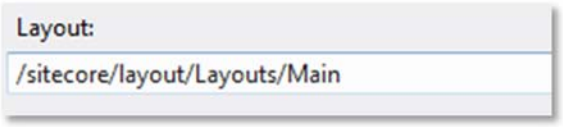
Optional detailed steps are available below to help you.

Using **Sitecore Explorer**, assign the **Main** layout to the **Holiday Section** standard values, preview the **Family Holidays** item. Confirm that your sample HTML is now being displayed, including images and CSS.

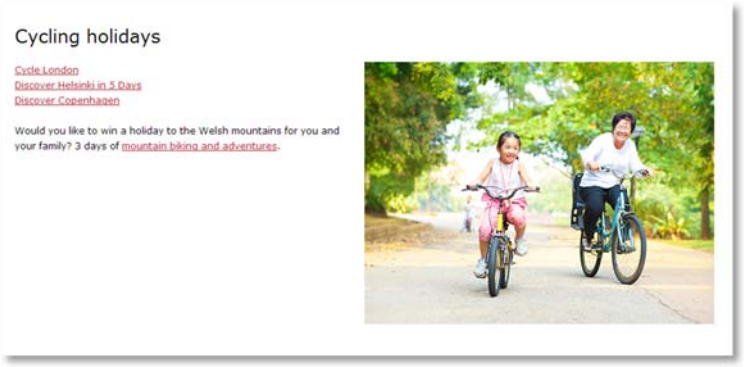
Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Open the **Sitecore Explorer** in Visual Studio.
2. Navigate to: */BasicSitecore/master/sitecore/templates/User Defined/Holiday Section/__Standard Values*
   Right-click on the standard values item and select **Tasks > Design Layout**.
3. In the Layout dropdown, choose the ***Main*** layout:



4. Navigate to: */BasicSitecore/master/sitecore/Content/Home/Family Holidays*
5. Right-click the item and select **Tools>Browse>Preview**.
6. Confirm that the **Family Holidays** item looks like this:



Topic 3.4 Creating Components

Introduction

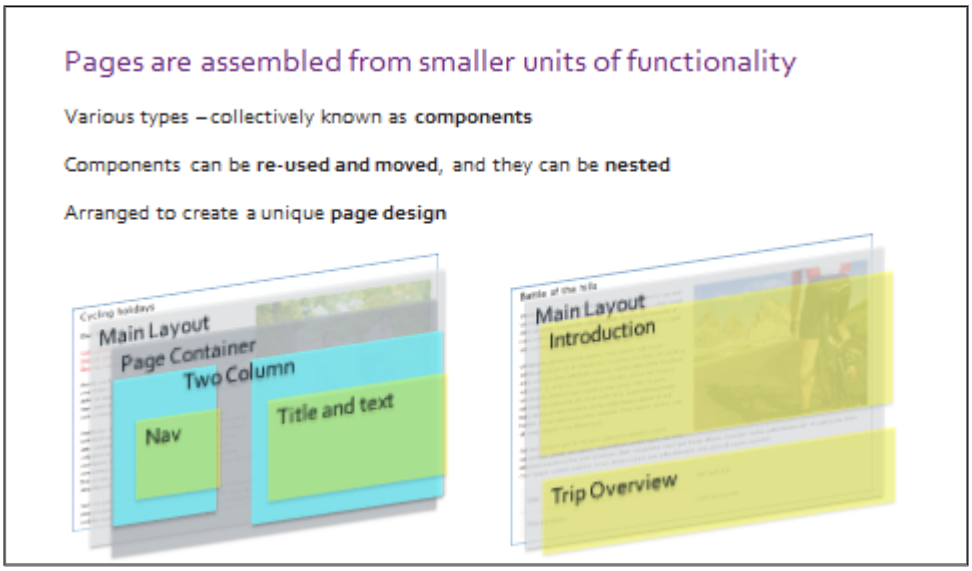
Objectives

By the end of this topic you will be able to:

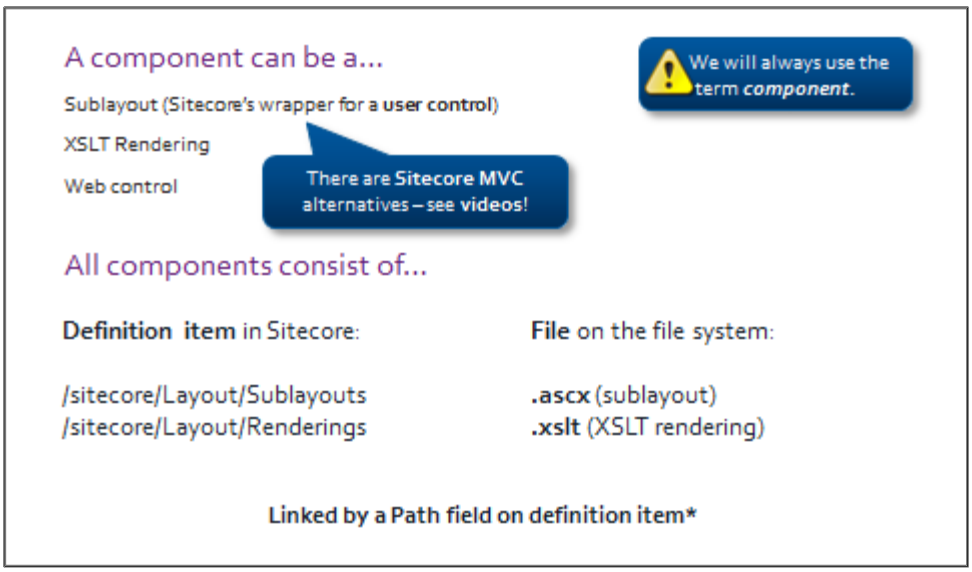
- Create a sublayout using Sitecore Rocks.
- Name three types of Sitecore components.

Content

Movable, Reusable Components



What Is a Component?

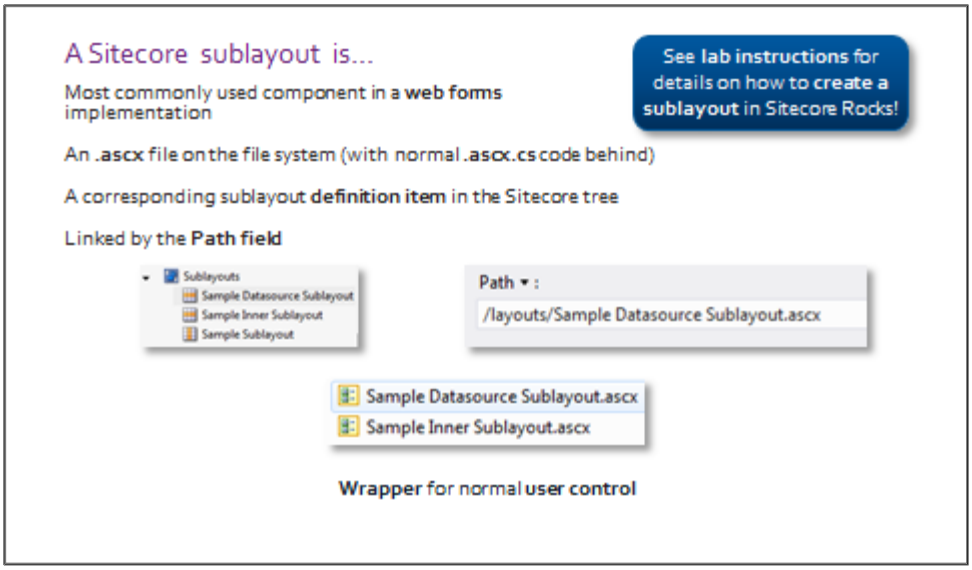


Tip – Sitecore MVC Videos

Part 1: https://www.youtube.com/watch?v=i3Mwcphtz4w

Part 2: https://www.youtube.com/watch?v=dW_rQpgbMmE

What Is a Sitecore Sublayout?



Demo – Creating a Sublayout

In this demo we will:

- Create a sublayout called *Introduction*.
- Move the heading, content, and image from Main.aspx into the Introduction sublayout.

You will perform these tasks in the upcoming lab.

Apply – Topic 3.4

Create a Sublayout

In the following lab you will create a new sublayout called *Introduction* and move the heading, text, and image from the sample HTML in Main.aspx into the Introduction.ascx.

Lab A. Create a Sublayout Called *Introduction*

Overview Steps

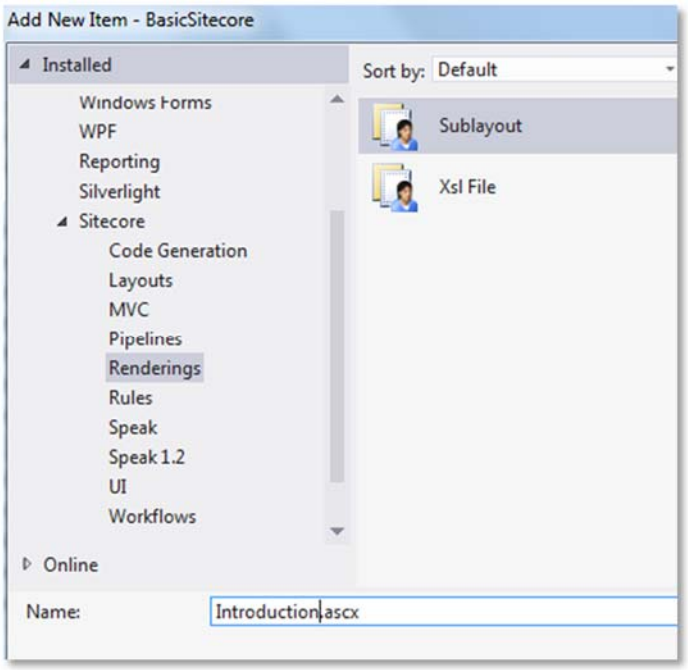
Optional detailed steps are available below to help you.

Using **Sitecore Rocks**, create a new sublayout called *Introduction*. Ensure that the corresponding definition item is created under: */BasicSitecore/master/sitecore/Layout/Sublayouts*.

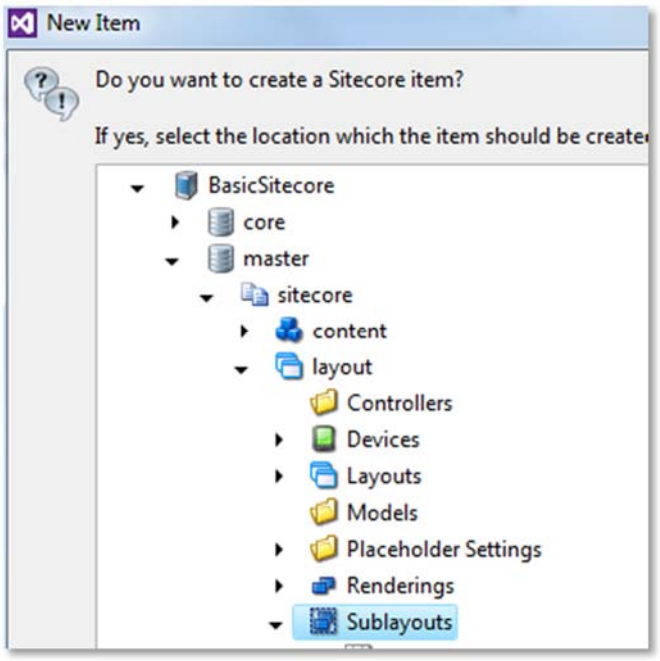
Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio, open the Solution Explorer (*not* Sitecore Explorer).
2. Right-click the **layouts** folder and select **Add>New Item…**
3. In the **Add New Item** dialog, navigate to **Sitecore>Renderings** in the left-hand menu.
4. Select **Sublayout** in the right-hand panel and name it *Introduction*:



5. Click **Add**, and wait for Sitecore Rocks to prompt you to add a corresponding item.
6. In the **New Item** dialog, expand the BasicSitecore tree until you can see the **Sublayouts** item under: */BasicSitecore/master/sitecore/Layout/Sublayouts*:



7. Click **OK**.

Lab B. Move the Heading, Text, and Image Sample HTML from Main.aspx into Introduction.ascx

Overview Steps

Optional detailed steps are available below to help you.

In **Visual Studio**, cut the sample HTML from **Main.aspx** that contains the sample heading, image, and text. Insert it into **Introduction.ascx** instead. For reference, the HTML is available below. Ensure that **Main.aspx** no longer has this HTML.

```
<div class="indentedSection content">
<!-- HEADING -->
<h1>Cycling holidays</h1>
<!-- MAIN IMAGE -->
<img src="/img/main-image.png" class="right" />
<!-- MAIN CONTENT -->
<ul>
<li><a href="campaign-page-holiday-detail.html">Cycle London</a></li>
<li><a href="campaign-page-holiday-detail.html">Discover Helsinki in 5 Days</a></li>
<li><a href="campaign-page-holiday-detail.html">Discover Copenhagen</a></li>
</ul>
<p>Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of <a
href="#">mountain biking and adventures</a>.</p>
</div>
```

Preview the Family Holidays item. The page should be **blank**.

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio's **Solution Explorer**, double-click **Main.aspx** to open it.
2. Locate the following HTML:

   Code Sample

   ```
   <div class="indentedSection content">
   <!-- HEADING -->
   <h1>Cycling holidays</h1>
   <!-- MAIN IMAGE -->
   <img src="/img/main-image.png" class="right" />
   <!-- MAIN CONTENT -->
   <ul>
   <li><a href="campaign-page-holiday-detail.html">Cycle London</a></li>
   <li><a href="campaign-page-holiday-detail.html">Discover Helsinki in 5 Days</a></li>
   <li><a href="campaign-page-holiday-detail.html">Discover Copenhagen</a></li>
   </ul>
   <p>Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of <a
   href="#">mountain biking and adventures</a>.</p>
   </div>
   ```

3. **Cut** this HTML from Main.aspx (or copy it from the code sample above) and **paste** it into **Introduction.ascx**. Take care not to delete the **top two lines** of the .ascx file.
4. Ensure that **Main.aspx** no longer contains the sample HTML above.
5. **Build** the solution.
6. Using **Sitecore Explorer**, right-click: */BasicSitecore/sitecore/master/sitecore/Content/Home/Family Holidays* Then select **Tasks>Browse>Preview**. Confirm that the page is blank.

Topic 3.5 Dynamic Binding

Introduction

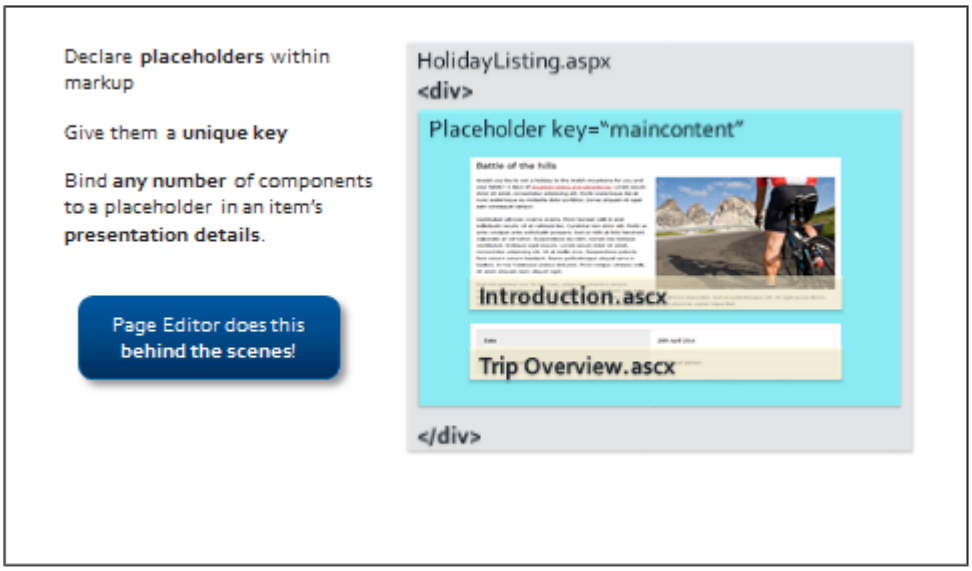
Objectives

By the end of this topic, you will be able to:

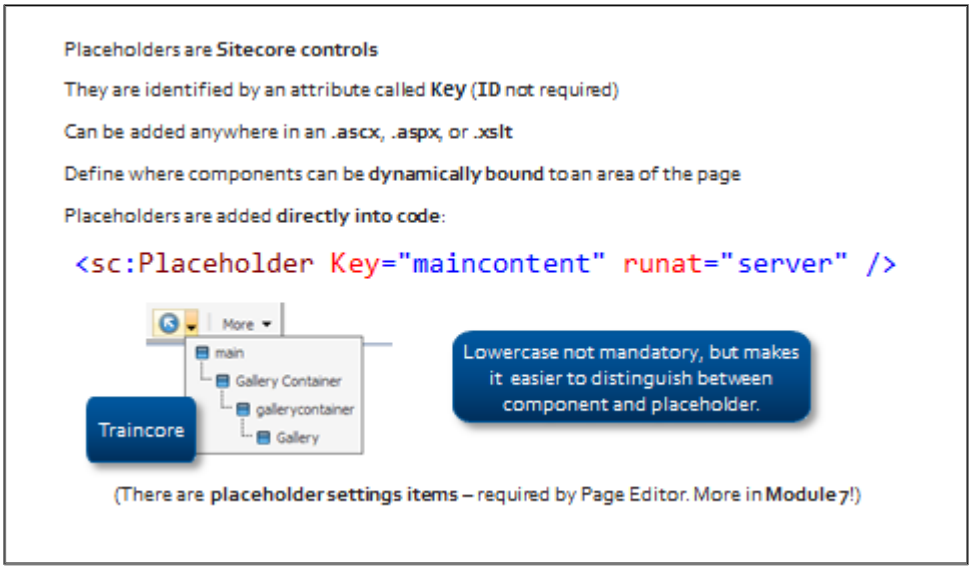
- Assemble a page using dynamic binding.
- Explain the pros and cons of static binding.

Content

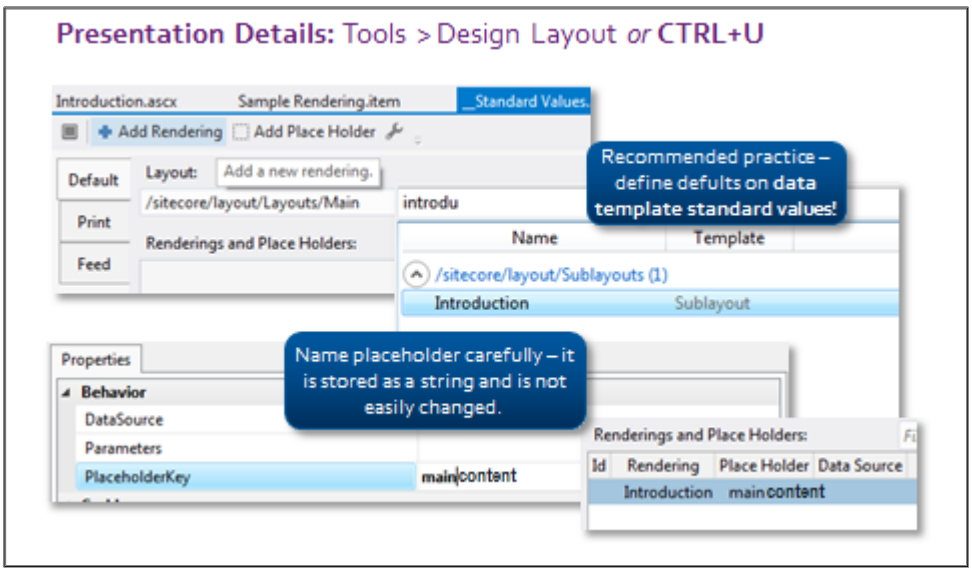
How Do You Bind Components to a Layout?



What Are Placeholders?



Binding Components to Placeholders



Demo – Holidays Section Presentation Details

In this demo we will:

- Create a **maincontent** placeholder in **Main.aspx.**
- Bind **Introduction** component to the **maincontent** placeholder on **Hoidays Section** standard values.
- Preview **Family Holidays.**
- Add an additional Introduction component to Family Holidays and **reset the presentation.**

You will perform these tasks in the upcoming lab.

Dynamic vs. Static Binding

What are the benefits of dynamic binding?

New page types can assembled **from existing components**

Changes to page structure **do not require a developer**

Supports **content re-use** *(more in Module 5!)*

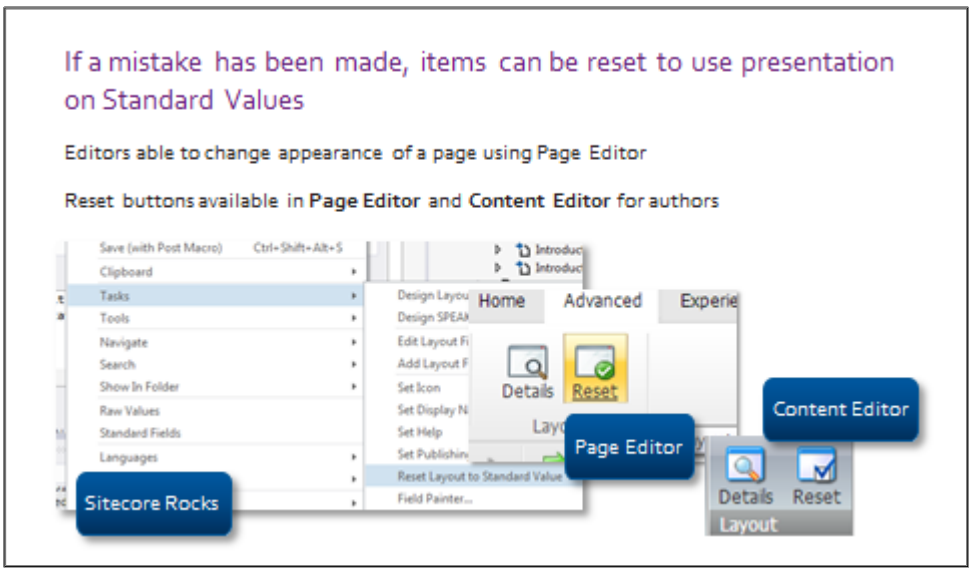
Supports Sitecore's **personalization** and **testing** features *(more in Module 10!)*

Static binding

Some components may benefit from being **statically bound** – such as headers and footers

```
<sc:Sublayout Path="/layouts/Header.ascx"runat="server" />
```

Resetting the Presentation



Apply – Topic 3.5

Bind Components to Placeholders on Standard Values

Lab A. **Add a Placeholder to Main.aspx**

Overview Steps

Optional detailed steps are available below to help you.

In Visual Studio, open **Main.aspx**. Add a placeholder with a key of *maincontent* into the space where you cut the heading, image, and text HTML. **Hint:** inside the tag with the class ***"container"*.**

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Open the Visual Studio's **Solution Explorer** (*not* the Sitecore Explorer).
2. Open **Main.aspx**, located in the **layouts** folder.
3. Inside the tag with the class ***"container"***, insert a placeholder with a key of *maincontent*:

Code Sample

```
<div class="container">
        <sc:Placeholder Key="maincontent" runat="server" />
</div>
```

4. Click **Save**.

Lab B. **Bind *Introduction* to the Maincontent Placeholder on Base Standard Values**

Overview Steps

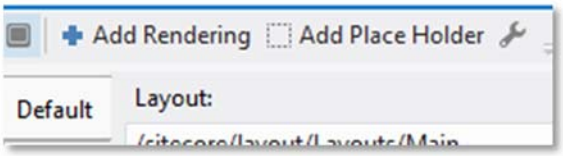
Optional detailed steps are available below to help you.

Assign the ***Introduction*** component to the **maincontent** placeholder on the Base data template's standard values.

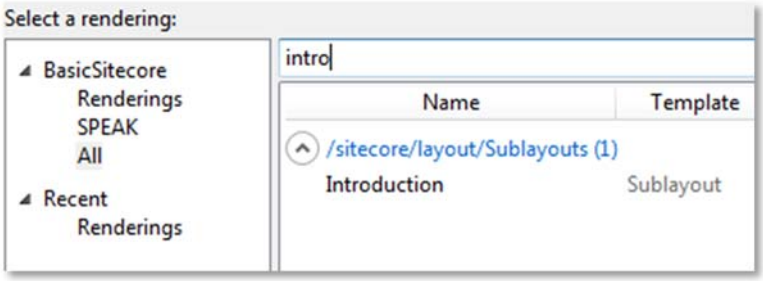
Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

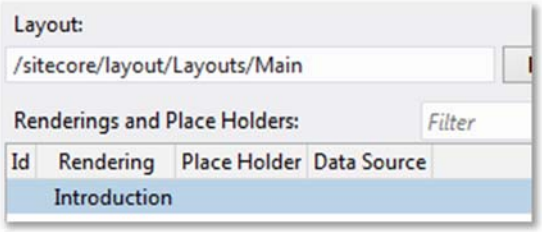
1. Using the Sitecore Explorer, locate the **Base** data template's standard values: */BasicSitecore/master/sitecore/Templates/User Defined/Base/__Standard Values.*
2. Right-click the **__Standard Values** item and select **Tasks>Design Layout**.
3. In the **Presentation Details** window, click **Add Rendering**:



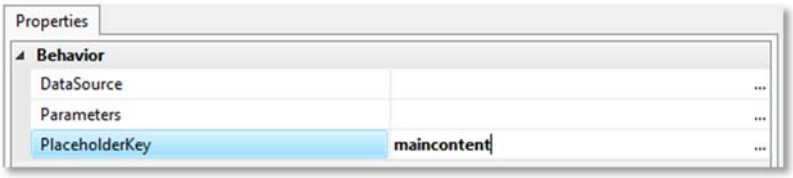
4. Enter ***Introduction*** in the Search field. Ensure that ***All*** is selected on the left-hand menu:



5. Select the **Introduction** rendering and click **OK**.



6. Double-click the **Introduction** rendering to open the **Properties** window.
7. In the **PlaceholderKey** field, type *maincontent*.



8. Click **Save**.

Lab C. Create and Bind the Trip Overview Component to Trip Details Standard Values

Overview Steps

Optional detailed steps are available below to help you.

Create a new component called **Trip Overview**. Use the following sample HTML, or copy from **Student Resources Folder > WND Labs > Module 3 >Topic 3.5 > trip-overview.html**:

```
<!-- THIS IS THE HOLIDAY OVERVIEW SECTION -->
<div class="indentedSection">
        <table>
                <tr>
                        <th>Date</th>
                        <td>20th April 2014</td>
                </tr>
                <tr>
                        <th>Price per  person</th>
                        <td>&pound;2000 per person</td>
                </tr>
        </table>
</div>
<!-- END -->
```

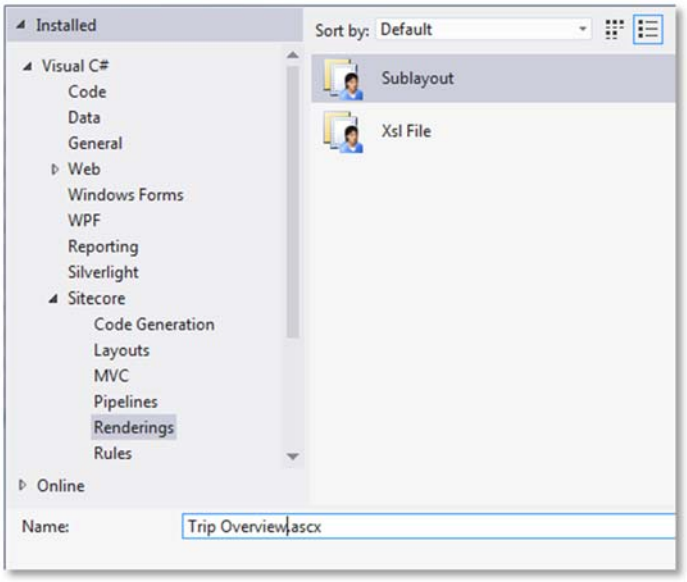
Bind this component to the **maincontent** placeholder on the **Trip Details** data template's standard values. **Hint**: This data template inherits from base. When prompted, copy the layout from the standard values item.

Build the solution and look at the **Explore Holland** item in the Page Editor. Confirm that you are able to select the **Trip Overview** component.

Detailed steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. In Visual Studio's **Solution Explorer,** right-click the **layouts** folder and select **Add>New Item…**
2. Create a new **Sitecore sublayout**. Name it **Trip Overview,** and wait for the prompt from Sitecore Rocks to create a corresponding item.



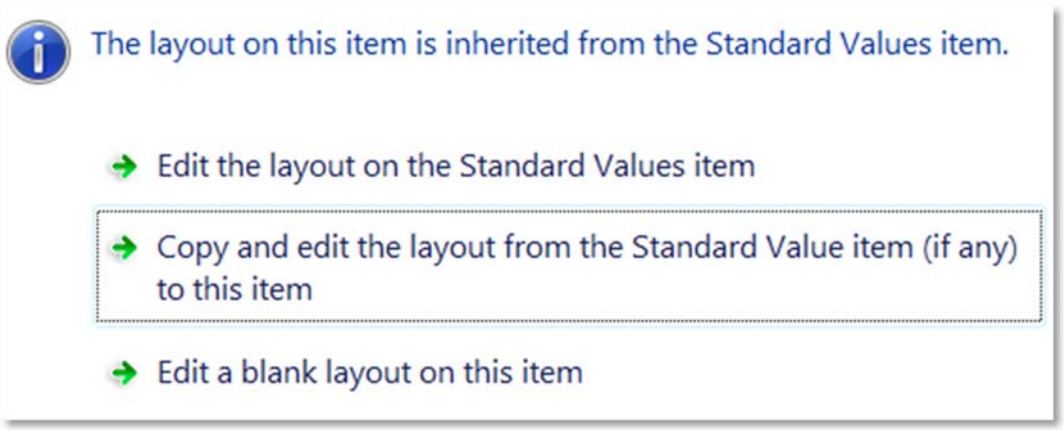
3. When prompted by Sitecore Rocks to create a new **sublayout item,** select: */BasicSitecore/master/sitecore/Layout/Sublayouts* and click **OK**.

4. Open the **Trip Overview.ascx** file.
5. Copy the following **sample HTML** into the .ascx file. Take care not to delete the two lines of code already in the file:

Code Sample

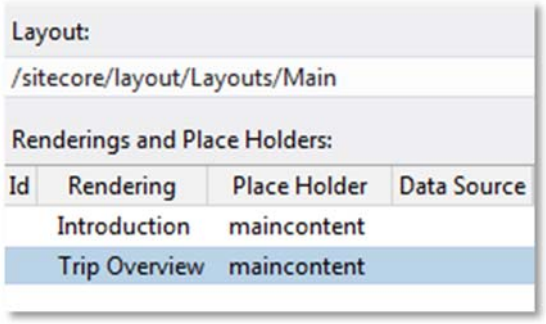
```
<!-- THIS IS THE TRIP OVERVIEW SECTION -->
<div class="indentedSection">
	<table>
		<tr>
			<th>Date</th>
			<td>20th April 2014</td>
		</tr>
		<tr>
			<th>Price per  person</th>
			<td>&pound;2000 per person</td>
		</tr>
	</table>
</div>
<!-- END -->
```

6. **Build** the solution and click **Save**.
7. Using the **Sitecore Explorer**, locate the **Trip Details** item under: */BasicSitecore/master/sitecore/Templates/User Defined/Trip Details.*
8. Double-click the item to open it.
9. Right-click inside the grey area in the right-hand content panel and select **Create Standard Values**.
10. When the newly created **__Standard Values** item appears, press **CTRL+U** to open the **presentation details**. Make sure that you have selected the **__Standard Values** item before you do this.
11. When you are prompted to choose an action, select **Copy and edit the layout from the Standard Value item (if any) to this item**:

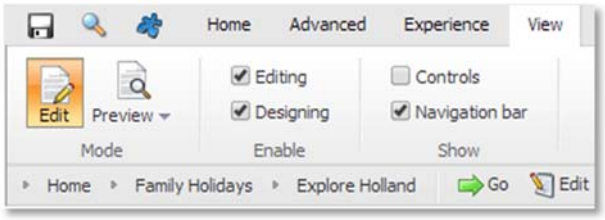


12. Click **Add Rendering**.
13. Search for the **Trip Overview** component, select it, and click **OK**.
14. Double-click on the **Trip Overview** component to open its properties.

15. In the **PlaceholderKey** field, type *maincontent*.



16. Click **Save**.
17. In Sitecore Explorer, navigate to */BasicSitecore/master/sitecore/Content/Home/Family Holidays* and select the **Explore Holland** item.
18. Right-click and select **Tools>Browse>Page Editor**.
19. On the **View** tab of the Page Editor, ensure that the **Designing** checkbox is selected:



20. Confirm that you are able to select the **Trip Overview** component.

Topic 3.6 Outputting Content

Introduction

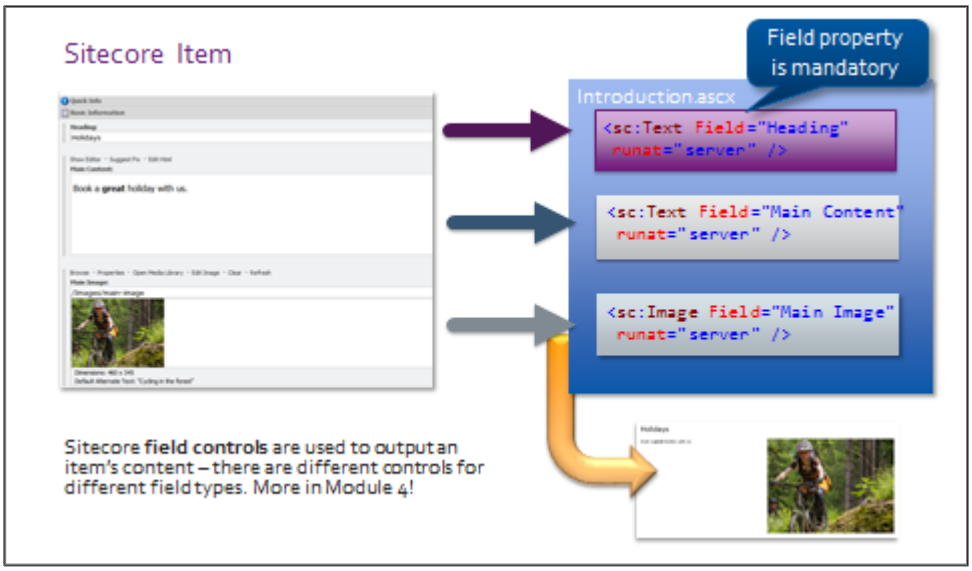
Objectives

By the end of this topic, you will be able to:

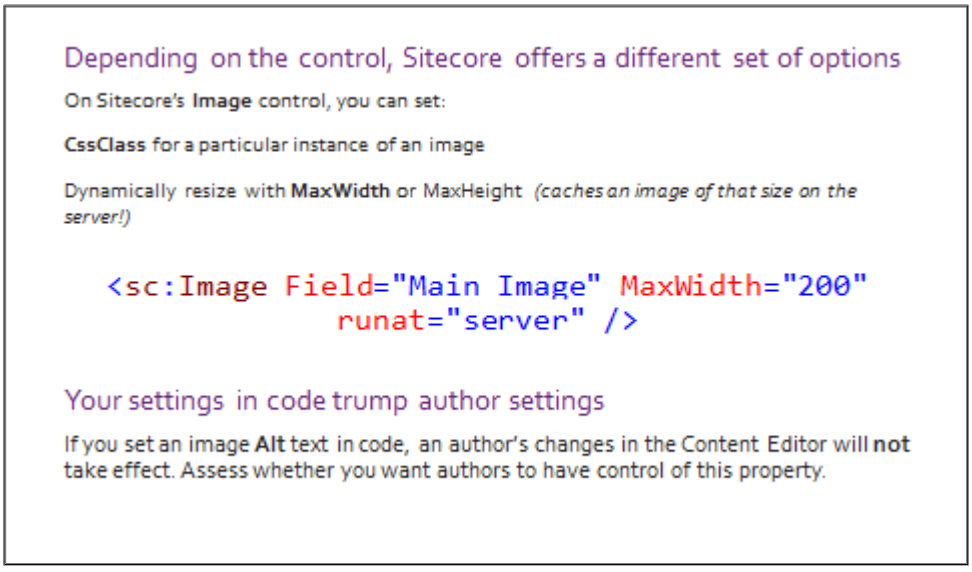
- Output content using a number of Sitecore controls.
- Use Sitecore control parameters to vary the output of Sitecore controls.

Content

How Do You Output Content?



Sitecore Controls Have Optional Attributes



Apply – Topic 3.6

Output Content Using Sitecore Controls

Lab A. **Swap All Sample Content for Sitecore Controls**

Overview Steps

Optional detailed steps are available below to help you.

In Visual Studio, open **Trip Overview.ascx** and **Introduction.ascx**. Replace all sample content with actual content by using Sitecore controls. Match the control used to the type of content being output.

Open various pages in the Page Editor and confirm that you are able to edit all content fields.

Detailed Steps

*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

1. Open the Visual Studio's **Solution Explorer**.
2. Open the **Introduction.ascx** file.
3. Replace the sample content inside the <h1> tag with a Sitecore **Text control** that outputs the **Heading** field:

Code Sample

```
<h1><sc:Text Field="Heading" runat="server" /></h1>
```

4. Replace the **image tag** with a Sitecore **Image control** that outputs the **Main Image** field.
5. Replace the sample text (*everything* under the green <!-- MAIN CONTENT --> comment, including paragraphs) with a Sitecore Text control that outputs the **Main Content** field.
6. Click **Save**.
7. Open **Trip Overview.ascx**.
8. Replace the sample date with a Sitecore **Date control** that outputs the **Starting date** field.
   Replace the sample price with a Sitecore **Text control** that outputs the **Price per person** field.

Tip

Keep in mind that that field names are **case sensitive**.

9. Click **Save**.
10. In **Sitecore Explorer,** locate the **Explore Holland** item: */(BasicSitecore/master/sitecore/Content/Home/Family Holidays/Explore Holland*).
11. Right-click the **Explore Holland** item and select **Tools>Browse>Page Editor**.
12. In the Page Editor, select the **Editing** checkbox on the **View** tab.
13. Click into the **Heading** field – you are now able to edit this field in the Page Editor:

Topic 3.7 Summary and Optional Lab

Introduction

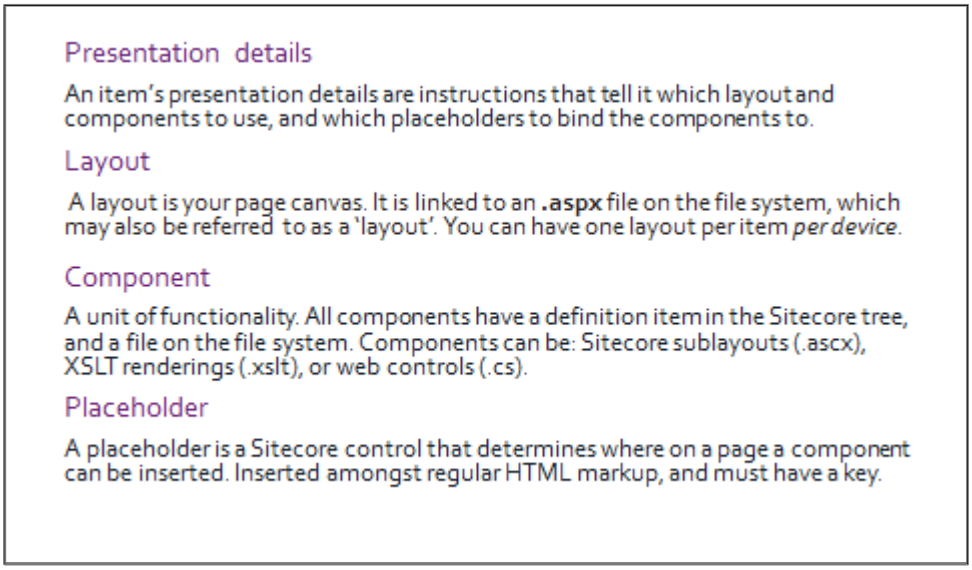
Objectives

This topic covers:

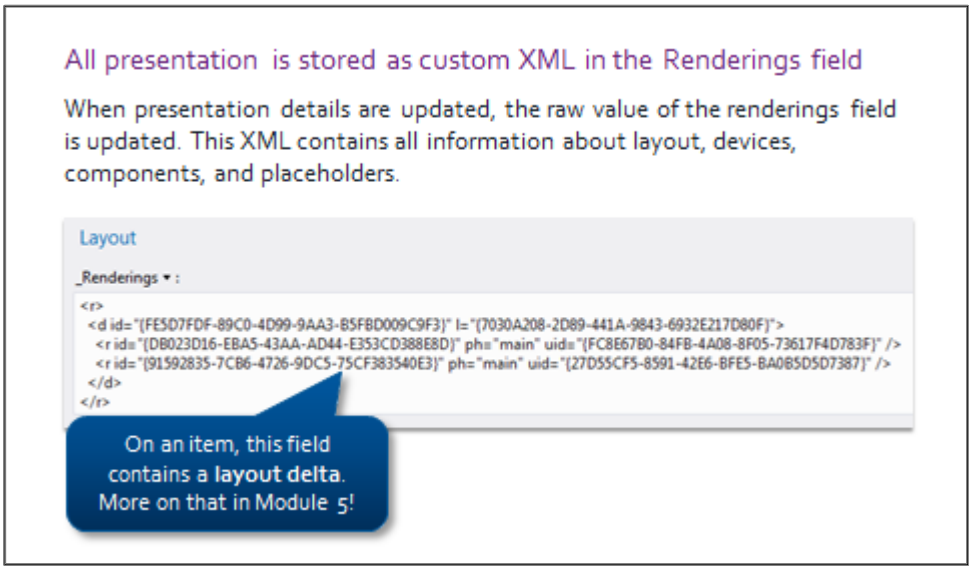
- A summary of key vocabulary
- The Renderings field
- A summary of key steps
- An optional lab

Content

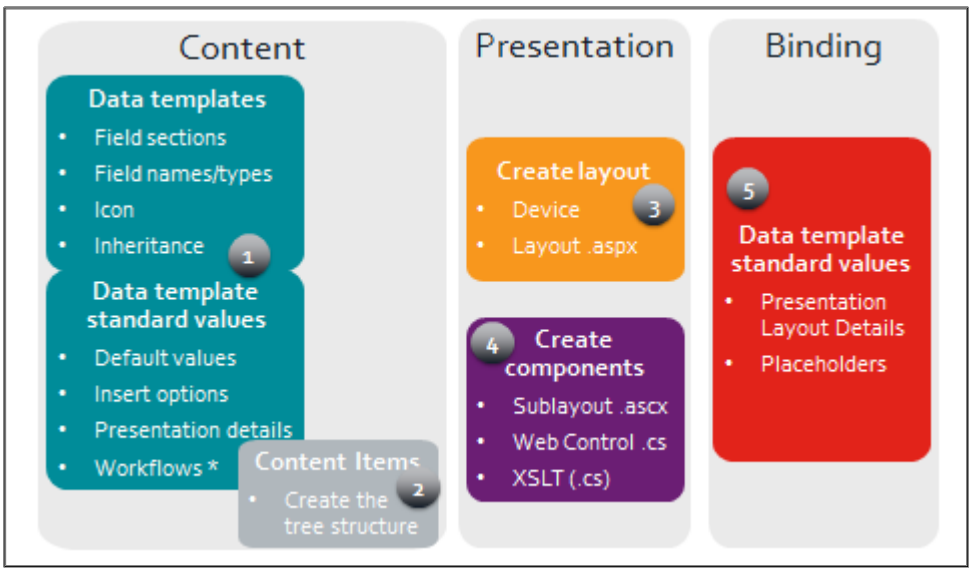
Vocabulary



The Renderings Field



Review of Steps



Apply – Topic 3.7 – Optional

Set a Default Presentation on the Bicycle Detail Data Template

Overview Steps

*This is a **summary lab** based on everything you have learned in this module. Refer to previous labs for detailed instructions.*

Using **Sitecore Rocks,** create a **Bicycle Overview** component using the following sample HTML or copy from **Student Resources Folder > WND Labs > Module 3 > Topic 3.7 > bicycle-overview.html**:

```
<div class="indentedSection">
	<table class="bikes">
		<tr>
			<th>Type</th>
			<td>Hybrid</td>
		</tr>
		<tr>
			<th>Suitability</th>
			<td>Family outings, country lanes</td>
		</tr>
	</table>
</div>
```

Set up a **default presentation** for the **Bicycle Details** data template that includes the **Introduction** component and the new **Bicycle Overview** component.

Module 4
Sitecore API

Contents:

- Basic API Concepts and Retrieving Items
- Item Links
- Creating, Deleting, and Modifying Items
- Working with Complex Fields

Topic 4.1 Basic API Concepts and Retrieving Items

Introduction

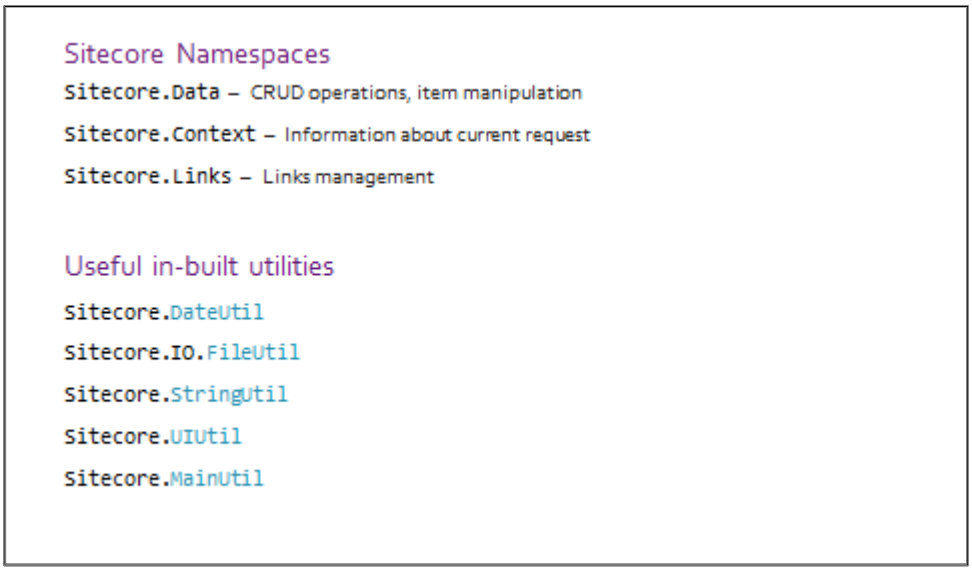
Objectives

By the end of this topic you will be able to:

- Describe the information the Sitecore.Context class provides.
- Discuss how Sitecore code is debugged in a .NET solution.
- Use the GetItem() method to retrieve items.
- Obtain a site's start item.
- Describe how to get items from another database.

Content

Sitecore.Kernel Assembly



Knowledge Check

When you view a page in *Preview* mode, which context database is accessed?

Sitecore Context

Sitecore Namespaces

`Sitecore.Data` – CRUD operations, item manipulation

`Sitecore.Context` – Information about current request

`Sitecore.Links` – Links management

Useful in-built utilities

`Sitecore.DateUtil`

`Sitecore.IO.FileUtil`

`Sitecore.StringUtil`

`Sitecore.UIUtil`

`Sitecore.MainUtil`

Debugging

Sitecore is a regular .NET application so you can use the usual Visual Studio debugging techniques:

Attach to w3wp (tip: if you have more than one instance listed, select *all* to save you time)

You can use Sitecore Rocks to attach to worker process (use commandy)

Step through your code

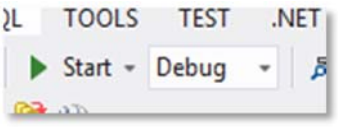
F5 not recommended

```
11  public partial class GeneralWidget : BaseWidget
12  {
13      private void Page_Load(object sender, EventArgs e)
14      {
15          Item source = this.GetItem();
16
17          if (source != null)
18          {
```

Demo – Basic API

In this demo we will:

- Explore Sitecore.Context



1. In **Visual Studio,** open the **code behind** for any existing sublayout (for example, *Introduction.ascx.cs*).
2. Ensure that the **Debug** configuration is selected.
3. In the **Page_Load** method, **retrieve** a few **properties** from the Sitecore.Context class (for example, Database, Item, User, and Language).
4. **Build** the solution.
5. Insert a **break-point,** attach it to **w3wp,** and navigate to a page that uses that sublayout to execute the code.

The GetItem() Method

Sitecore is a regular .NET application so you can use the usual Visual Studio debugging techniques:

Attach to w3wp (tip: if you have more than one instance listed, select *all* to save you time)

You can use Sitecore Rocks to attach to worker process (use commandy)

Step through your code

F5 not recommended

```
11  public partial class GeneralWidget : BaseWidget
12  {
13      private void Page_Load(object sender, EventArgs e)
14      {
15          Item source = this.GetItem();
16
17          if (source != null)
18          {
```

Getting Items from Other Databases

The `Sitecore.Context.Database` object will typically point to the **web** database when browsing the site

If you want to get an item from **another database** (e.g., editing an item in the **master** database), get the desired database by name:

```
Sitecore.Configuration.Factory.GetDatabase("master");
```

Use the same `GetItem()` method directly on the database object:

```
Database master = Factory.GetDatabase("master");
Item item = master.GetItem("/sitecore/content/home");
```

Important

You must *not* query the Sitecore databases directly. (This excludes the analytics database.) Always go through the API.

Tip

```
Item item1 = Sitecore.Context.Database.GetItem("/sitecore/content/home");

Item item2 = Sitecore.Context.Database.GetItem("/sitecore/content/home");

if(item1==item2) // this is always false;
```

You are comparing references to objects. The variables above are pointing to different objects, although they are the same item in Sitecore.

Since Sitecore items are uniquely identifiable by their IDs, comparisons should be done on the item ID.

```
if(item1.ID==item2.ID) // do something;
```

Demo – Retrieving Items

In the following demo, we will

- Use .GetItem() to retrieve an item by path and ID
- Explore Item properties and methods
- Use .GetChildren()and .Children

1. In **Visual Studio**, open the **code behind** for any existing sublayout (for example, *Introduction.ascx.cs*)
2. **Retrieve** an **item** from the Sitecore tree using the Sitecore.Context.Database.GetItem() method – either by passing in a **path,** or an **ID**:

Code Sample – Item Properties

```
Sitecore.Context.Database.GetItem("/sitecore/content/home");
Sitecore.Context.Database.GetItem(new ID("{E1B92921-75C9-475A-A04C-67401C54B39A}"));
```

3. Look at the following properties and methods of the Item class (try using the */sitecore/content/Home* item):

Code Sample – Item Properties and Methods

```
Sitecore.Context.Item.Parent.Name;

Sitecore.Context.Item.GetChildren();

Sitecore.Context.Item.Children;
```

4. Notice that Sitecore.Context.Item.GetChildren() allows you to pass in a Sitecore.Collections.ChildListOptions enum (for example, to ignore any security applied on those items).
5. **Build** the solution.
6. Add a **break-point** and attach it to **w3wp.**
7. In a browser, navigate to a page that uses the sublayout that you modified. Step through to see what each property or method returns.

Apply – Topic 4.1 – 30 min

Render Item Children

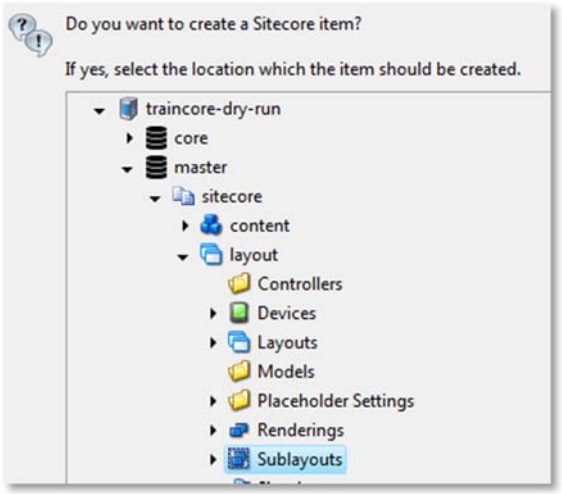
In the following lab, you will:

- Build a subnavigation component to output the children of the context item in a list. Leave the **href** attribute blank.
- In a later exercise, you will populate the link's **href** attributes with the item URL.

Lab A. Build a Navigation Sublayout

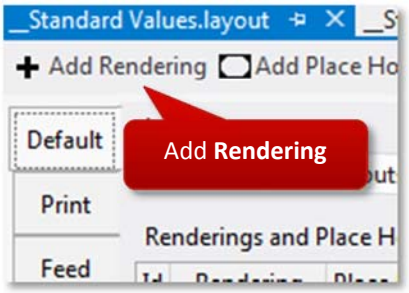
1. Using **Visual Studio**, add a new sublayout called *Subnavigation* (use **Add > New Item…**).
2. Navigate to Sitecore and, on the **Renderings** menu, select the **Sublayout** option.

3. You will be prompted to add an accompanying sublayout definition item. Select your **location** in the **Sitecore tree.**



4. Copy the **HTML** from the **student resource folder** *(WND Labs > Module 4 > Topic 4.1 > Lab A > HTML > campaign-page-navigation.html)*, and paste it into the .ascx file.
Do not overwrite everything in the .ascx. Keep the directives and includes at the top of the file.
5. The finished product will look like this:

6. Using **Sitecore Rocks**, navigate to the **Holidays Section** data template standard values at */sitecore/templates/User Defined/Holidays Section/__Standard Values.*
7. Right-click the **standard values** item and select the **Tasks > Design Layout** options, or use Commandy.
8. Add a new rendering using the **Add Rendering** button.



9. Add the **Subnavigation** sublayout to the **main** placeholder. Ensure that it is listed below the **Introduction** sublayout.
10. **Save** your work and **preview** the **Family Holidays** item. You should see the **Subnavigation** sublayout at the bottom of the page. (If you do not see it, reset presentation details to standard value).
11. In **Visual Studio**, navigate to the **Subnavigation** sublayout's **code behind.**
12. In the **Page_Load** method, **get** the **context item's children** (trip details items) as a **list.**

13. Bind this list to a **repeater.** There is a partially completed code sample in the **student resource folder** *(WND Labs > Module 4 > Topic 4.1 > Lab A > Code > Subnavigation.ascx / Subnavigation.ascx.cs)*.

14. For each item that has been bound to the repeater, output a standard .NET HyperLink object into the repeater's <ItemTemplate>.

    IMPORTANT! For the purposes of this lab, set the NavigateURL property to #

Code Sample – Item Properties

```
<li>
<asp:HyperLink Text="[ITEM NAME]" NavigateUrl="#" runat="server" />
</li>
```

Tip

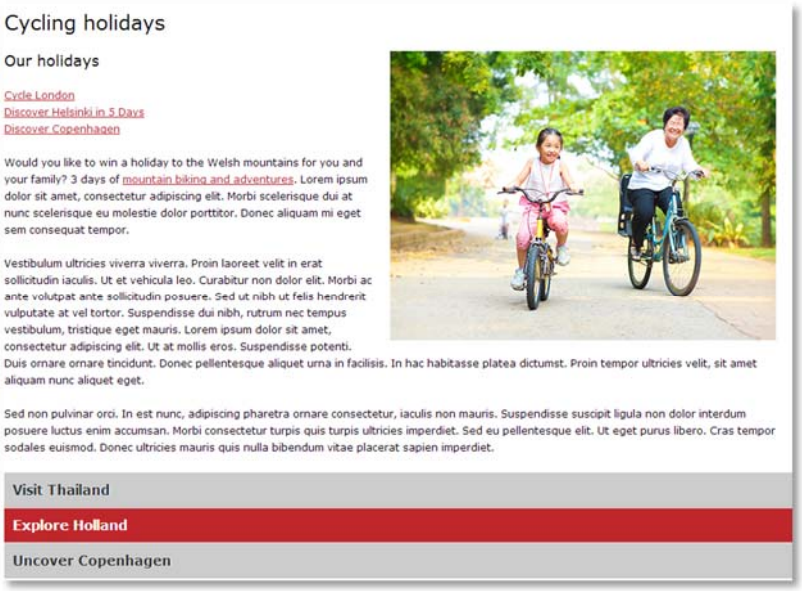
You do not need to use the **OnItemDataBound** property for this exercise.

The sample code uses a strongly-typed repeater – ItemType="Sitecore.Data.Items.Item". This means that you can use the following syntax to output properties of the object, for example, the parent ID of the item. **Note** the colon after the #. This automatically escapes any HTML.

```
<%#: Item.ParentID %>
```

15. **Save.**

16. Using a browser, navigate to: http://BasicSitecore/**Family Holidays**. Confirm that the Subnavigation component now appears and that it is listing the child items by **item name**:



Extend

- **Content API Cookbook**
  http://sdn.sitecore.net/upload/sitecore6/64/content_api_cookbook_sc64_and_later-a4.pdf
- **Data Definition API Cookbook**
  http://sdn.sitecore.net/Reference/Sitecore%207/Data%20Definition%20API%20Cookbook.aspx

Topic 4.2 Item Links

Introduction

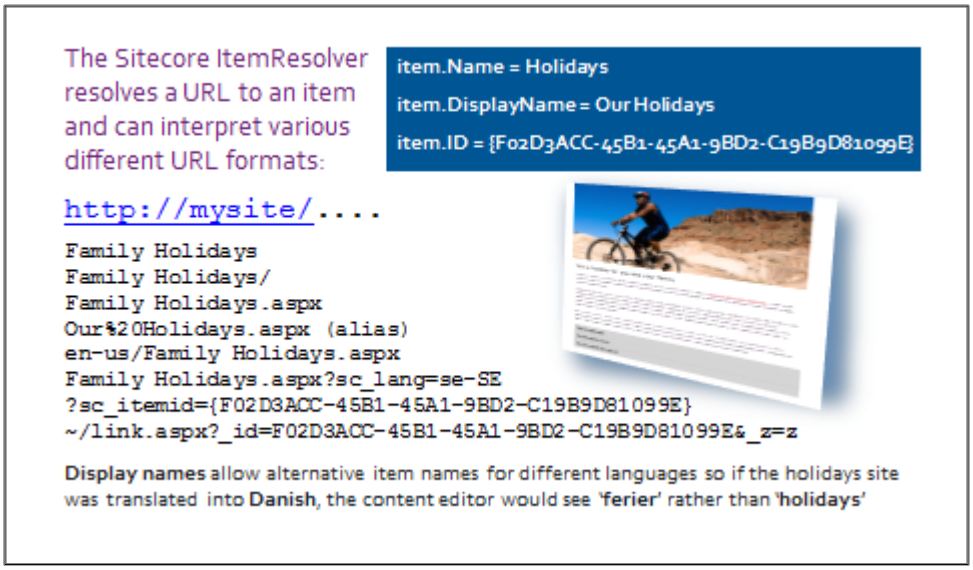
Objectives

By the end of this topic you will be able to:

- Explain what the ItemResolver does.
- Explain what the LinkManager does.
- Resolve a URL to an item.
- Resolve an item to a URL.
- Use URL options to output a URL in a context language.
- Customize LinkManager.

Content

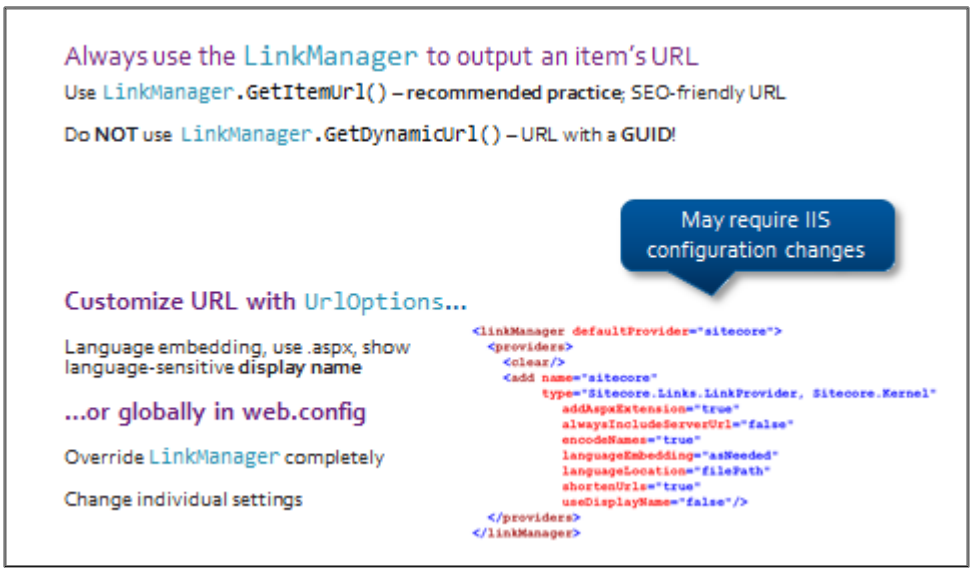
The ItemResolver



Knowledge Check

What issue do you see with duplicate URLs for the same item?

LinkManager and Item URLs



Demo – Retrieving an Item's URL

In the following demo, we will:

- Retrieve item URL using LinkManager.GetItemUrl()
- Use UrlOptions to output display name
- Change item **display name** in Sitecore

1. Using **Visual Studio**, open a sublayout, such as **Introduction.ascx**, and add a **Literal** control. Set the ID to *ItemUrl*.
2. Open that same sublayout's code file, for example, *Introduction.ascx.cs*.
3. In the **Page_Load** method, use Sitecore.Links.LinkManager.GetItemUrl() to **retrieve** the **URL** of the **context item** as a string, and set the .Text property of the **Literal** control to this value.

Code Sample – Retrieving Item URL

```
LinkManager.GetItemUrl(Sitecore.Context.Item);
```

4. In the same file, *before* retrieving the context item's URL, create a UrlOptions object and set the UseDisplayName property to true.
5. Pass the object into the Sitecore.Links.LinkManager.GetItemUrl() method along with the context item.

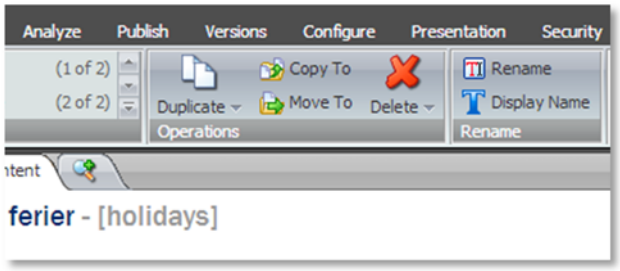
Code Sample – Retrieving Item URL

```
UrlOptions options = new UrlOptions();

options.UseDisplayName = true;
```

6. **Browse** to a page containing the sublayout that you edited. The URL of the **context item** is displayed as a string.
7. Using the **Sitecore Desktop**, open the **Content Editor** and locate the */sitecore/content/Home/Family Holidays item*.
8. Change to another language, for example, Danish.

9. Select the **Home** tab > **Display Name** command, and change the display name to *ferier.*



10. The content tree will refresh and display the Danish word for **holidays** in the tree as well as in the content pane. **Note** the original item name in square brackets next to the display name.
11. **Change** the **language back** and notice that the item display name changes again.
12. In your **browser,** change the **context language** to the language you previously set the alternative display name in. Use **?sc_lang=** query string at the end of the URL in the browser's address bar.
13. Press the ENTER key on the keyboard. The **Literal** control now displays a URL containing the word *ferier*.

Controlling Your URLs

Tip – Media Item URLS

Notice that we are instantiating a MediaItem object that accepts a regular Item. The MediaItem class is a wrapper that allows you to access properties that are only relevant to media items, such as its extension, or whether or not it's file-based.

```
string path = "/sitecore/media library/images/example";


Item item = Sitecore.Context.Database.GetItem(path);

MediaItem mediaItem = new MediaItem(item);

MediaUrlOptions options = new MediaUrlOptions();

options.MaxWidth = 200;

string url = MediaManager.GetMediaUrl(mediaItem, options);
```

Apply – Topic 4.2 – 10 min

Render Item Links

In the following lab, you will:

- Modify the **Subnavigation** sublayout that you created in the previous lab to output item links.

Lab A. Rendering Subnavigation Links

1. In **Visual Studio**, open the **Subnavigation** sublayout that you created in an earlier lab and open the **.ascx file.**
2. Within the repeater, use the Sitecore.Links.LinkManager.GetItemUrl() method to output each repeater item's URL as the Hyperlink control's **.NavigateURL** property.
3. Within the GetItemUrl() method, use an instance of UrlOptions to specify that the URLs should contain the item's **display names.**

Tip

You can do all of this on one line.

Create a new URLOptions object inside the GetItemURL method:

new Sitecore.Links.UrlOptions { PropertyName = x })

4. **Save** and solution and then browse to the http://BasicSitecore/Family Holidays page. Confirm that you can now use the **Subnavigation** list to navigate to trip details sub-pages.

Topic 4.3 Creating, Modifying, and Deleting Items

Introduction

Objectives

By the end of this topic you will be able to:

- Describe the implications of choosing the correct database to retrieve an item from.
- Choose the correct user to perform an operation on an item.
- Place an item into an editable state.
- Create an item programmatically.
- Update a simple item via its raw value.
- Make a modified item live on the web database.

Content

Fields and Raw Values

All items have a collection of fields

```
FieldCollection fields = Sitecore.Context.Item.Fields;
```

A fields raw value (database value) depends on the field type

Field type	Controls
Single-Line Text	`Welcome to the world of Sitecore!`
General Link	`<link text="Read the feature guide" linktype="internal" url="/Home/holidays.aspx" id="{54089BF1-790F-44E0-93E4-66BEC9CD61D1}" />`
List Field	`{743C2FAD-0E01-426E-8972-2CCFEBDC20F9}|{243C2FAD-0E01-C26E-8972-1CCFEBDC20F9}`

(More on raw values in Topic 4.4)

Getting and Setting Values

Access the field value directly...

```
string fieldValue = Sitecore.Context.Item["Profile Text"];
```

(returns `String.Empty` *even if field is null)*

....or retrieve the field as an object first.

```
Field field = Sitecore.Context.Item.Fields["Profile Text"];

string fieldValue = field.Value;
```

Simple text fields can be changed by setting their `.Value`

```
field.Value = "My new value!";
```

If you bind `field.Value` to a literal, it will *not* be editable in the Page Editor!

Demo – Outputting Raw Values

In the following lab, you will:

- View raw values of items in **Rocks**
- Output raw value to the browser – it will not be editable in the Page Editor

1. In Sitecore Rocks, double-click an item to open it. Select an item that has a populated **image** field, such as a child of */sitecore/content/Home/Family Holidays*.
2. Right-click in a grey area between fields and select **Raw Values**.
3. Notice that the value is custom XML, and not an image tag.
4. Using **Visual Studio**, open an existing sublayout's **code behind** (for example, *Introduction.ascx.cs*).
5. In the **Page_Load** method, retrieve the **Heading** field as a Field object:

Code Sample – Field object

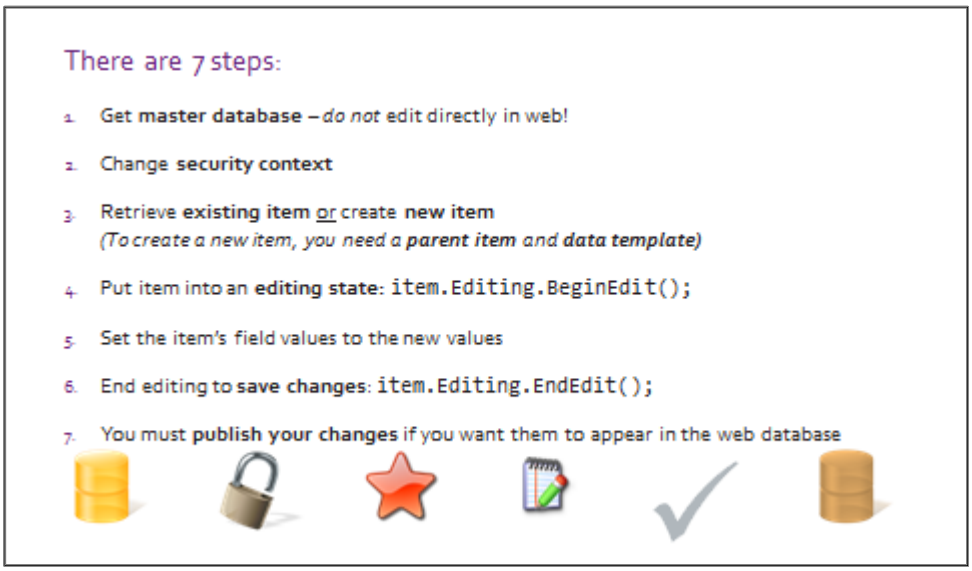
```
Field headingField = Sitecore.Context.Item.Fields["Heading"];
string headingValue = headingField.Value;
```

6. **Bind** the **value** of the **field to** a **Literal** control.
7. In a **browser, preview** a page that utilizes the sublayout, then **switch** to **Page Editor.**
   Note that the **value** is **not editable**. This is because we have output a **raw value** rather than a **rendered value.**

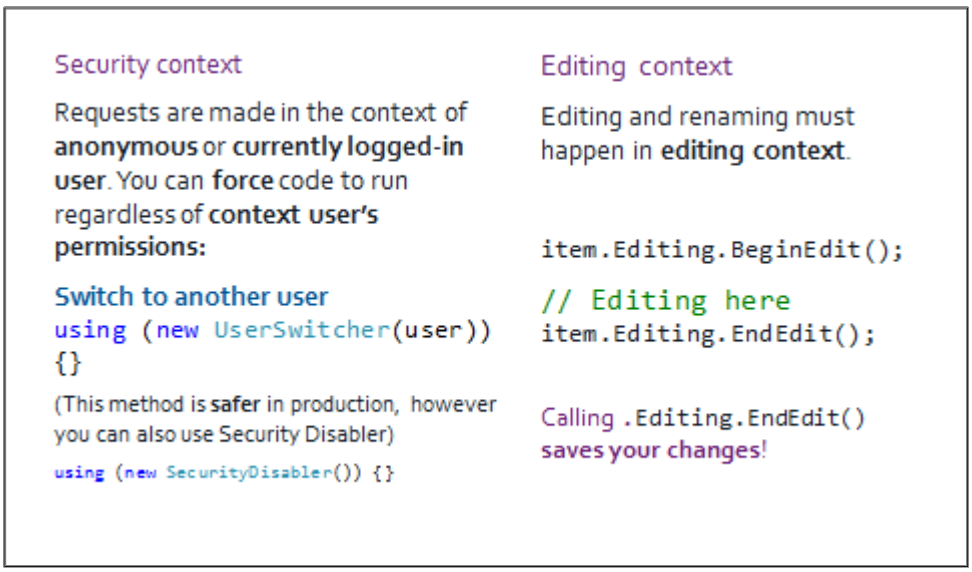
Knowledge Check

Why might it be difficult to set the value of an image or link field when you output raw values?

Creating and Editing Items



Security and Editing Contexts



Code Sample – UserSwitcher

```
if (User.Exists(user.Name)) {
   using (new UserSwitcher(user)) { }
}
```

Tip

Editing.EndEdit(); returns a *bool* that indicates whether your item was successfully saved. It also has some overloads that allow you to save in a silent manner, without raising any events. This can be useful when modifying an item as part of an event, so you do not end up creating an endless recursion.

Demo – Creating and editing an item

In the following demo, we will:

- Programmatically create an item based on the Trip Details template

1. In **Visual Studio**, locate an existing sublayout (for example, *Introduction.ascx*).
2. **Create** a **button** with an **OnClick** event.
3. In the **OnClick** event, create a new item based on the **Trip Details** data template beneath the */sitecore/content/Home/Family Holidays item*:

Code Sample – Creating and Editing an Item

```
using (new SecurityDisabler())
{
   // You should create new items in the master database
   Database master = Sitecore.Configuration.Factory.GetDatabase("master");
   Item holidayParent = master.GetItem("/sitecore/content/home/Family Holidays");
```

```
    // Get the ID of the data template, and check that item name is valid
    ID sitecoreID = new Sitecore.Data.ID("{565A8A17-032F-44AC-B506-B65F27A31241}");
    TemplateID templateID = new TemplateID(sitecoreID);

    string name = ItemUtil.ProposeValidItemName("Cycle the Cotswolds");
    // .Add will return the newly created item as an object
    Item newItem = holidayParent.Add(name, templateID);

    // Edit this new item's fields
    newItem.Editing.BeginEdit();
    Field heading = newItem.Fields["Heading"];
    heading.Value = "Our fantastic holidays";
    newItem.Editing.EndEdit();

    Response.Redirect(LinkManager.GetItemUrl(Sitecore.Context.Item), false);
}
```

4. **Save** the solution.
5. Using a browser, navigate to a page that contains the sublayout with the button that you created. Click the button and your code will execute.
6. Browse to the newly created item (*/Family Holidays/Cycle the Cotswolds*).

Recommended Practices

Do not hard code references to templates. Use a centralized class (for example, TemplateReferences.cs) with static properties for template GUID.

Knowledge Check

If you create an item in the master database, what needs to happen before it will be visible on the site?

Tip – Creating a TemplateID Object

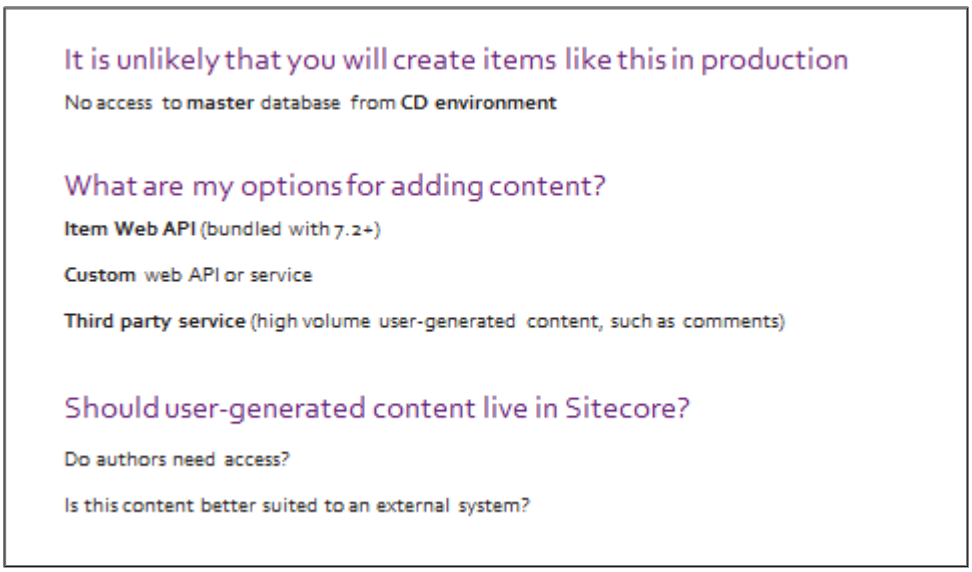
Sitecore defines an ID class that accepts a string or GUID:

```
ID sitecoreID = new Sitecore.Data.ID("{565A8A17-032F-44AC-B506-B65F27A31241}");
```

TemplateID accepts an ID object:

```
TemplateID templateID = new TemplateID(sitecoreID);
```

Recommended Practices



Tip – Deleting Items

You can delete an item using item.Recycle() or item.Delete(). Using item.Recycle() will move the item in Sitecore's **recycling bin** if the RecycleBinActive setting in the web.config is set to true. If this setting is set to true, item.Delete() will redirect to item.Recycle().

Tip – Publishing Items

When you create or delete an item programmatically in the **master** database, the changes will not appear in the **web** database unless you publish. Your options here are:

- Publish programmatically
- Wait for a scheduled publishing task to publish the entire tree
- If the item goes into a **workflow** (covered in Module 9), wait for it to be manually **reviewed and approved by a moderator**

Apply – Topic 4.3 – 30 min

Create and Edit Items

A common example of when you might want to create new Sitecore items programmatically is a **commenting system** (for example, allowing anonymous visitors to comment on particular trips they have been on). In the following labs, you will:

- Create a Comment data template.
- Create a Comments Submission sublayout with a form, and assign it to the Trip Details template standard values.Create a button OnClick action that creates a new item.

Lab A. Create Comment Data Template

1. Using Sitecore Rocks, create a **Comment** data template with the following fields:

Field section	
Comment	
Field name	**Field type**
Comment Author	Single-line text
Comment Text	Rich Text

2. **Assign** a template **icon** (for example, user1_message.png).
3. **Save** and **publish** the data template.

Knowledge Check

In this instance, you might not need to assign the Comment data template as an insert option on the Trip Details data template standard values. Why not?

Knowledge Check

What would happen if you browsed to: */holiday-1/comment-1?*

Lab B. Create a Comments Form Sublayout

1. Create a new sublayout called ***Comments Form*** (use **Add > New Item**... and select **Renderings > Sublayout** from the **Sitecore** menu).
2. Create standard .NET form inputs for **Comment Author** and **Comment Text**. There is sample code available in the **student resources folder** (WND Labs > *Module 4* > *Topic 4.3* > *Lab B* > *Code* > *CommentsForm.ascx).*
3. Assign the sublayout to the **maincontent** placeholder on the **Trip Details** data template **standard values.**
4. **Publish** Sitecore.
5. **Confirm** that the **form appears** on every instance of the **Trip Details** data template.

Lab C. Create Comment Items from Form Inputs

Finally, wire up the form so that submitting it creates a new item, beneath the context trip details item, based on the Comment data template.

1. Open **Comments Form.ascx.cs.** You will find a partially completed code sample with an **OnClick method (btnSubmit_Click)** in the **student resources folder** (WND Labs > *Module 4* > *Topic 4.3* > *Lab C* > *Code* > *CommentsForm.ascx.cs*).

   For the remaining steps in Lab C and Lab D, comments are available in CommentsForm.ascx.cs to guide you.

2. In the button's **OnClick** method, start by retrieving the **context item** from the **master database** (consider using Sitecore.Configuration.Factory.GetDatabase(). In this particular case, the item will be of type Trip Details.

Knowledge Check

Why are you getting the comment from the master database in this instance?

---

3. Having retrieved your intended comments **parent** (the context trip item), wrap your item creation code in an instance of Sitecore.SecurityModel.SecurityDisabler.

Knowledge Check

Is there another method that you could use instead of Sitecore.SecurityModel.SecurityDisabler? Why is it *not* best practice to use SecurityDisabler?

---

4. Use the .Add() method on the trip item that you retrieved from the master database to **insert a new comment item as a child**. This method will return an Item object.
5. Set the **template parameter** to the **Comments template** you created in **Lab A.**

Tip

*Create a Sitecore ID object from the template's GUID, and then use that to create a TemplateID object that the **.Add()** method can accept.*

6. Set the **name parameter** to the **current date/time** in **ISO format.** Use the Sitecore.DateUtil.IsoNow property.
7. **Save** the solution.
8. **Browse** to a trip page and **submit** the form.
9. Confirm that child items appear under the context **Trip Details** item. Unless you populated the data template's **standard values**, the comment items' fields will not contain any data.

Lab D. Populate Comments Item with Form Values

1. After using the **Add()** method to create a new item, put the resulting item in an **editing state** using the **.Editing.BeginEdit()** method.
2. Set the **Comment Author** and **Comment Text field values** to the corresponding **form input values**. At this point, you may want to use HttpUtility.HtmlEncode() to **sanitize** your input.

Code sample – Retrieving Field Value

```
item.Fields["Comment Author"].Value = CommentAuthorInput.Text;
```

Knowledge Check

Given that all field values are stored as strings, why can't you set the *Comment Date* field value to a DateTime.Now?

---

3. **Finish editing** by calling the **Editing.EndEdit()** method.
4. **Save** the solution.
5. **Browse** to a **trip** page, fill in and submit the form.
6. In **Sitecore Rocks**, confirm that the child items created under the context trip now contain your form content.

Extend

- **Information on programmatically publishing is in the API cookbook - section 2**
  http://sdn.sitecore.net/upload/sitecore6/64/content_api_cookbook-a4.pdf

Topic 4.4 Working with Complex Fields

Introduction

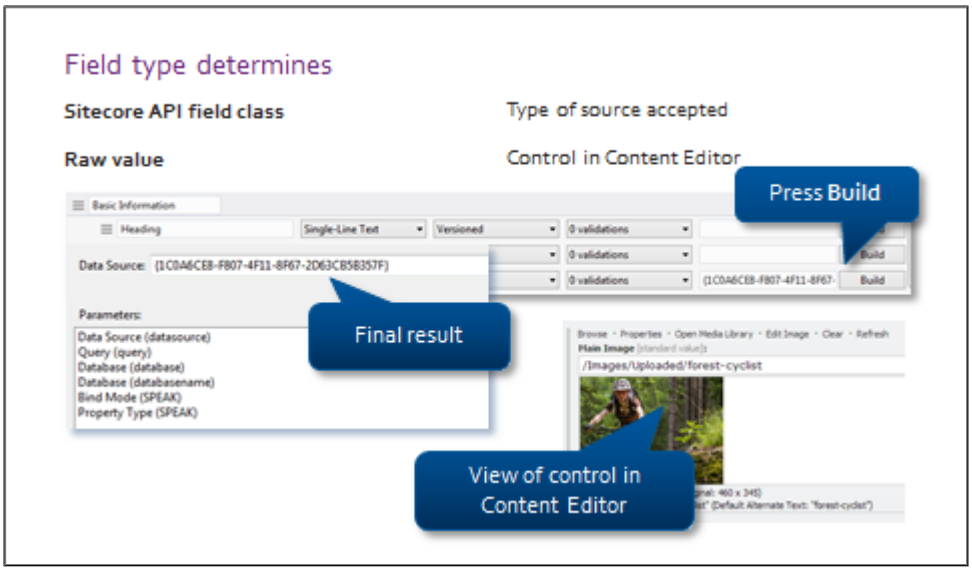
Objectives

By the end of this topic you will be able to:

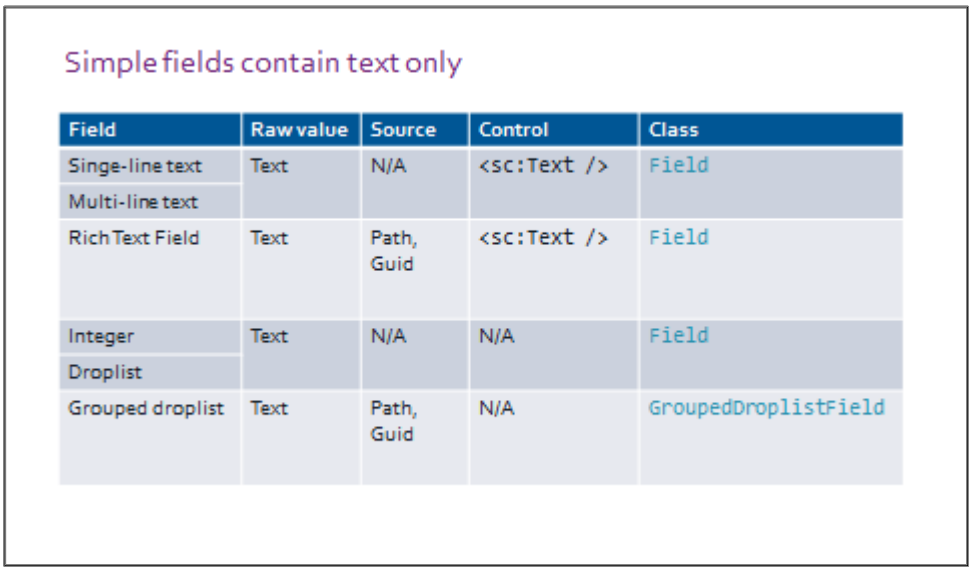
- Describe the correct way to retrieve complex field values.
- Get and set content in complex fields.
- Use ImageField, LinkField, ReferenceField and MultilistField.
- Render fields using the FieldRenderer.

Content

RECAP: Field Types



Simple Fields



Complex Fields

Field	Raw value	Source field	Control	Class
Image	Custom XML	Query, Path, GUID	<sc:Image />	ImageField
General Link	Custom XML	Query, Path, GUID	<sc:Link />	LinkField
Droplink	Single ID	Query, Path, GUID	N/A	MultilistField
Grouped Droplink		Query, Path, GUID		
Droptree		Query, Path, GUID		
Checklist	Pipe-delimited IDs	Query, Path, GUID		
Multilist		Query, Path, GUID		
Treelist		Source parameters		
Treelist-Ex		Source parameters		
Date	ISO date		<sc:Date />	DateField
Checkbox	True/False	N/A	N/A	CheckboxField
File	Custom XML	Query, Path, GUID	N/A	FieldField

Base Field Object

```
Sitecore.Data.Fields.Field
```

Example

```
Field headingField = item.Fields["Heading"];
string fieldValue = headingField.Value;
```

Raw Value (Rich Text Field example)

```
<p>Have an <a
href="~/link.aspx?_id=992233B83B0447F4B23BE548AACED6AA&am
p;_z=z">awesome holiday</a> with us!</p>
```

Notice _z=z in the url. This is a terminator, everything after this is treated as a normal querystring

Notice the URL – this uses non-SEO friendly GetDynamicUrl()

ImageField object

```
Sitecore.Data.Fields.ImageField
```

Example

```
ImageField imageField = item.Fields["Banner"];
MediaItem mediaItem = imageField.MediaItem;
string altText = imageField.Alt;
```

Raw value

```
<image mediapath="/BaseCore/Images/Holiday Image"
mediaid="{743C2FAD-0E01-426E-8972-2CCFEBDC20F9}"
src="~/media/743C2FAD0E01426E89722CCFEBDC20F9.ashx" />
```

Best Practice

Do *not* manually construct an <img.src="" /> tag from an ImageField object and render it to the page because the values will not be editable in Page Editor.

LinkField Object

`Sitecore.Data.Fields.LinkField`

Example

```
LinkField linkField = item.Fields["Search Engine"];
ID targetID = linkField.TargetID;
string linkText = linkField.Text;
string linkType = linkField.Type;
```

Raw value

```
<link text="Read more" linktype="internal"
url="/Home/holidays.aspx"
id="{54089BF1-790F-44E0-93E4-66BEC9CD61D1}" />
```

Best Practice

Do *not* manually construct an <a href="" /> tag from a LinkField object and render it to the page. The values will not be editable in Page Editor.

ReferenceField object

`Sitecore.Data.Fields.ReferenceField`

Example

```
ReferenceField refField = item.Fields["Featured Bike"];
Item targetItem = refField.TargetItem;
```

Raw Value

```
{54089BF1-790F-44E0-93E4-66BEC9CD61D1}
```

MultilistField Object

`Sitecore.Data.Fields.MultilistField`

Example

```
MultilistField listField = = item.Fields["Related Trips"];
IEnumerable<Item> itemList = listField.GetItems();

// Add or remove items
listField.Add(tripItem);
listField.Remove(tripItem);
```

Raw value

```
{54089BF1-790F-44E0-93E4-66BEC9CD61D1}|{14289BF1-29AF-4EE0-93E4-66BEC9CD62D4}
```

Some Fields Can Be 'Rendered'

Images, links, text, and dates can be output to the browser

Using the **FieldRenderer pipeline** in turns a raw value (unfit for public consumption):

```
<image mediaid="{743C2FAD-0E01-426E-8972-2CCFEBDC20F9}"
mediapath="/BaseCore/Images/Holiday Image"
src="~/media/743C2FAD0E01426E89722CCFEBDC20F9.ashx" />
```

...into this – **rendered field value**:

```
<img src="/~/media/BaseCore/Images/Holiday Image.ashx"  />
```

Lists, references, files, and checkboxes cannot be rendered.

FieldRenderer.Render()

You can use FieldRenderer.Render() to get rendered field values in code-behind

```
string literal = FieldRenderer.Render(Sitecore.Context.Item, "Image");
```

Sitecore controls use this method too!

Why should you use it?

Transforms field content into **valid HTML – Sitecore controls** ultimately go through this method

Automatically makes fields **editable in Page Editor**

Allows you to pass in **parameters** that match the ones available in Sitecore controls - e.g., image max width

Translates dynamic links in Rich Text Editor fields into SEO-friendly URLs

Parameters

The Render() method accepts a parameter query string – depending on the field being rendered, different parameters can be passed in

Takes an **item**, a **field**, and also a **parameters string** formatted as a query where **Parameters** depend on what's rendered e.g., **mw** (max pix width) for images:

```
FieldRenderer.Render(item, "Main Image", "mw=120; as=1");
```

(set the max width – mw – to 120px and allow stretching – as)

Remember! Hardcoded parameters override properties entered by author

Hard-coded parameters override any author-entered property i.e. a user has specified a 200 width in Sitecore, it will be **overridden** by the parameter string width

Image Parameters	
Parameter	**Property affected**
W	Width
H	Height
mw	Max Width
Mh	Max Height
La	Language
Vs	Version
Db	Database
Bc	Background Color
as	Allow Stretch
sc	Scale (floating point)

Recommended Practice

Always go through the Fieldrenderer method when you are rendering field content to the screen. This method allows:

- Your field to be **editable** in **Page Editor.**
- **Override properties** to be **specified by** the **author** (for example, the **Class** on a link or the **Width** on an **image**).
- Links in Rich Text fields to be transformed into SEO-friendly URLs.

Demo –Using FieldRenderer

In the following demo, we will:

- Use the FieldRenderer.Render() method with a few parameters to output an image field.

1. Using **Visual Studio,** open the **Introduction** sublayout's **code behind.**
2. In the **Page_Load** method, render the **Main Image** field and bind the resulting string to a **Literal** control.
3. Pass in a number of **parameters** that are relevant to Image fields (for example, max width and max height [**mw=200&mh=100**]).
4. Show that these parameters are represented as properties on an image control.

Apply – Topic 4.4 – 50 min

Populate and Render Complex Fields

In the following labs, you will:

- Add a General Link field to your Comment data template that accepts the comment author's website.
- Create a new sublayout that renders a list of comments at the bottom of pages based on the **Trip Details** template.

Lab A. Populate a Comment Link Field

1. Using **Sitecore Rocks,** locate the **Comment data template** and add the following additional field:

Field section	
Comment	
Field Name	Field
Comment Author Website	General Link

2. Using **Visual Studio,** open Comments Form.ascx and add a **textbox input** for a **link.** There is sample code for this **additional field** in the **student resource folder** *(WND Labs > Module 4 > Topic 4.4 > Lab A > Code > CommentsForm.ascx).* Do not overwrite the contents of your file; insert the sample.
3. Open Comment Form sublayout's **.cs file**. Retrieve the Comment Author Website field as a LinkField **object** where the comment item's fields are being populated (between .BeginEdit and .EndEdit).
4. We want the **rendered output** to look like this: http://www.sitecore.net/. Set the following **properties** on the LinkField object:

Object property	Object value
.URL	The contents of the link input (e.g. AuthorWebsite.Text)
.Text	The *same* contents of the link input (http://www.sitecore.net/)
.Target	"_blank"
.LinkType	"external"

5. **Save** and **publish** Sitecore.
6. Browse to a **trip details page,** fill in and submit the form.

7. Switch to **Sitecore Rocks** and locate the newly created comment. Confirm that the **link field** is populated with the form content.

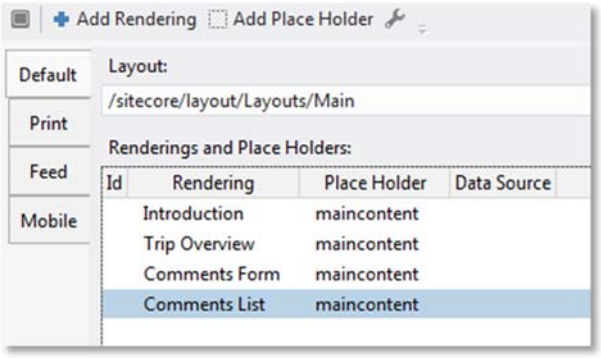
Lab B. Render List of Comments

In the following lab, you will:

- Bind a list of comments to a repeater and output the author's name, web address, and comment.

1. Create a new sublayout called *Comments List*. Paste sample HTML for the comments list in the student resources folder (WND Labs > Module 4 > Topic 4.4 > Lab B > HTML> campaign-page-comments.html).

   Remember to **append** this sample HTML to the content already in the **Comments List** sublayout. Do not overwrite the directives and include it at the top of the file.
2. Add the new sublayout to the **standard values** of the **Trip Details data template,** below the **Comments Form** sublayout:



3. **Save** and **publish** Sitecore.
4. **Browse** to a trip details page to confirm that the sample HTML appears.
5. **Replace** the dummy HTML with a repeater. A sample repeater is available in *(WND Labs > Module 4 > Topic 4.4 > Lab B > Code > CommentsList.ascx).*
6. Open the CommentsList.ascx **code behind.**
7. In the Page_Load method, retrieve the context item's **children** and **filter it** so that only items with the Comment data template are included.
8. **Bind** this list to the repeater.

Tip

Use .Where() to compare each item's .ID to the ID of the Comment template:
.Where(x => x.TemplateID == MyTemplateIDObject))

In the repeater, use **Sitecore controls** (for example, <sc:Text />) to output the following field values:

a. Comment Author (**Text**)
b. Comment Text (**Text**)
c. Comment Author Website (**Link**)
d. Date created (**Date** – __Created field – use Sitecore.FieldIDs.Created to retrieve field and considering experimenting with the Format attribute).

For each control, set the DataSource attribute to the repeater item's **ID property**. This forces the control to draw its content from that particular comment item as opposed to the context item.

Tip

Just as in Topic 4.1, we are using a strongly typed repeater. If you are using Sitecore controls in the repeater, set each control's DataSource property to the following:
DataSource="<%#: Item.ID %>"

Note the colon used above in the binding expressions: <%#: %>, by adding the colon the data being bound is automatically HTML-encoded.

Save the solution, then use a browser to preview a Trip details page. Use the form to add a few comments. The final comments list should look like this:

Knowledge Check

When you submit a comment, why is it only visible in Preview mode?

Module 5
Advanced Presentation Concepts

Contents:

- Reusable Components
- Layout Deltas

Topic 5.1 Reusable Content

Introduction

Objectives

By the end of this topic you will be able to:

- Define a datasource.
- Retrieve the datasource from a component.
- Describe the benefits of using parameters.
- Set and retrieve parameters.
- Describe what the ParseURLParameters utility does.

Content

Demo – What Is a Datasource?

In the following demo, we will:

- Using TrainCore, in Page Editor design mode.
- Change a General Widget's datasource.
- Add an additional General Widget to the page.
- Illustrate the reuse of components and content.

Business Use Case

What if you have a quotes component that displays the same quote on a number of different pages?

Without datasources, you must define those quote fields on **every single item** where you want to display this quote. This results in content duplication.

What if you have a number of quotes that you want to display on one page, for example, on the bottom-right and on the top-left?

Without datasources, you must define several fields with similar names so that you can target each field individually, as well as several similar components (for example, Quote Title1, Quote Title2, Quote Author1, Quote Author2 resulting in field duplication).

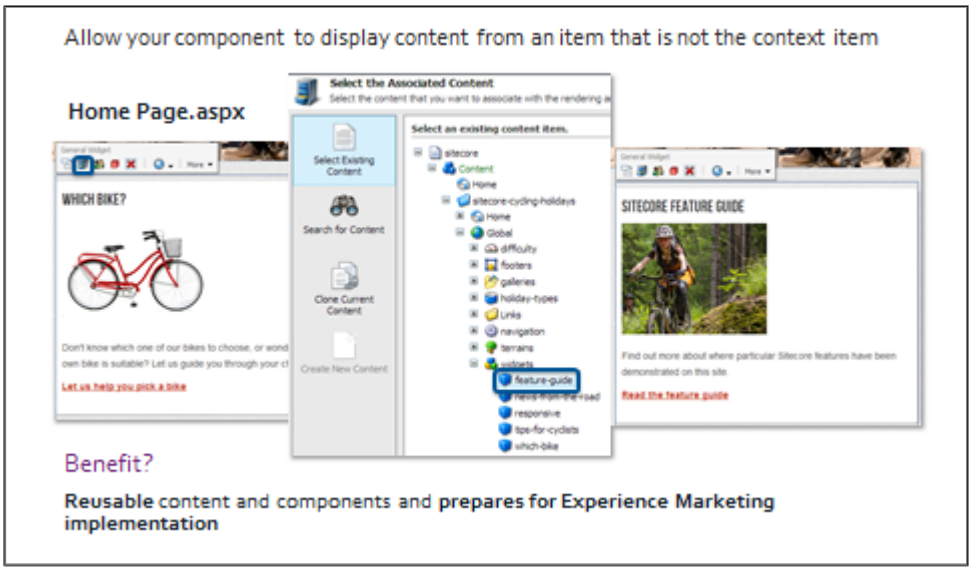
With datasources, you have one location in the tree with your quote items. Your quote component accepts a datasource of one or more of those items and displays them, resulting in *no duplication*.

1. Log in to the **Page Editor** interface in Traincore, and ensure that **design mode** is enabled.
2. Change the datasource of a **General Widget** component on the home page to another item under: */sitecore/content/sitecore-cycling-holidays/Global/reusable content.*
3. Navigate to the **which-bike** page. Add a General Widget to the left or right-hand column of the page.
4. Set the datasource to the same item you used on the home page.

Knowledge Check

What kind of content can be *global content*?

What Is a Datasource?



Datasources Allow…

Best Practice

It is best practice to use IDs instead of paths. If the item gets renamed or changes location, the ID will prevent a broken link. When a user selects a datasource in the UI, Sitecore inserts an ID by default.

Tip – XSLTs and Web Controls

In XSLT, the $sc_item variable represents the datasource. No additional configuration is required. If no datasource has been specified, $sc_item defaults to the context item. By contrast, $sc_currentitem always returns the context item.

Similarly, there is a base web control class named **Sitecore.Web.UI.WebControl** that provides the GetItem() method to retrieve the datasource item.

Using the API for Sublayout Datasources

Sublayouts are just .NET user controls, no prior method to retrieve datasource

To get a sublayout's datasource, add the following method to your code-behind:

```
var sublayout = Parent as Sublayout;

var datasource = sublayout.DataSource; // GUID as string

Item datasourceItem = Sitecore.Context.Database.GetItem(new
ID(datasource));
```

You are casting the control's parent as a Sublayout, and using the .DataSource property on that object

Knowledge Check

What If you need to get the datasource for several components – which you will – what could you do with this code?

Forcing Controls to Use the Datasource

By default the target of a Sitecore control is the context item

If you use an `<sc:Text />` without explicitly setting the item that it should target, it will try to render the specified field from the context item.

To force Sitecore controls to target the data source, you can either set the `.Item` property or the `.DataSource` property in code-behind:

```
<sc:Text Field="Widget Heading" ID="Heading" runat="server" />

WidgetHeading.Item = myItem

OR

WidgetHeading.DataSource = "{6C0EB349-5FBA-481C-A350-7B9ECAD7D394}"
```

Or set Sitecore control's DataSource property in .ascx file!

Demo – Datasource API

In the following demo, we will:

- Modify the **Trip Details** component to accept a datasource.

1. Open http://BasicSitecore, and log into the Page Editor interface. Locate a page that displays the **Trip Overview** component. Change the datasource of the component to **another trip**.
2. **Save**, and then switch to Visual Studio.
3. In the code-behind for the **Trip Overview** component, wire up the two **Sitecore controls** to read their content from a datasource. As a bonus, if no datasource exists, fall back to the **context item**.
   - Give each Sitecore control an **ID**.
   - In the code-behind, retrieve this.Parent (the user control's parent) and cast it as a Sitecore.Web.UI.WebControls.Sublayout object. This object **wraps the user control**, and is Sitecore-specific.
   - Set .Item or .DataSource of each Sitecore control to the datasource provided by the Sublayout object.

Tip

You can then check if the component has a datasource. If it does, then display it, otherwise return the context item.

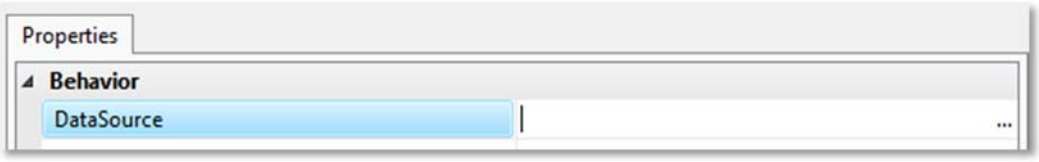
Tip

What if your component always had to accept a datasource? You can hide your controls by checking what mode the page is using in Sitecore.Context.PageMode then performing your logic.

4. **Save,** and switch to the **Page Editor.**
5. Verify that the Sitecore controls are outputting content from the **chosen datasource**, not the context item. Change the datasource again to confirm.

Tip

You can use **Sitecore Rocks** to set the datasource of a component. Select the content item whose presentation details you wish to edit, and press **CTRL+U**. Double-click the component whose datasource you wish to change, and click the **ellipsis** to choose a content item.

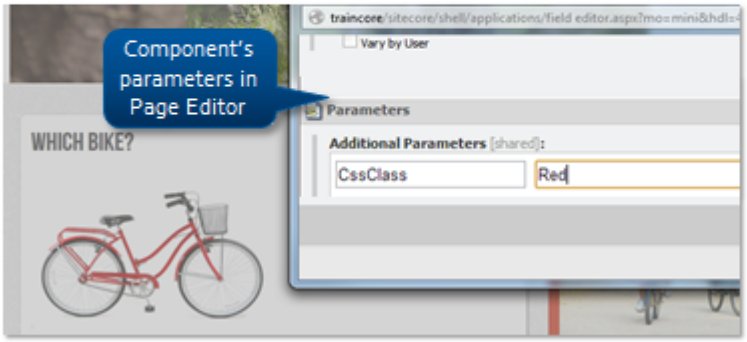


Component Rendering Parameters

Rendering parameters allow properties to be set **per instance** of a component

Rendering parameters stored as **Key Value Pairs** – specified as plain text

Value same format as Sitecore **raw value** – e.g., reference multiple items - use pipe-delimited GUIDs



Retrieve Rendering Parameters Using the API

Rendering parameters are stored as strings, to retrieve parameters in code for...

Sublayouts
Cast parent as a Sublayout and retrieve using the `.Parameters` property

```
Sublayout sublayout = this.Parent;
string parameters = sublayout.Parameters;
```

Web Controls
Use the `.Parameters` property

Sitecore has a utility for turning parameters into a NameValueCollection:

```
Sitecore.Web.WebUtil.ParseUrlParameters(this.Parameters)
```

Retrieve values as usual from a NameValueCollection i.e.

```
parameters["myindex"];
```

Demo – Parameters

In the following demo, we will:

- Change the background color for the **Trip Overview** component from grey to red using rendering parameters.

1. Open http://BasicSitecore in the **Page Editor,** and locate a page that displays the **Trip Overview** component. Ensure that designing mode is enabled.
2. In Visual Studio's **Solution Explorer,** open **Trip Overview.ascx**. Change the color for the **Trip Overview** table from grey to ***red*** by modifying the HTML to include a ***red*** class. In order for an author to be able to do this, this value needs to become a rendering parameter. Save and reload the browse to see the change in color.
3. Switch back to the **Page Editor,** and click the **Trip Overview** component to select it.
6. In the floating toolbar, select **More > Edit Page Editor Options**.
7. In the **Parameters** field at the bottom of the dialog, add ***CssClass*** as a key, and ***red*** as a value. Click **OK,** and **save** the page.

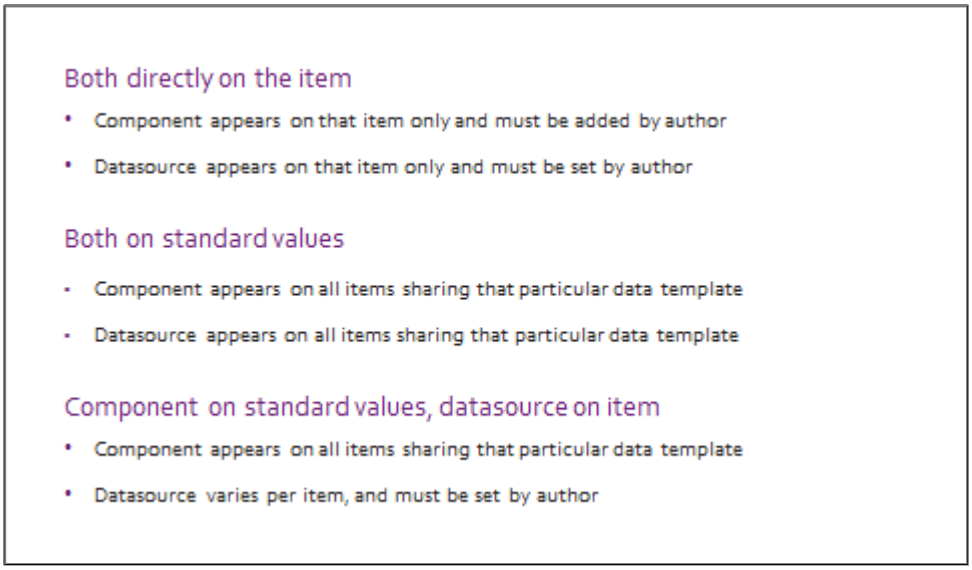
8. Switch to Visual Studio's **Solution Explorer**, and open **Trip Overview.ascx.cs**.
9. In this file, retrieve the component's rendering parameters as a string (use ((Sublayout)Parent).Parameters).
10. Convert the string to a NameValueCollection object using the Sitecore.Web.WebUtil.ParseUrlParameters() method.
11. Extract the value of the **CssClass** key that you specified earlier, and set it to a **public property**.
12. Output that public property in **Trip Overview.ascx** in the following location:

Code sample

```
<div class='indentedSection <%= CssClass %>'>
```

13. **Save**.
14. Switch to the **Page Editor** and reload the page. Confirm that **red** is being output in the HTML source, and that the Trip Overview now has a red background.

Where Should I Define Presentation and Datasources?



Apply – Topic 5.1 – 40 min

Build a Featured Trip Component

In these labs you will build a reusable **Featured Trip** component that accepts a trip details datasource and can be styled using parameters.

Lab A. Create Component and get the DataSource Item

1. Create a sublayout named **Featured Trip**.

2. Paste the following sample HTML into the .ascx file. Alternatively, copy from **Student Resources Folder > WND Labs > Module 5 > Topic 5.1 > featured-trip.html**:

Code Sample

```
<div class="indentedSection widget fill red">
        <div class="indentedSectionInner">
                <h2>Explore Holland</h2> <!-- Heading field -->
                <img src="img/forest-cyclist.png" style="width:100%" /> <!-- Main Image field -->
                <p>An excellent tour of the country.</p> <!-- Main Content field -->
</div>
</div>          <p>An excellent tour of the country.</p> <!-- Main Content field -->

        </div>

</div>
```

3. Where indicated by comments in the HTML, use **Sitecore controls** to **output** the following **fields**:
   - Heading
   - Main Image
   - Main Content

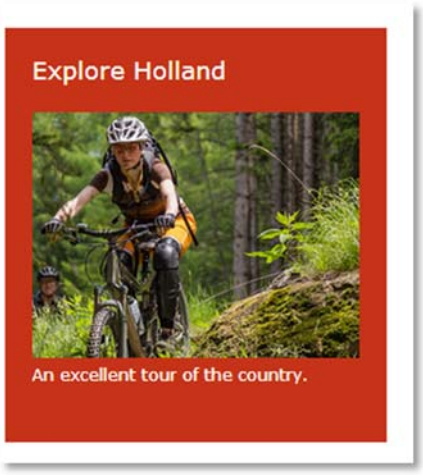
4. Give each control an **ID.**
5. Set the **Main Image** max width to **280**. **MaxWidth** is a property on the <sc:Image /> control.
6. **Save** the solution.
7. Using **Sitecore Rocks,** add the **Featured Trip** component to the **maincontent** placeholder on the **Family Holidays** item, (in this case not in the standard values – you might not want the component to appear every time you create a holiday).
8. Select the **sublayout** and open the **Properties** window. You can also open the **Properties** window by pressing F4, or double-clicking the component.
9. Set the **DataSource** property to any trip details item.
10. Save and preview the **Family Holidays** item, using Commandy.
11. **Confirm** that the **component** is **outputting data** from the **context item**. We have to explicitly tell the controls to output data from the datasource.

12. In the **Page_Load** method of the featured **Featured Trip.ascx.cs** file, get the component's datasource using the following code:

Code Sample – Datasource

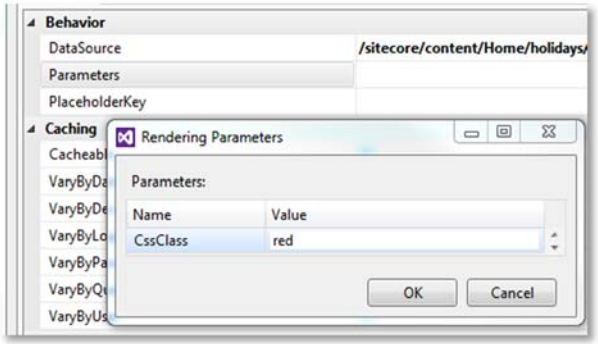
```
var sublayout = Parent as Sublayout;
var datasource = sublayout.DataSource;
```

13. In the **Featured Trip.ascx.cs** file, set the **.DataSource** property of each control to the **datasource item.**
14. Click the **Save** button to publish the solution.
15. **Preview** the page and confirm that the **content** is now **coming** from the **component's datasource** item rather than the context item.



Lab B. Retrieve the CssClass Parameter Value

1. Using Sitecore Rocks, navigate to the Family Holidays page where you added the **Featured Trip** component.
2. In the **Design Layout** pane, double-click the **Featured Trip** component to open its **Properties.**
3. Click the **Ellipses (. . .)** button next to the **Parameters** property.
4. Add a property called *CssClass* and give it a value of *red*. You will be injecting this value in the **Featured Trip** component's HTML.



5. In **Featured Trip.ascx.cs**, get the component **parameters** as a **string** (use ((Sublayout)Parent).Parameters).
6. **Convert** the **string** to a NameValueCollection object using the Sitecore.Web.WebUtil.ParseUrlParameters() method.

7. **Extract** the **value** of the **CssClass** key that you specified earlier and set it to a **public property** within your class:

Code Sample – Public Property

```
public string CssClass
{
  get;
  set;
}
```

8. Output that public property in **Featured Trip.ascx** in the following location:

Code Sample – Output Public Property

```
div class='indentedSection <%= CssClass %>'>
```

9. **Save** the solution.
10. **Browse** to the **Family Holidays** page and confirm that the indentedSection has an additional class that is being pulled from your component's parameters.

Topic 5.2 Layout Deltas

Introduction

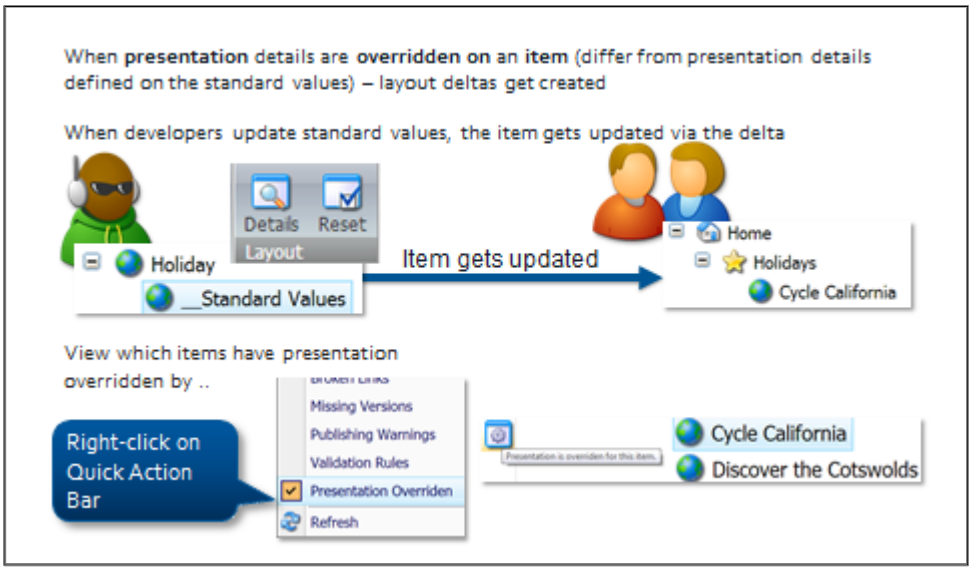
Objectives

By the end of this topic you will be able to:

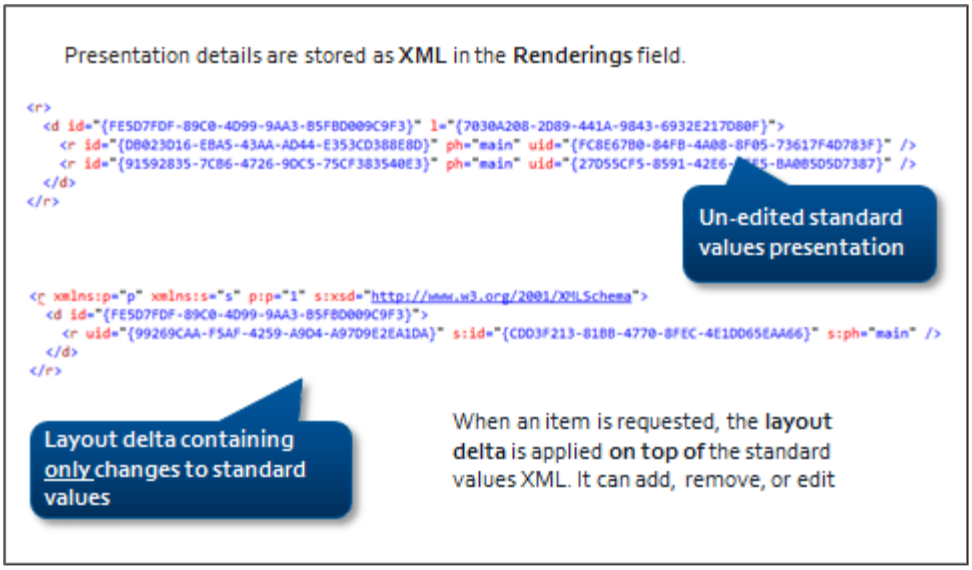
- Define the purpose of a layout delta.

Content

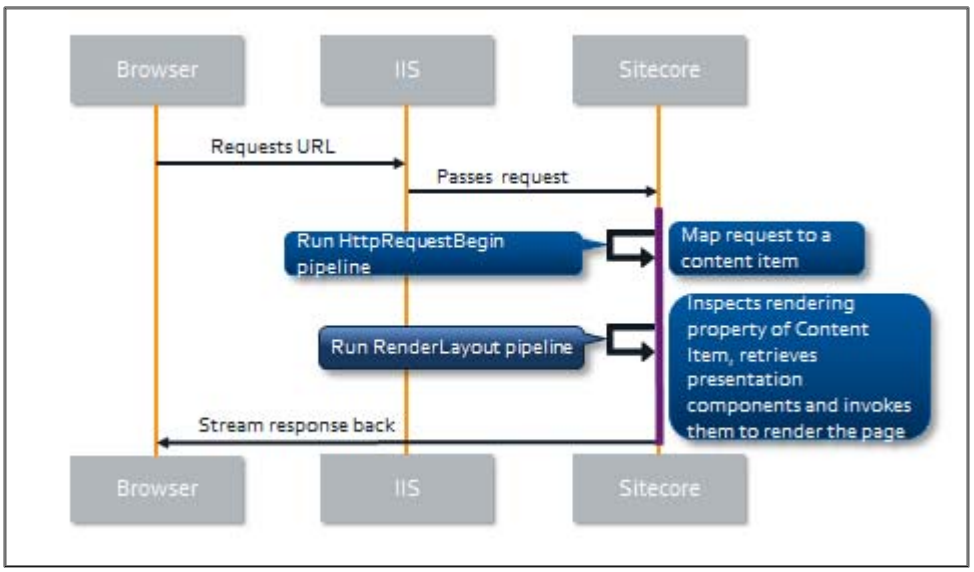
What Are Layout Deltas?



A Layout Delta Is Like a Transform



The RenderLayout in the Request



Module 6
Real World Solutions

Contents:

- Familiar Concepts
- Dealing with Larger Sites
- Sitecore Query

Topic 6.1 Familiar Concepts

Introduction

Objectives

By the end of this topic you will be able to:

- Describe a realistic Sitecore project setup.
- Describe a realistic content tree.
- Describe realistic data template and presentation configurations.
- Create and configure a new device.

Content

Demo – Solution, Topology, and Templates

In the following demo, we will look at:

- Traincore's project structure.
- Traincore's Sitecore tree structure.

Visual Studio Projects

The Traincore project folder sits **outside the web root** (more on the pros and cons of working inside the web root in **Module 9)**.

- The core of the application lives in **Training** (Training.csproj).
- Like all ASP.NET projects, common or shared functionality has been abstracted into three additional projects:
  - Training.Utilities
  - Training.Controls
  - Sitecore.Utilities (can be re-used with any Sitecore project)
- **Visual Studio publishing profiles** are used to move files from the project into the Sitecore web root. Right-click in the toolbar and select **Web One Click Publish** to enable the publish profile dropdown.

References

Referenced .dlls are stored in a **Libraries** folder in the project root. **Note** that **Copy Local** is set to false, in case you install Traincore into a different version of Sitecore than the referenced .dll. You could set up a local **NuGet** server within your company and store different versions of the Sitecore .dlls there.

Configuration

In the Visual Studio project, there are a number of smaller configuration files inside **/App_Config/Include** – such as *BaseCore.Sites.config* or *BaseCore.Search.config*. These **patch changes into the web.config,** so you can avoid keeping track of changes to web.config, which is already huge.

Browse to **http://traincore/*sitecore/admin/ShowConfig.aspx***. This page allows you to see the final patched web.config. (More on this subject in Module 9.)

Sitecore Content

Log into the **Content Editor** and expand Traincore's main site tree*: /sitecore/content/sitecore-cycling-holidays.*

- There are **two sites** in this installation:
- Sitecore-cycling-holidays
- Sitecore-cycling-holidays-corporate

We will cover multi-site implementations in a later topic.

- **Home** is the site root. Traincore has several other non-website folders including **Global** under: *sitecore-cycling-holidays*.
- Global contains small chunks of reusable content that are used as datasources by components, such as gallery slides and '*General Widget*' content (under: */Global/resusable content*).
- Global also contains the **entire site's navigation** as a tree structure under: /Global/navigation.
- **Bookings** is a **bucket** of unstructured holiday booking items. (More in this in Module 8). This is a sample site, and it is unlikely that you would build your own e-commerce system in Sitecore so consider products such as **uCommerce for Sitecore**.
- The **Settings** item under: */sitecore/Content* is for content that is **shared across all sites** such as **rendering parameter options** (more on this in Module 7).Most of this content uses **shared** fields, which means the content is identical across versions and languages.
- Traincore's **templates** inherit multiple, smaller base templates – there isn't a single 'base'. This is because certain pages, such as the home page, do not require a 'Heading' or 'Main Content' field. We recommend that you avoid redundant fields that don't output anything.

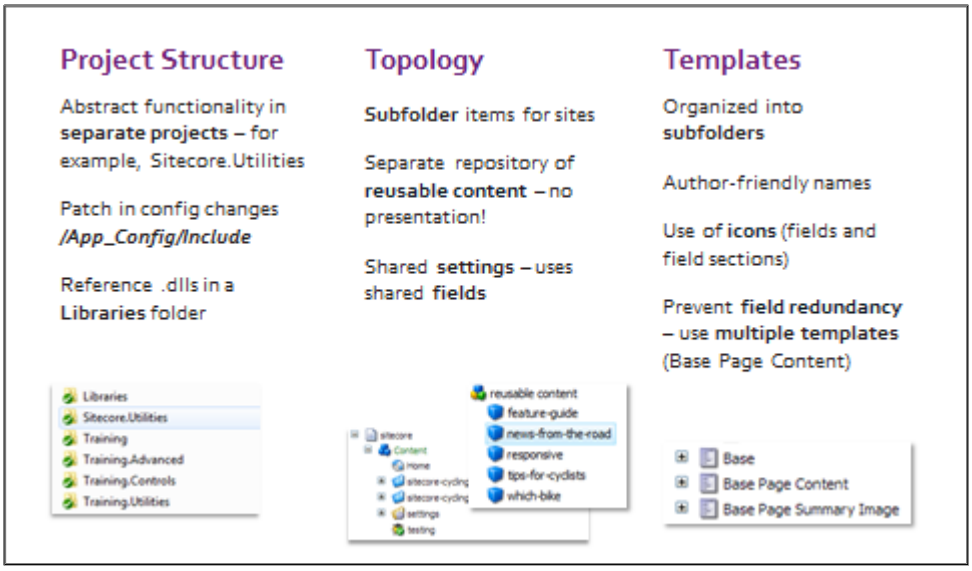
Tip

Traincore defines main and footer navigation in a separate folder under: /Global rather than generating it from the site structure. This is a much more inexpensive query; it allows you to include external link items and makes it easy to see at a glance what the site's navigation is going to look like.

Recommended Practices

All item fields should have an effect. An author should not fill in and publish a field without seeing a result. This makes the site appear broken.

Recommended Practices



Draw It

The training site has a lot of nested components, both statically and dynamically defined:

- **Containers** with placeholders that divide pages into horizontal slices.
- **Columns** including one-column, two-column, and three-column functional components.

Demo – Presentation Configuration

In the following demo, we will look at:

- **Presentation details** of various items and particularly how sublayouts are nested.
- Which components accept **datasources**.
- Which components make use of **rendering parameters**.

1. Using Visual Studio, expand the **layouts** folder. **Note** the number and variety of sublayouts.
2. Open */layouts/BaseCore/containers/* and */layouts/BaseCore/columns*. There are many more **placeholders** in this solution than the BasicSitecore solution from Modules 1 – 5. This is because Traincore makes use of **sublayout nesting** to build up a variety of page designs.

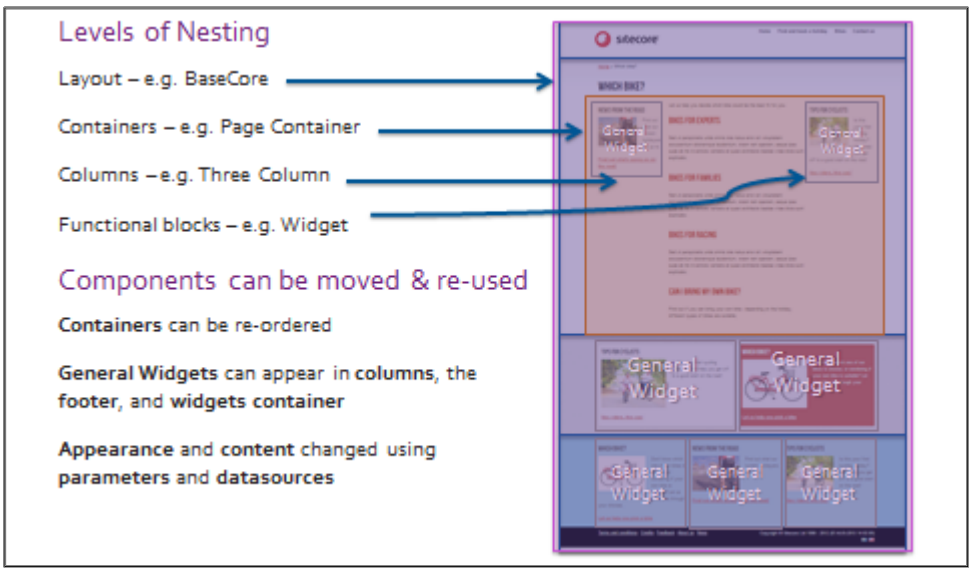
Important

You can only use one instance of a placeholder on a page. For example, if you have used the Gallery Container (which defines the **GalleryContainer** key), you cannot add an additional Gallery Container because you cannot target its placeholder a second time.

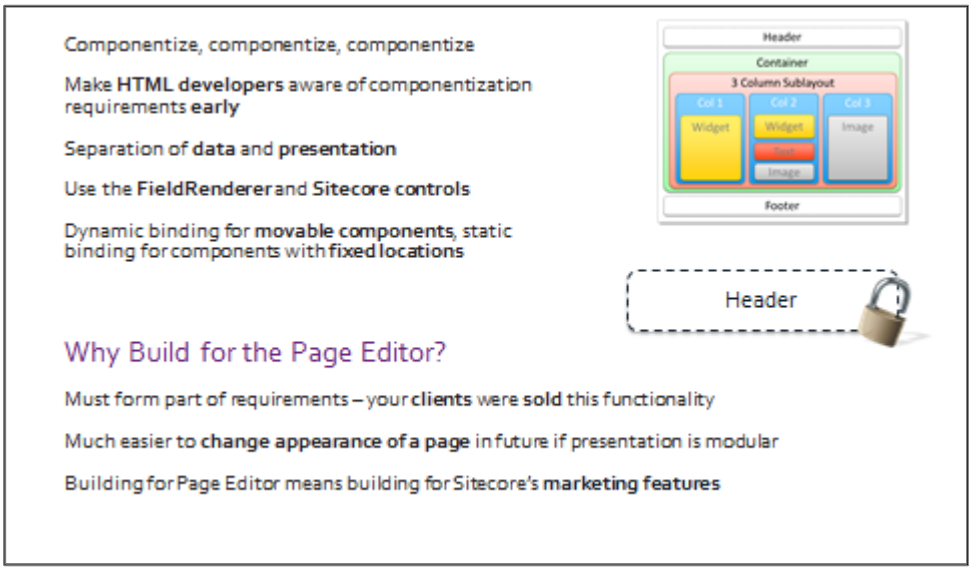
3. Log into http://traincore/, and select the **Page Editor** interface.
4. Directly under Home, create a **new page** of the type **Standard Content** and name it *Favorite Bikes*.
5. Ensure that **design mode** is enabled on the **View** tab.
6. The page is divided into **containers**. By default, Standard Content items have a **Header**, **Page**, and **Widgets** container. Insert a new **Gallery Container** component.
7. Into the **gallerycontainer** placeholder, insert a **Banner** component – choose a datasource when prompted. **Note** that the selection tree starts under the **galleries** item, and that incompatible items are greyed out. More on this in Module 7.
8. Select the **Widgets Container** component, and move it above the Page and Gallery containers. Splitting a page into containers allows for design flexibility.
9. Insert a new **General Widget** into the **widgetcontainer** placeholder and select a datasource for it.
10. **Click More > Edit Component Properties**, and change some of the values, such as **Class** and **Widget Width**. Using rendering parameters allows you to apply many different styles to the same component.

Although the **Standard Content** data template has a default presentation, we can completely change the look and feel using a **hierarchy of nested components**.

Placeholder Nesting and Component Re-use



Supporting the Page Editor



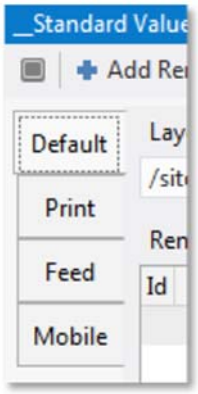
Demo – Create a Mobile Site

In the following demo, we will:

- Create a **mobile device**
- Create and assign a simple **Mobile layout** to a holiday item's Mobile device
- Output Page Heading and Page Content in the layout
- Add a number of existing components (also used by the Default device) to the Mobile layout's placeholder.

1. Using **Sitecore Explorer**, open **Traincore** and create a new device named **Mobile** under: */Traincore/master/sitecore/Layout/Devices*.

2. View the presentation details of any item and note that **Mobile** now appears as a device option in the Device Manager.



3. Using **Solution Explorer**, create a new **layout** called *Mobile.* You will be prompted to create a layout definition item in Sitecore.
4. Use Sitecore controls to output the **Page Heading** and **Page Content** fields inside the .aspx. In this example, we will not be using any styles.
5. Add a placeholder control to the bottom of the page and name its key *widgets*.
6. **Save** and **deploy** the solution.
7. Assign the new layout to the **Battle of the Hills** item (under: */sitecore/content/Home/sitecore-cycling-holidays/holidays/battle-of-the-hills*) on the **Mobile** device. Add a number of widgets to the **Widgets** placeholder.
8. Open the **Battle of the Hills** item in **Page Editor**. Use the **Experience** tab to change the device between **Default** and **Mobile**. **Note** that the content being displayed is the same as the content used by the Default device.

Recommended Practices

There are three main approaches to Sitecore and mobile:

Standard Responsive Design

Use the **responsive design** by itself, without using any Sitecore devices at all. Sitecore will not interfere. This means little extra work for back-end developers, and gives the responsibility to the front-end developers.

However, ignoring devices means that you cannot personalize content based on device, and large pages with many components may not look as good. You will also need to test all combinations of components that an author might use.

Devices

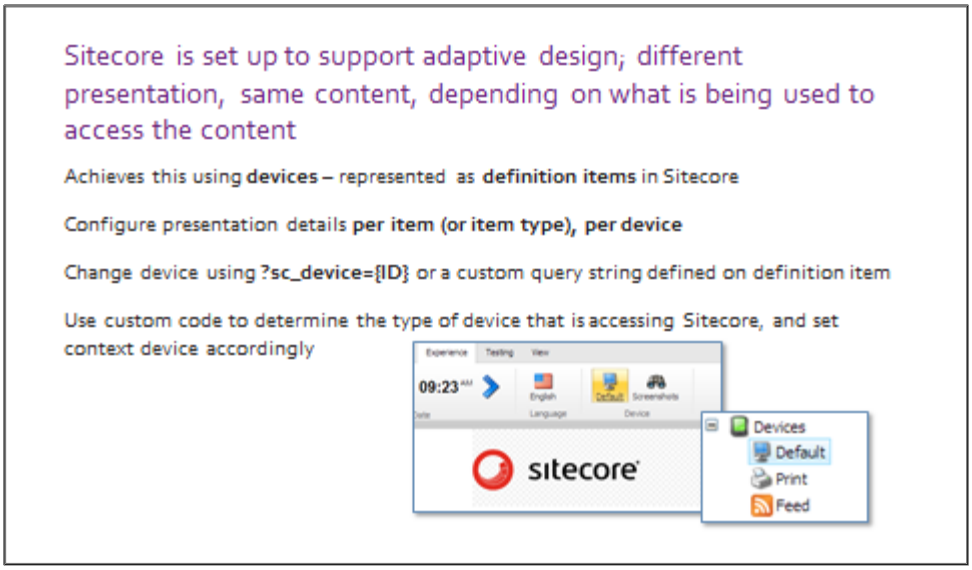
You can *either* create an **entirely separate** set of presentation details for your mobile device *or* use the **same presentation details** in a **different configuration,** and rely on responsive design for formatting. The first approach allows you to use the same content, but completely different components or CSS to style shared components. While this allows you to target mobile devices specifically, you may find that you need to maintain two sets of components and/or stylesheets.

The benefits of the second approach are that while you are relying on responsive design to format your site based on screen size, you can choose to use more or less components, and use them in a slightly different order. Responsive design is still responsible for the appearance of your site, but you can cut out excessive components to improve the experience.

Separate Site

Finally, if it suits your requirements, you can create an entirely separate mobile site (much like sitecore-cycling-holidays-corporate) with separate content.

Devices

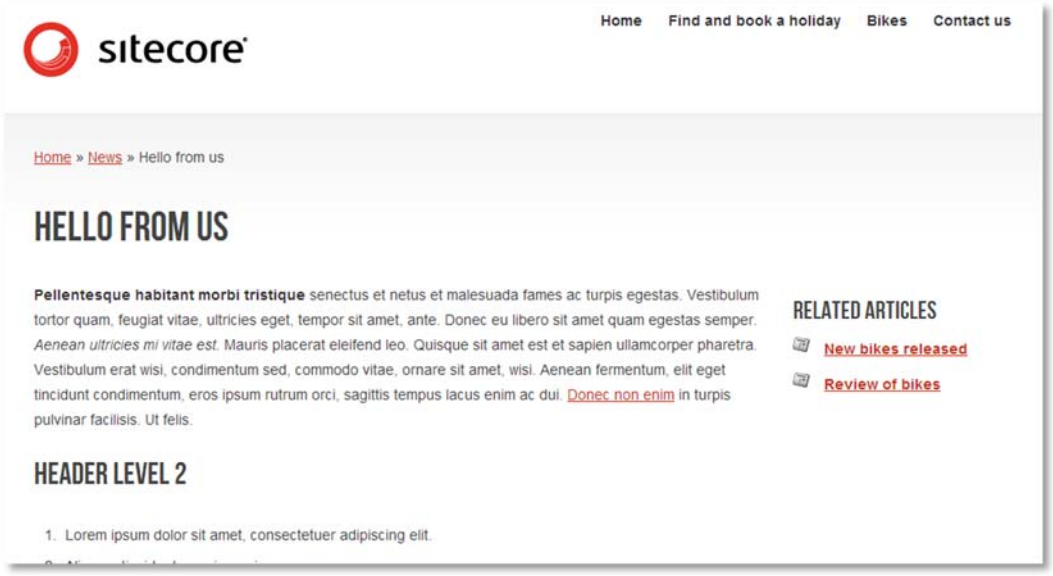


Apply – Topic 6.1 – 60 min

In this set of labs, you will create a news article page from a design and specification supplied by Sitecore Cycling Holidays.

- Sitecore Cycling Holidays is building a news section and have asked you to work on the **news article** template. News items are going to be created under the news listing: http://traincore/news.
- The current news section uses standard content pages: http://traincore/news/our-plans-for-the-new-year.
- The news article page will look similar to the standard content page, but with a news-specific feature – **related articles.**

News Article Page



Create a News Article Data Template and Standard Values

In the following labs you will:

- Create a **News Article** data template, configure its base templates and add a number of additional fields.
- Create the **News Article** template's standard values.
- Configure **News Article** as an insert option on **News Listing.**
- Create and populate a number of items based on the **News Article** template.

Lab A. Create a News Article data template

1. Using Sitecore Rocks, create a new data template called ***News Article*** under: */sitecore/templates/BaseCore/Pages.*

Knowledge Check

A news article page requires common fields such as **Page Heading** and **Meta Description**. Given that this is a functioning site and these fields have already been defined in a base template, how would you add them to the **News Article** template you have just created?

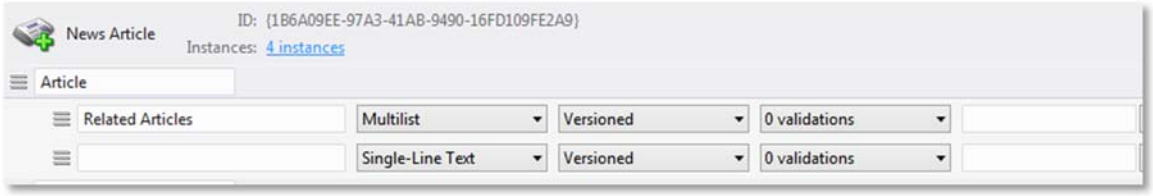
2. Select the following base templates: **Base Page, Base Page Content,** and **Base Page Summary Image.**

Tip

Rather than finding commands using menus, you can right-click and select the **Commandy** option, or press the **CTRL=SHIFT=SPACE** keys on the keyboard to bring up the Commandy toolbar. Commandy allows you to search for your command by name (for example, Browse or Page Editor) and execute it in the context you are in (for example, with a particular item selected).

3. Add an additional field under a field section called ***Article***:

Field Name	Field Type
Article	
Related Articles	Multi-list



Knowledge Check

Why should you inherit from base templates rather than repeat fields on individual templates?

---

4. Set the **source** of the **Related Articles** field to the path of the **News Listing content item** in the tree (*/sitecore/content/sitecore-cycling-holidays/Home/news*). This allows us to choose from the news item's children in the Related Articles multi-list.
5. Set the template icon to the **News Add** icon (search for or type *Network/16x16/news_add.png*).

Lab B. Create the News Article standard values, assign insert options, and create an article

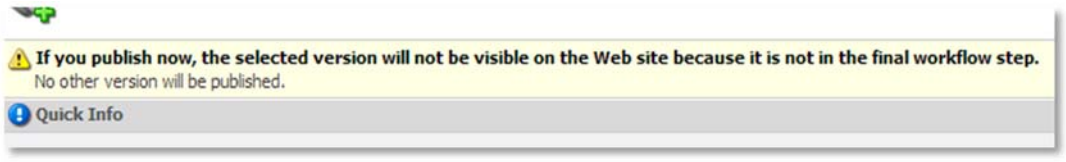
1. Create the News Article template's standard values.
2. On the template's standard values, check that the *$name* token appears in **Page Title, Page Heading,** and **Navigation Title**.
3. Navigate to the **News Listing** template's standard values and set *News Article* as an insert option.
4. Using Sitecore Rocks, create several new items based on the **News Article** template under: */sitecore/content/sitecore-cycling-holidays/home/news.*

Tip

Use the 'create multiple items' functionality in Sitecore Rocks. In the bottom left-hand corner of the ***Add New Item*** *dialog, increase the number of items to create from 1 to 5 or more:*

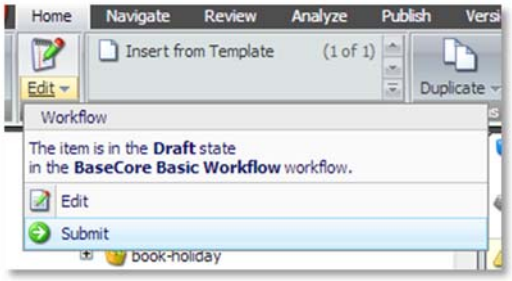
5. Populate each item with sample content. For each article item, select two or more **related articles** in the **Related Articles Multi-list.**

6. The following will appear at the top of each news article item:



In Sitecore, workflow is a highly configurable approval process that items go through before they appear on the live site (for example, to make sure that news articles that have not been approved by the news team do not appear on the site).

7. To approve the news article, click the **Home** tab, then click the arrow next to the **Edit** button > Click the **Submit** option as detailed below. We will cover workflow in more detail in Module 9.



8. **Smart publish** the database and then **browse** to the **news** page to make sure that your new items are being displayed in the **news list.**

Knowledge Check

Click on the link to your news item. You should see an empty page. Why can't you see the '*The layout for the requested document was not found*'?

Create a Component and Bind it to the Placeholder

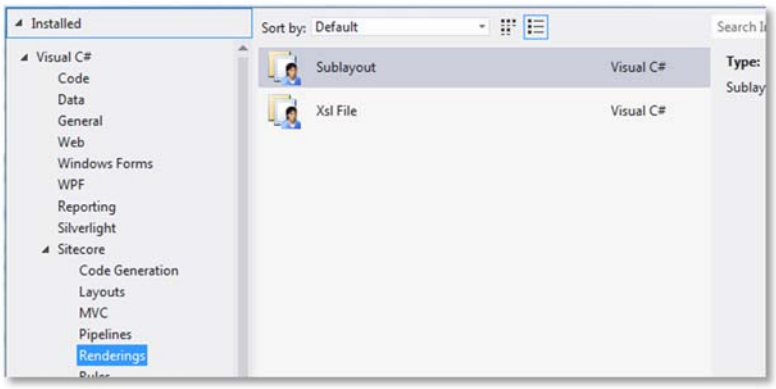
In the following labs you will:

- Add the **Two-Column** sublayout to the **News Article's** standard values.
- Create a sublayout to output the contents of the **Related Articles** field.
- Bind your sublayout to a placeholder.

Lab A. Create a Related Articles Sublayout

1. Using Visual Studio **Solution Explorer,** right-click the **/layouts/BaseCore/content/** folder and select the **Add > New Item…** option. Insert a new sublayout called *Related Articles.*

2. Use the Sitecore section to **insert** a sublayout and name it *Related Articles.*

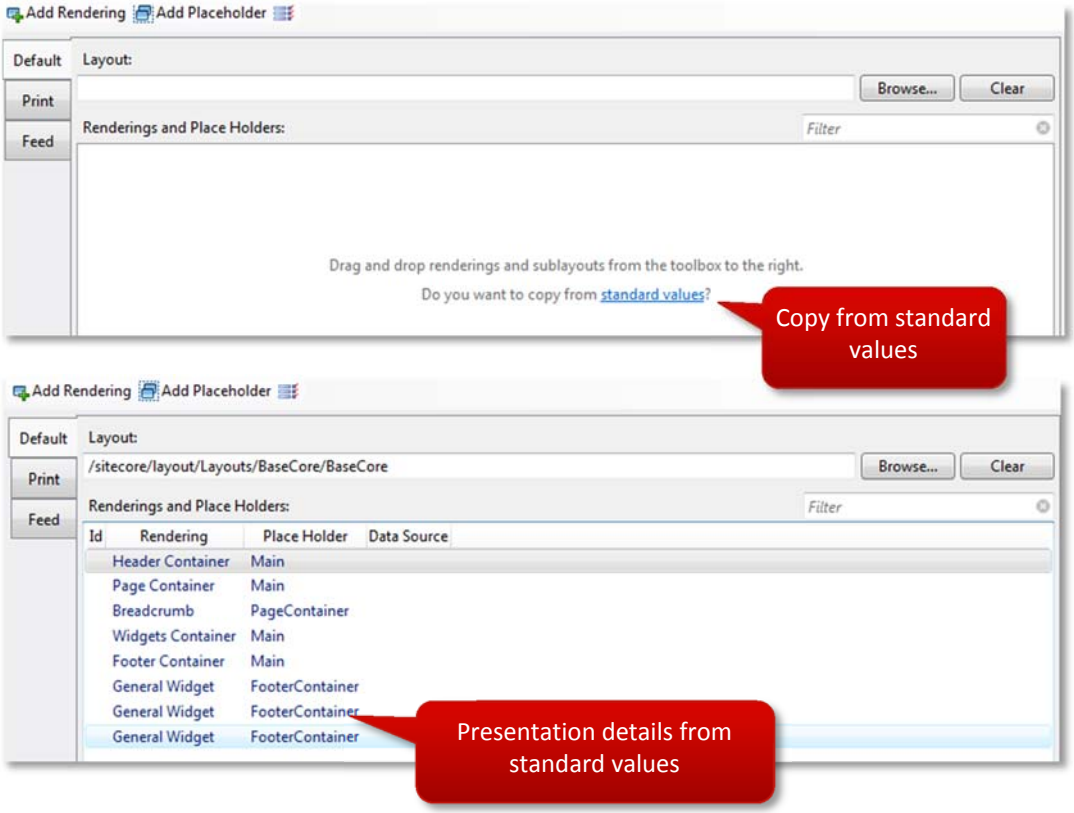


3. When prompted by Sitecore Rocks, add the corresponding **sublayout definition** item to */Traincore/master/sitecore/ /layout/Sublayouts/BaseCore/Content/.*

Tip

If you are not prompted to add a sublayout definition item, your project might not be connected to a Sitecore instance. Right-click the project, select the **Sitecore>Connect…** option and pick the Sitecore instance that you want to use.

4. A sample repeater has been provided in the **student resources folder** *(WND Labs > Module 6 > Topic 6.1 > Lab C > Code > Related Articles.ascx).* **Copy** and **paste** this code into your .**ascx** and then **save.**
Do *not* overwrite the entire file because the sample only contains the code for the repeater.
5. Navigate to the **News Article** template's **standard values** and open the **Design Layout** pane.
6. **Copy** presentation details from standard values when prompted.



7. Add the 2 following sublayouts, and assign them to the placeholder keys as shown in the following table. **Two Column Content** is a pre-existing sublayout.

Rendering name	Placeholder
Two-Column Content	PageContainer
Related Articles	TwoRightColumn

8. **Save** the solution.
9. Because we are working outside the web root, we need to **deploy** the changed files. To deploy, right-click on the Visual Studio toolbar and select **Web One Click Publish.** Depending on your setup, this may already be checked.
10. A **publish** toolbar will appear. Do not confuse this with Sitecore publishing!

11. Select the Traincore profile, and click the **globe icon** to deploy your files. This profile is set up as part of the Traincore installation process. If there aren't profiles available, please notify your instructor.

Knowledge Check

When you use a Sitecore control and specify a field – for example:
`<sc:Text field="Page Heading" runat="server" />`, what item does the control default to as its data source?

Lab B. Output a List of Related Articles

1. Using **Visual Studio,** open Related Articles.ascx.cs.

Knowledge Check

Look in the Sitecore.Data.Fields class. Which field object type would be most suitable for retrieving the **Related Articles** field from the context item?

---

2. In the **Page_Load** method, retrieve the **Related Articles** field on the context item as a MultilistField object.
3. Use the **GetItems()** method on the MultilistField object to retrieve a list of **selected items.** These items represent related articles you chose on each article.

Knowledge Check

What would the **.Value** property of the Related Articles field object contain?

Knowledge Check

Why wouldn't you use FieldRenderer.Render to output the contents of a multi-list field?

---

4. **Bind** the list of items to the repeater.
5. The sample repeater's <ItemTemplate> prompts you to add a Sitecore control. Replace **[SITECORE CONTROL]** with a **Sitecore text control** that outputs the **Page Heading** field.

Code Sample

```
<ItemTemplate>
        <li><a href="[URL HERE]">[SITECORE CONTROL]</a></li>
</ItemTemplate>
```

6. Because the control is in a repeater, we want to tell it to retrieve the **Page Heading** from each repeater item and not from the context item. To do this, set the **text control's** DataSource property to the **item's ID.** (**Note** the use of the <%#: Item.ID %> syntax because the repeater is strongly-typed; **Item** represents an object of type Sitecore.Data.Items.Item):

Code Sample

```
DataSource="<%#: Item.ID %>"
```

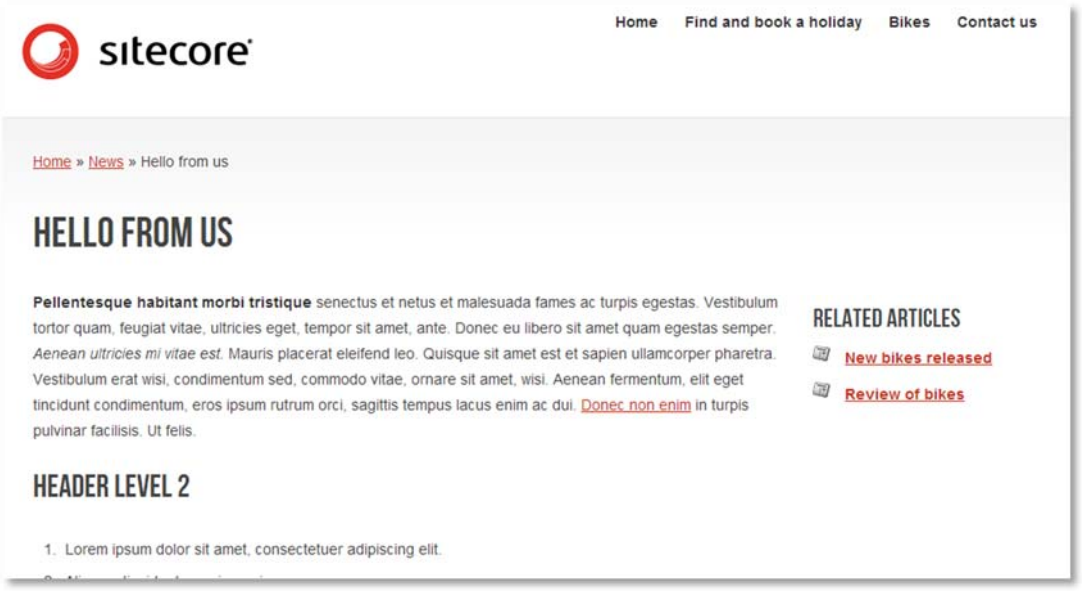
7. The sample repeater also prompts you to add a URL. Replace **[URL HERE]** with a **method** that retrieves the **SEO-friendly URL**. (Hint: refer to Module 4 if you are unsure of which method to use).

Code Sample

```
<a href="<%#: YOURMETHOD(Item) %>">
```

8. **Save** and **deploy** the solution.
9. **Preview** a **news article** page. The component in the right-hand column now displays a list of live links to the related articles specified in the **Related Articles** Multi-list.

News Article Page



Topic 6.2 Dealing with Larger Sites

Introduction

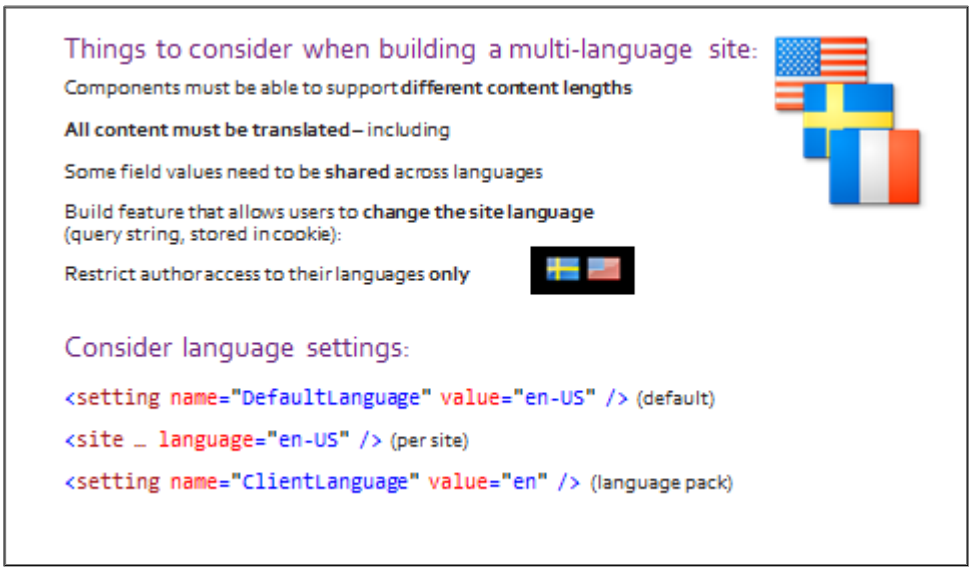
Objectives

By the end of this topic you will be able to:

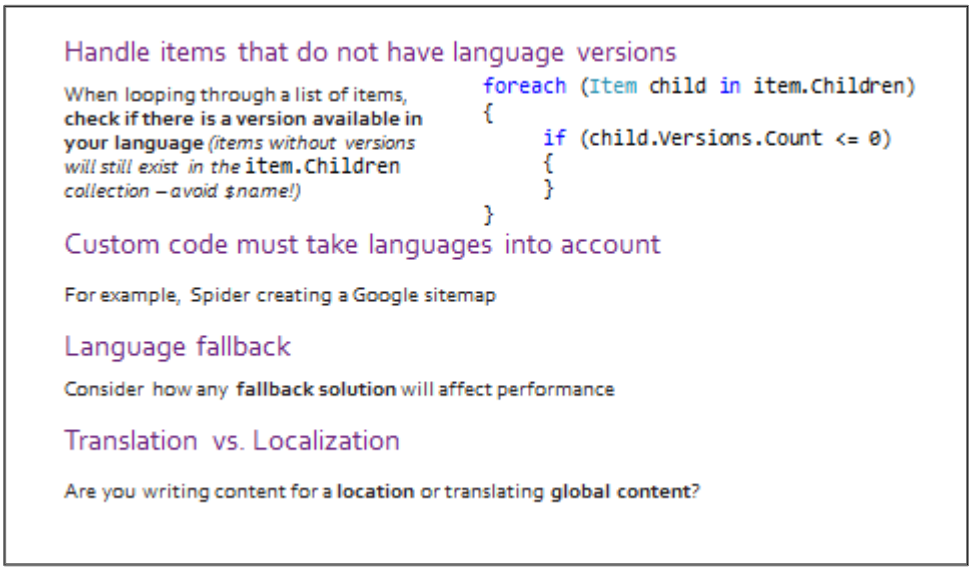
- List three considerations that must be taken into account when architecting a multi-language site.
- Name four language configurations and what they do.
- Account for items without versions.
- Configure a multi-site solution in web.config.
- List the limitations of a multi-site solution.

Content

Multi-Language Support



Remember!

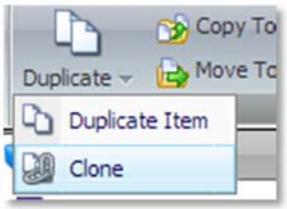


Demo – Multi-site

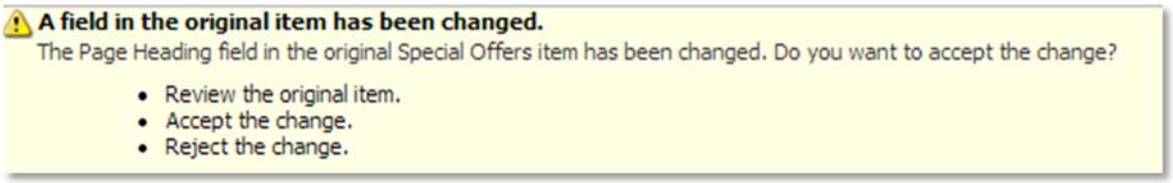
In the following demo, we will:

- Look at the configuration for multi-site, and add http://traincore-corporate/ .
- Create an **item clone.**

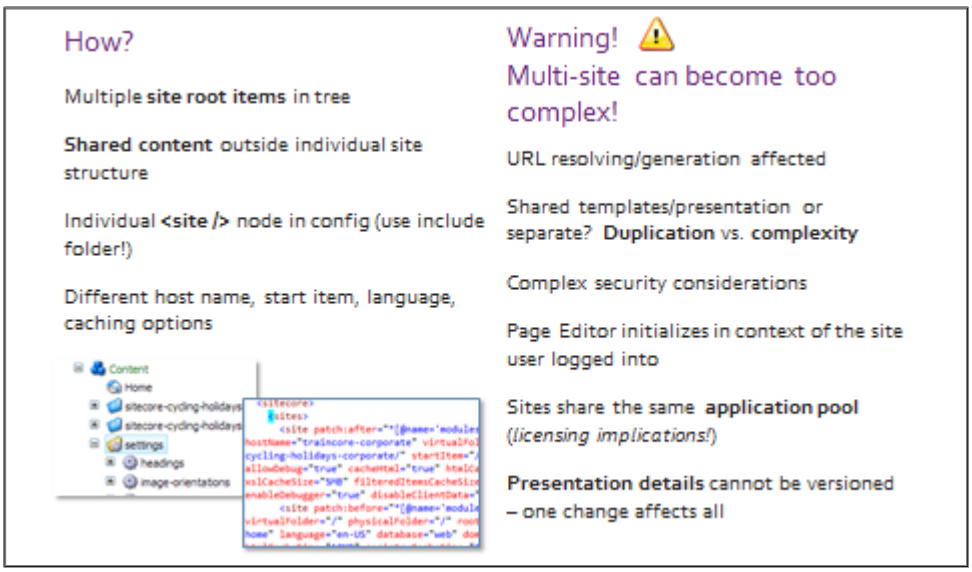
1. Using the Sitecore Rocks, expand the **/sitecore/content** item in Traincore and look at the two sites – *sitecore-cycling-holidays* and *sitecore-cycling-holidays-corporate.*
2. Preview the **Home** item of each site. They use the same presentation details in different configurations, and the corporate site is significantly smaller.
   - Content is independent, but presentation and data templates are shared. Both sites have exactly the same functionality.
   - You could have two entirely different sites stored in the same implementation (for example, a mobile-specific site, taking advantage of devices).
3. In the **Solution Explorer**, open *App_Config/Include/BaseCore.Sites.config*. **Note** the presence of two <site> nodes. **Note** that the site **host name, language, start path** and **start item** differs between sites.
4. Browse to */sitecore/admin/ShowConfig.aspx* to see the final, transformed configuration file. Locate the **<sites>** nodes again – they are merged into the main <sites> configuration section.
5. Open IIS and select **Traincore**. Add **traincore-corporate** to the site's bindings, and make sure it is included in your **hosts file**.
6. Browse to http://traincore-corporate to see the live site.
7. You can share content between sites using **cloning**. In the **Content Editor**, Select the **terms-and-conditions** item under */sitecore-cycling-holidays/Home*. You can do this in Sitecore Rocks as well, but this task tends to be performed by authors.
8. Select the item, and click the **Home** tab **> Duplicate drop-down > Clone**:



9. Choose **/sitecore-cycling-holidays-corporate/Home** as the item's new parent. **Note** that the cloned item's name is greyed out.
10. Make a change to the clone's **Page Heading** field.
11. Go back to the original item and make a change to the Page Heading of this item.
12. Go back to the clone. It has picked up on the change to the parent, and offers tree choices:



Multi-site



Important

Because the corporate site is very small and receives very little traffic, it uses the same application pool as the main site. **Note** that multiple sites using multiple application pools have licensing implications.

Recommended Practices

A multi-site installation is not always appropriate. For example:

The more discrepancies there are between sites in the same installation the larger and more complex the codebase becomes and the more convoluted the Sitecore configuration becomes.

When all sites use the same application pool, the performance of one affects all of the others.

Sitecore Cycling Holidays works as a multi-site solution because the sites share exactly the same functionality. You would not mix a holiday booking site and a retail site with nothing in common.

Extend

- **Dictionary Domains**
  http://www.sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog/Posts/2012/11/Sitecore-ASPNET-CMS-6-6-Features-Dictionary-Domains.aspx
- **Re-using and Sharing Data**
  http://sdn.sitecore.net/upload/sitecore7/70/reusing_and_sharing_data_sc70-a4.pdf

Topic 6.3 Sitecore Query

Introduction

By the end of this topic you will be able to:

- Write simple Sitecore queries using a variety of operators and functions.
- Write Sitecore queries that use axes.
- Discuss appropriate uses of Sitecore query and when to opt for the search API.

Content

Demo - Sitecore Query

In the following demo, we will:

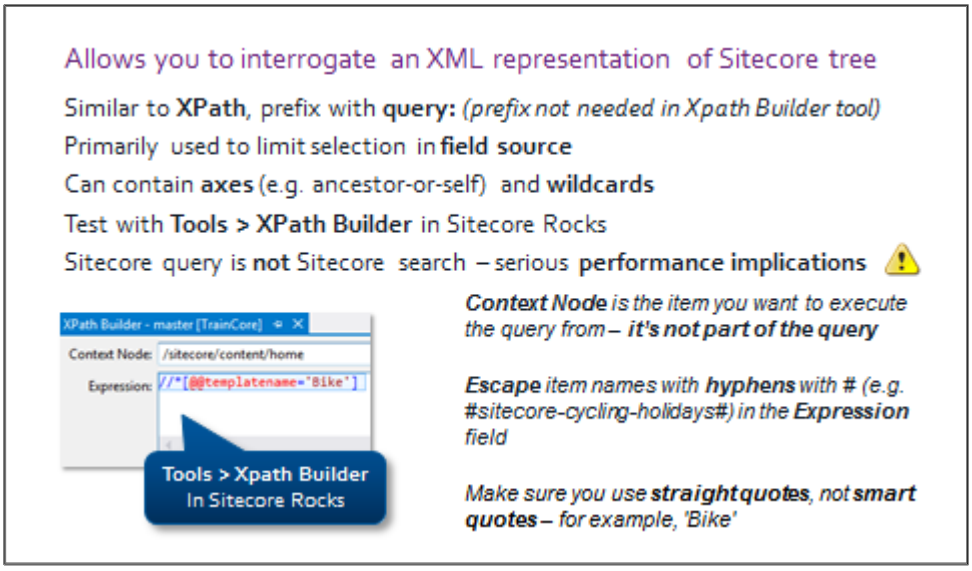
- Perform various queries on the Traincore tree using the XPath Builder in Sitecore Rocks.

1. Using Sitecore Rocks, open the **XPath Builder** by right-clicking anywhere in your tree and selecting *Tools > XPath Builder*.
2. Experiment with various queries (examples below). Use */sitecore/content* as your Context Node:

Absolute query
/sitecore/content/#sitecore-cycling-holidays#/*
Relative query
.//*[@@templatename='Bike']
Filter by template
/sitecore/content/#sitecore-cycling-holidays#//*[@@templatename='Bike']
Filter by content
/sitecore/content/#sitecore-cycling-holidays#//*[@#Page Title# = 'News']
Return at index
/sitecore/content/#sitecore-cycling-holidays#/*[3]
ancestor-or-self::*
*/sitecore/content/#sitecore-cycling-holidays#/ancestor-or-self::**
ancestor-or-self::[@@...]
/sitecore/content/#sitecore-cycling-holidays#/Home/holidays/#battle-of-the-hills#/ancestor-or-self::[@@templatename='Holidays']* *./ancestor-or-self::*[@@templatename='Holidays']* ***(Context node: /sitecore/content/sitecore-cycling-holidays/Home/holidays/battle-of-the-hills)***
Parent
/sitecore/content/ #sitecore-cycling-holidays#/parent::/child::**
Following
/sitecore/content//[@@name='sitecore-cycling-holidays']/following::**

3. Using Sitecore Rocks, navigate to the **Holiday** data template and look at the **Terrain** field. **Note** the presence of an **ancestor-or-self** query that returns all values relevant to the context site.
4. Go to an item based on the **Holiday** data template – the selection is restricted by the query.

Sitecore Query



Important

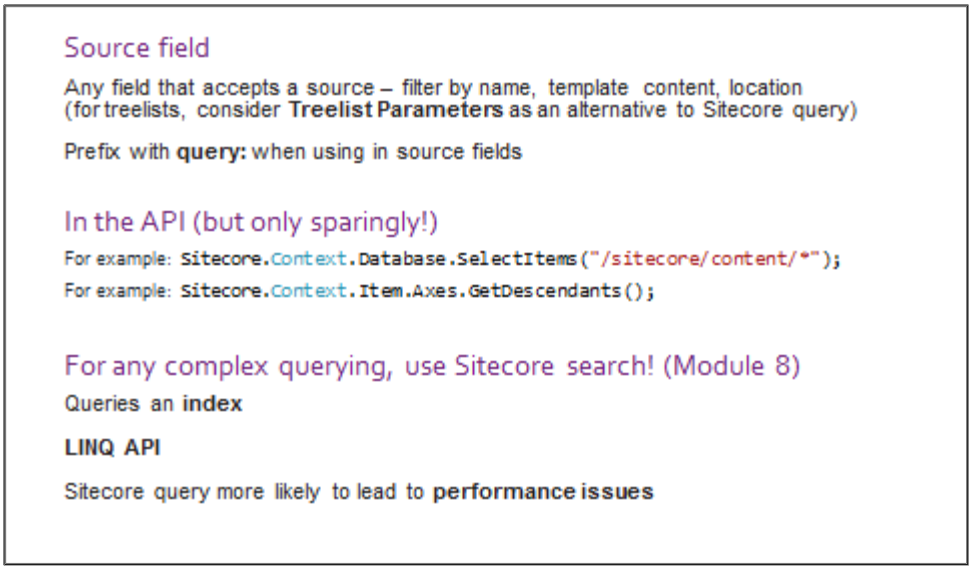
- Queries are useful in a **multi-site environment** when you need to get items without using a path or GUID.
- Queries can be **expensive** – We do not recommend using queries in a **high traffic environment.**
- Don't query the entire content tree.
- **Note** that when you use complex queries it may affect performance.
- The result set will be limited by the **Query.MaxItems** setting in the web.config.
- You have to **sort the results** when you return them.

Query Syntax

Specific path	*query:/sitecore/content/home/*
Gets all immediate items under *home* **Note:** Use if there is only one location, for example: /settings and when settings is shared by all sites **Cannot be used in a multi-site.**	
* wildcard	*query:/sitecore/content/*/news*
Gets items under content/[anything] called news	
/ children	*query:/sitecore/content/home/**
Gets all children of *home* / followed by GUID, item name, or *	
// descendants	*query:/sitecore/content//*[@@templatename='News']*
Gets all items that have home as ancestor and have *News* as the template name // followed by GUID, item name, or * **Note:** Can be **expensive** if the tree is large	

@@ XML attribute	*query:/sitecore/content/[@@templatename='News Listing']*
Gets all items under */home* that use the *News* template e.g. @@templatename or @@templateid **Note:** Case sensitive	
@ field	*query:/sitecore/content/*[@#Page Title# = 'Home']*
Gets any items under */home* that have a field called *Page Title* and field value of *News* **Note:** Use # to escape spaces and hyphens in field names	
. context item	
In the case of a source field, it's the item you are currently viewing in the content tree	
[1] index	*query:/sitecore/content/*[3]*
Returns the item under *home* at index 3 **Note:** It's not a zero-based index	

Uses for Sitecore Query



Tip

For more information on Treelist Parameters, see p12:
http://sdn.sitecore.net/upload/sitecore6/datadefinitioncookbook-a4.pdf

Apply – Topic 6.3 – 10 min

Sitecore Query in Datasources

Lab A. Set Related Articles Source

To ensure that the correct items are picked in a list field, restrict the selection to a particular location and/or template type. This is particularly important in multi-site installations, where each site might have individual locations for list content.

1. Use Sitecore Rocks to open */master/sitecore/content/templates/BaseCore/Pages/News Article.*
2. In the far-right text field, next to the **Build** button, build a query that satisfies the following conditions:

   - The items must have the template **News Article**.
   - The items must appear under the same parent as each other (NOT all items in the tree based on the News Listing item. For example, if there are 10 articles in your tree in the tree, your query should only return the News Article items that are siblings of the one you are currently looking at) (**Hint**: you are looking for any item whose ancestor-or-self:: has a type of **News Listing**).

Extend

- **Fast Query**
  http://sdn.sitecore.net/upload/sdn5/developer/using%20sitecore%20fast%20query/using%20sitecore%20fast%20query001.pdf
  http://www.sitecore.net/unitedkingdom/Community/Technical-Blogs/John-West-Sitecore-Blog/Posts/2012/05/Sitecore-Query-Cheat-Sheet.aspx
- **Parameterized Datasources**
  http://sdn.sitecore.net/upload/sitecore6/datadefinitioncookbook-a4.pdf (Search: "Treelist Parameters")

Module 7
Configuring the Page Editor

Contents:

- Datasource Restrictions
- Parameters and Parameter Templates
- Placeholders Restrictions
- Advanced Page Editor Configuration

Topic 7.1 Datasource Restrictions

Introduction

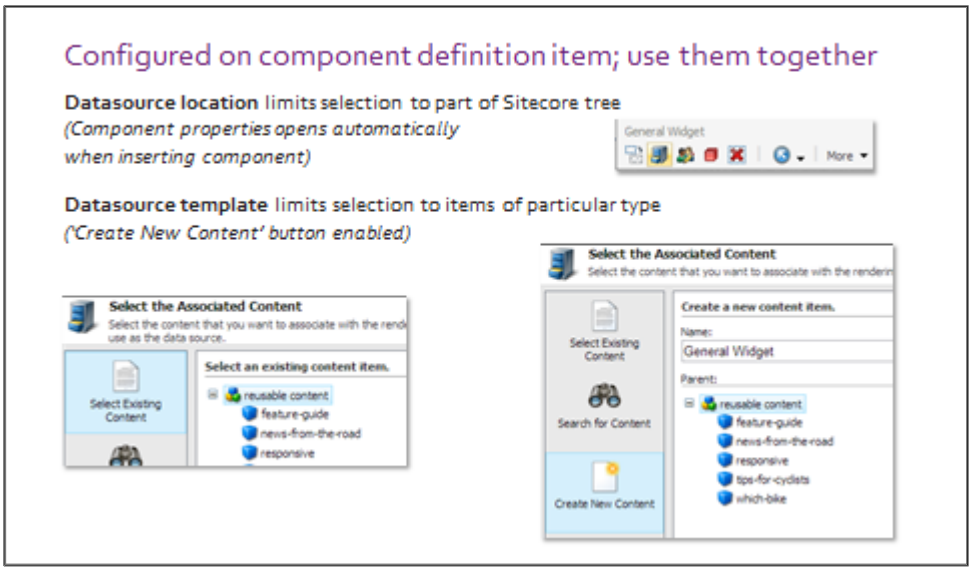
Objectives

By the end of this topic you will be able to:

- Restrict the type of item that can be selected as a datasource.
- Restrict the location from which an item can be selected as a datasource.

Content

Datasource Location and Template Restrictions



Tip

An item can be selected from anywhere in the tree, so location restrictions do not apply. If you restrict **only** the datasource template, then items that are not selectable will be **ghosted** (greyed out), and authors will not be able to select them.

The **Select the Associated Content** dialog will not pop up automatically, but it can be accessed via the **Select Associated Content** button in that component's floating toolbar in the Page Editor.

Demo – Apply Restrictions

In the following demo, we will:

- Apply datasource restrictions using the **Page Editor.**
- View datasource restrictions in Sitecore Rocks.

1. Open the **Page Editor,** and ensure that the **design mode** is enabled.
2. Select the **General Widget** component and select **More > Edit Page Editor Options** in the floating toolbar.
3. Click **Insert Link** above the **Datasource Location** field and select */sitecore/content/sitecore-cycling-holidays/Global/reusable content*. Click **OK**.

4. Add a second General Widget component.
5. A prompt appears, asking you to "**select associated content**". Select a datasource item, and click **OK.**
6. Select the same General Widget component, and select **More > Edit Page Editor Options** again.
7. This time, click **Insert Links** above the **Datasource Template** field and select */sitecore/Layouts/Sublayouts/BaseCore/Global Content/General Widget.*
8. Add another General Widget component to the page. Notice that you now have an option to **Create New Content** when you are prompted to add a datasource. Items that do not match the selected data template, such as the root **reusable contents** item, are greyed out.
9. Select **Create New Content**, give the new datasource a name, such as *The Best Mountain Bikes*, and click **OK**.
10. Populate the content of the datasource inline, and **Save** the page.
11. Switch to **Sitecore Rocks**.
12. Navigate to the **reusable content** folder, and note that the datasource named *The Best Mountain Bikes* was created behind the scenes.
13. Navigate to */sitecore/Layouts/Sublayouts/BaseCore/Global Content/General Widget*. Notice that the **Datasource Location** and **Datasource Template** are available in Sitecore Rocks as well.

Apply – Topic 7.1 – 25 min

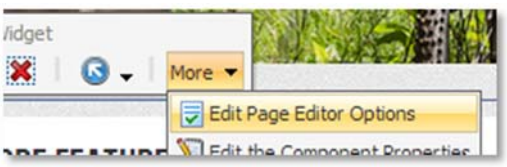
Datasource Restrictions

In the following labs you will learn to use the **Page Editor** to restrict the location and type of items authors can select as a datasource.

Lab A. Set Datasource Location

1. Log in to the **Page Editor** interface.
2. Ensure that you are in **design mode**.
3. Navigate to the **Home** page, and select any existing General Widget component.

4. In the **floating toolbar**, select **More > Edit Page Editor Options.**



5. Click the **Insert Link** button above the **Datasource Location** field.

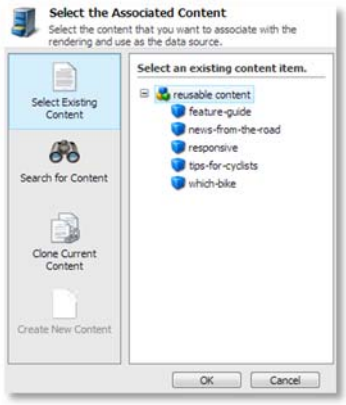
6. In the item browser dialog, locate and select: /sitecore/content/sitecore-cycling-holidays/Global/reusable content, then click **OK.**

Lab B. Add a Component

1. Select **Add New Component** on the ribbon.

2. Add a new **General Widget** to the **widgetscontainer** placeholder.

3. The **Select Associated Content** dialog appears. Notice that the **Create New Content** tab is greyed out. Select any **item** from that list and click **OK.**



4. **Save** the page.

Lab C. Set Datasource Template

1. Select the General Widget component you just added to the page.
2. Select **More > Edit Page Editor Options** in the floating toolbar.
3. Set the **Datasource Template** field to point to */sitecore/templates/BaseCore/Global Content/General Widget.*

Lab D. Add Another Component

1. Add another General Widget component to the **widgetscontainer** placeholder.
2. Notice that the **Create New Content** button is now available in the **Select Associated Content** dialog. Click it, and give the new datasource a name,such as *New Bikes Available*. This creates a new Sitecore item in the background.
3. Populate the General Widget inline, and **Save** the page.

Topic 7.2 Parameters and Parameter Templates

Introduction

Objectives

By the end of this topic you will be able to:

- Create and assign a parameter template.
- Describe the benefits to an author of using a parameter template.

Content

Demo – Using a Parameter Template

In the following demo, we will:

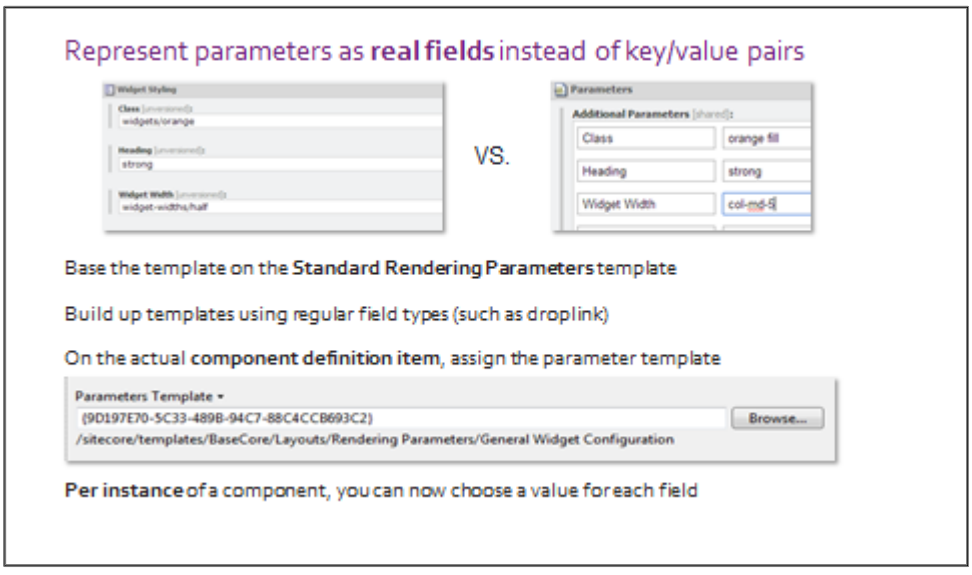
- Create and assign a parameter template using Sitecore Rocks.
- Show the result in the Page Editor.

1. In Sitecore Rocks, create a parameter template based on **Standard Rendering Parameters**. Name it *Related Articles Rendering Parameters.*
2. Add a sample single-line text field named **CSS Class** to the template.
3. Assign the rendering parameters to the **Related Articles** component.
4. Using the Page Editor, browse to one of the News Article items you created in Module 6. Select the **Related Articles** component and select **Edit the Component Properties**.
5. **Note** the presence of the **CSS Class** field alongside **Datasource** and **Placeholder**.

Knowledge Check

What problems might you encounter if you do not use parameter templates, and allow authors to specify parameter key/value pairs themselves?

Specify Parameters Using a Parameter Template



Apply – Topic 7.2 – 50 min

Parameter Templates

In the following labs you will:

- Create a parameter template.
- Assign a parameter template to a component.
- Use the Page Editor to set parameters on a component.
- Output the contents of the parameter field.

Lab A. Create and Assign Parameter Template

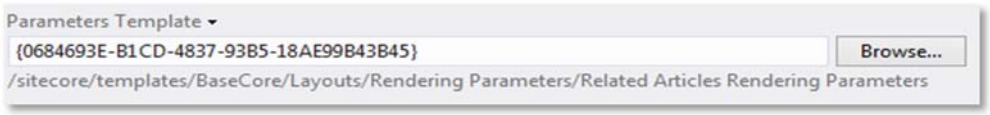
1. Using Sitecore Rocks, navigate to: */master/sitecore/templates/BaseCore /Layouts/Rendering Parameters* and **create** a new rendering parameters template.
2. Right-click and select the **Insert a New Template…** option.
3. Name the new template *Related Articles Rendering Parameters.*
4. Set the base template to **Standard Rendering Parameters.**
5. **Name** the first section *Styles* and **create** a field called *Class* of the type *Droplink* and the source */sitecore/content/settings/styles/widgets.*

Knowledge Check

Given that you will probably have a limited set of CSS classes that you want to re-use across the site, what might be a good choice for field versioning?

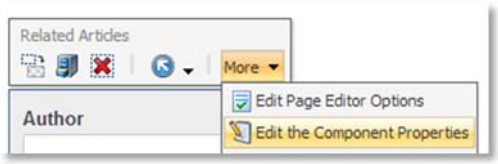
---

6. Navigate to the **Related Articles** sublayout that you created in Module 6: */master/sitecore/layout/Sublayouts/BaseCore/Content/Related Articles.*
7. Find the **Parameters Template** field in the **Editor Options** field section, and click the **Browse**… button.
8. Locate and select the **Related Articles Rendering Parameters** template that you created in step 2.



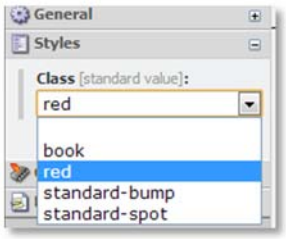
9. **Save** the sublayout.
10. Using the Page Editor, navigate to one of the news article that you created earlier.

    Select the **Related Article** component and in the floating toolbar and click the **More** button and then the **Edit the Component Properties** option.



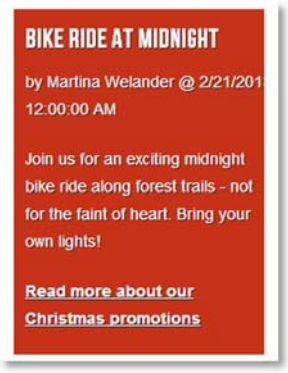
11. Confirm that the **Class** field now appears in the **Component Properties** window.

12. Select *red* from the **Class** drop-down list and click **OK** and **Save.**



Lab B. Use Parameters in Code

1. Using **Visual Studio,** navigate to the **Related Article** sublayout and **open** the .cs file. Refer to **Module 5** for code samples.
2. **Retrieve** the **sublayout parameters.**
3. **Return** the **Class** parameter.
4. The parameter template you created earlier defined "Class" as a droplink. The **value** of the **Class parameter** is going to be a **Sitecore ID** as a **string.**
5. **Convert** this **string** to a **SitecoreID** and use it to **retrieve** the **referenced item.**
6. The referenced item has a field named **CSS Classes**. Retrieve the **raw value** of this field as a string, and make it available as a **public property**.
7. In the .ascx file, use code blocks to output the Class string `<div class="chunk widget">` (Be sure to leave a space).
8. **Save** and **Deploy** the Visual Studio solution.
9. **Browse** to the news article with the **Related Article** component and confirm that this component now has the style specified in the **Parameter Template** field.



Topic 7.3 Placeholder Restrictions

Introduction

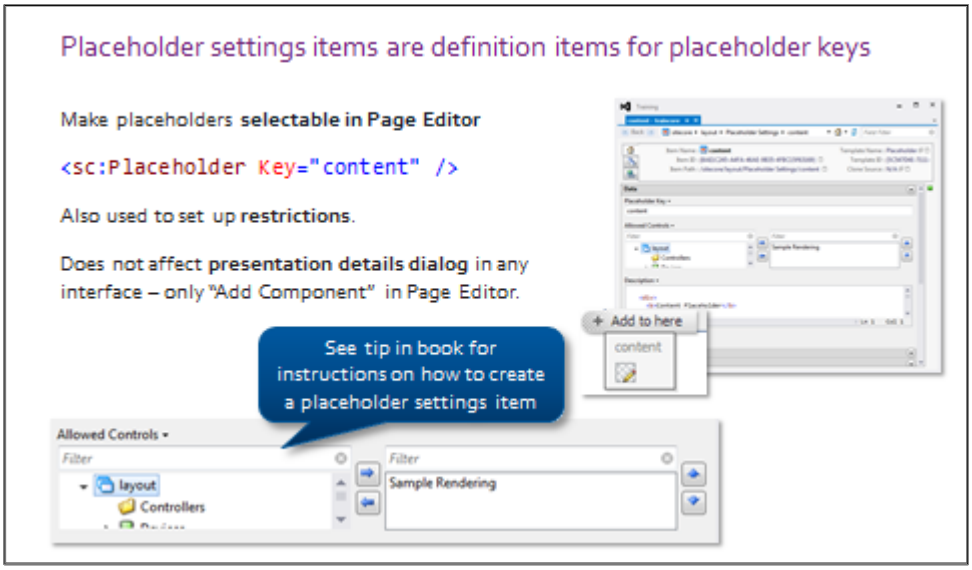
Objectives

By the end of this topic you will be able to:

- Describe the purpose of a placeholder settings item.
- Explain what compatible renderings are.

Content

Placeholder Settings Items



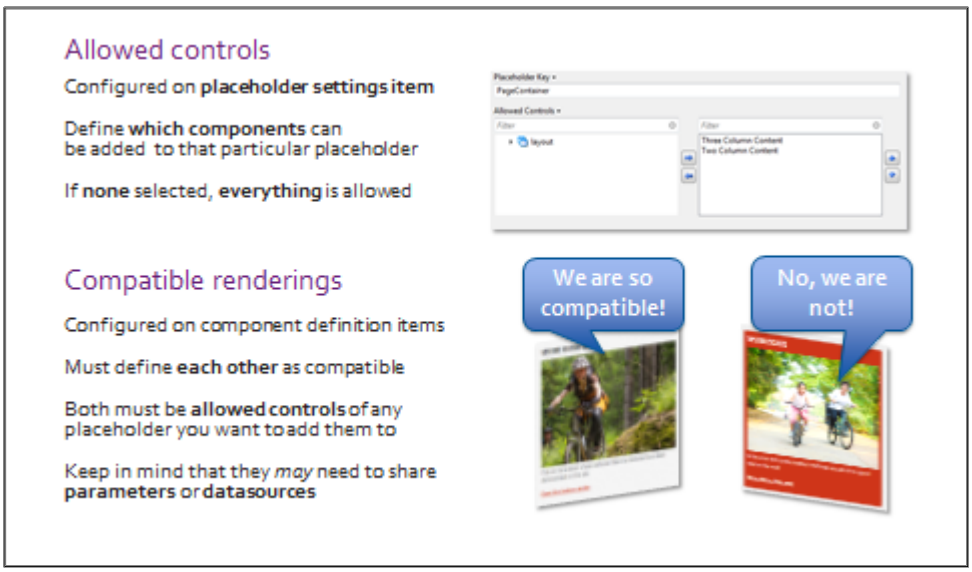
Tip

To create a placeholder settings item, insert a new item under: */sitecore/Layouts/Placeholder Settings*. Make sure the **Placeholder Key** field is populate. We recommend that you use lowercase for the placeoholder item name and key, because this makes it easier to identify as a placeholder in the component hierarchy.

Tip

Use the WebEdit.PlaceholdersEditableWithoutSettings setting in web.config to make all placeholders selectable even if they do not have a placeholder settings item.

Compatible Renderings and Allowed Controls



Demo – Allowed Controls and Compatible Renderings

In the following demo, we will:

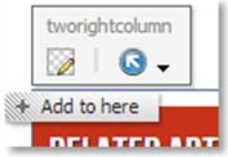
- Use the **Compatible Renderings** button.
- Assign allowed controls.
- Assign compatible renderings.

Compatible Renderings

1. Open the **Page Editor,** and ensure **design mode** is enabled.
2. On the **Home** page, select the Gallery component. You may need to click the **blue arrow** to navigate to the gallery component itself. Notice the presence of an icon called **Replace with compatible rendering**.
3. Select **Banner** when prompted, and **save**. Rather than a scrolling gallery with multiple images, you now have a single image.
4. Replace the **Banner** component with the **Gallery** component again.
5. Select the **Banner component** again and, in the **floating toolbar,** click the **blue arrow** to navigate to the components **placeholder** (gallerycontainer).

Allowed Controls

1. In the Page Editor, browse to the **news article** you created earlier under: */sitecore/content/sitecore-cycling-holidays/Home/news*
2. Select the **tworightcolumn** placeholder, which is the narrow column within the **Two-Column Content** sublayout. Click the pencil icon in a square to **edit the placeholder settings**.

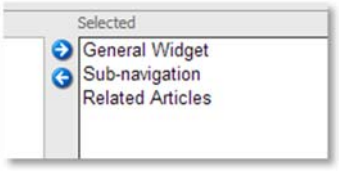


Knowledge Check

Why can't authors add Related Articles to the placeholder?

---

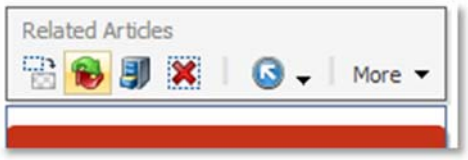
3. Click **Edit** next to the **Available Controls** multi-list, and add **Related Articles** to the right-hand side. Click **OK**.



4. Select the **tworightcolumn** placeholder again and click **Add to here.** Add **Related Articles** to the page.

Configuring Compatible Renderings

1. Remain on the news article page, in the Page Editor interface. Select the Related Articles component and select **More** > **Edit Page Editor Options.**
2. Scroll down to the **Compatible Renderings** field. Locate the **Subnavigation** component (the path is */sitecore/layout/Renderings/BaseCore/Sub-navigation).* Double-click the component to move it into the right-hand column, then click **OK**.
3. Select the **Related Articles** component. Notice that the compatible renderings button appears in the floating toolbar. Click it, and select **Subnavigation** as a replacement for Related Articles.



Knowledge Check

If you select the Subnavigation component, why is there now a **Compatible Renderings** button?

Topic 7.4 Advanced Page Editor Configuration

Introduction

Objectives

By the end of this topic you will be able to:

- Create a custom experience button.
- Create an edit frame.

Content

Walkthrough – Custom Experience Buttons

In the following walkthrough, we will:

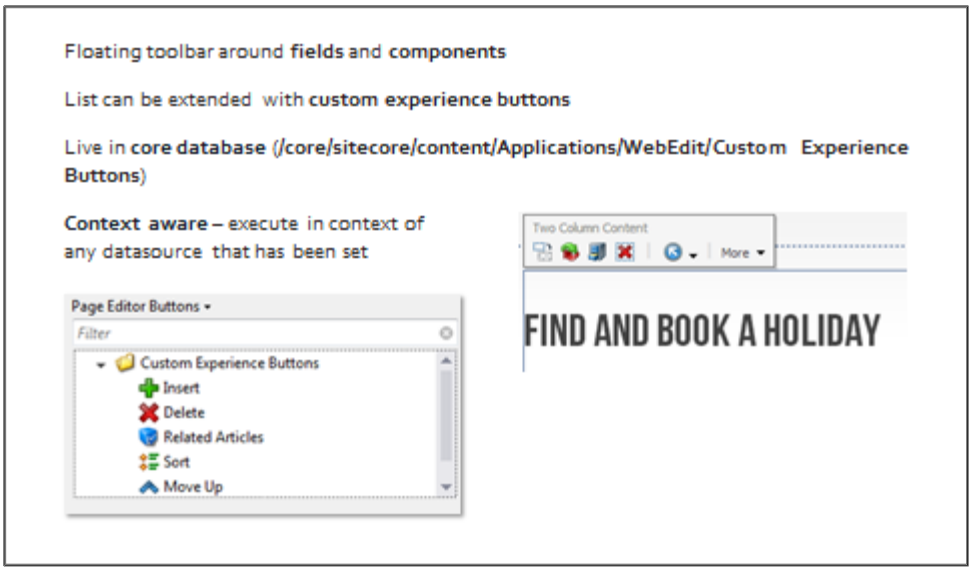
- Change an existing button on the floating toolbar for all components.
- Add an additional custom experience button to the floating toolbar of the **General Widget** component.

1. In the **Page Editor,** ensure design mode is enabled and select any component. **Note** the variety of buttons that appear on the floating toolbar. Each of these buttons fires a **command**. These commands are configured in the **core database.**

2. Switch to **Sitecore Rocks.** Open the **core** database and navigate to: */sitecore/content/Applications/WebEdit/Default Rendering Buttons/Delete.* Double-click to open it. This is where Sitecore's default buttons live.
3. In the Editor pane, change the **Tooltip** field from *"Remove component" to "Delete component".*
4. Switch back to the **Page Editor**. Refresh the page. Select the same component, and hover over the **Delete** button. **Note** that the tooltip text has changed.
5. You can create your own buttons or use some of Sitecore pre-existing ones. On the floating toolbar, click **More > Edit Page Editor Options.**
6. In the **Page Editor Buttons** field, select a number of additional controls. Click **OK,** refresh the page, and confirm that the new buttons appear.

7. Switch back to Sitecore Rocks. The button you just selected comes from the **Custom Experience Buttons** list in the core database. Navigate to: */sitecore/content/Applications/WebEdit/Custom Experience* **Buttons.** You can insert your own custom experience buttons here.

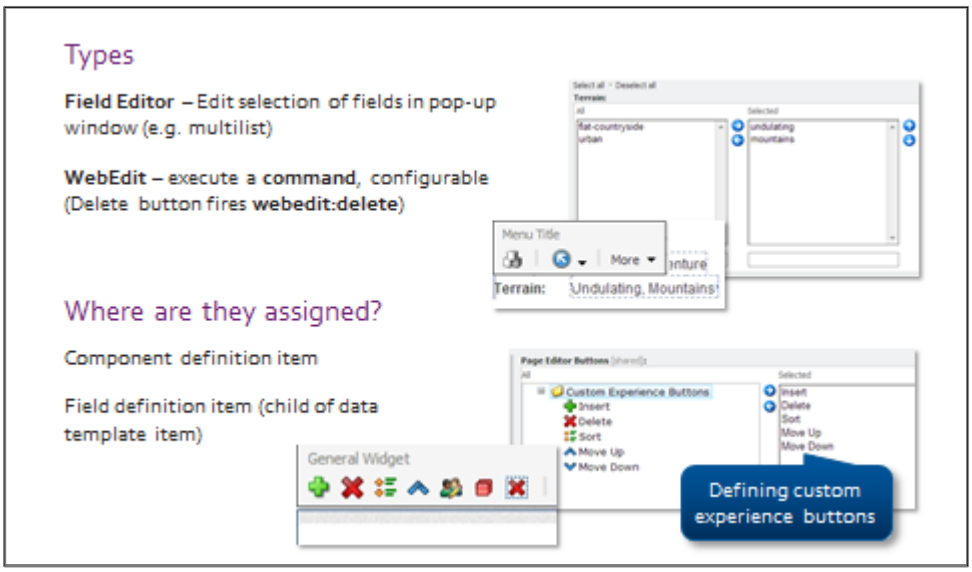
Custom Experience Buttons



Tip

Commands are listed in the /App_Config/Commands.config in the file system. They have a name and an associated type, and inherit from: Sitecore.Shell.Applications.WebEdit.Commands.WebEditCommand.

Types and How to Assign Them



Edit Frames

Sitecore Control Pointing to List of Custom Experience Buttons

```
<sc:EditFrame Buttons="My Edit Frame" runat="server">
    Option 1, Option 2, Option 3
<sc:EditFrame />
```

Defined in code; surround **any mark-up** (for example, content from a Multilist field)

Floating toolbar will appear in that location

Buttons attribute points to folder of **WebEdit/Field Edit buttons** in Sitecore tree

Defined in **core database** – */core/sitecore/content/Applications/WebEdit/Edit Frame Buttons*

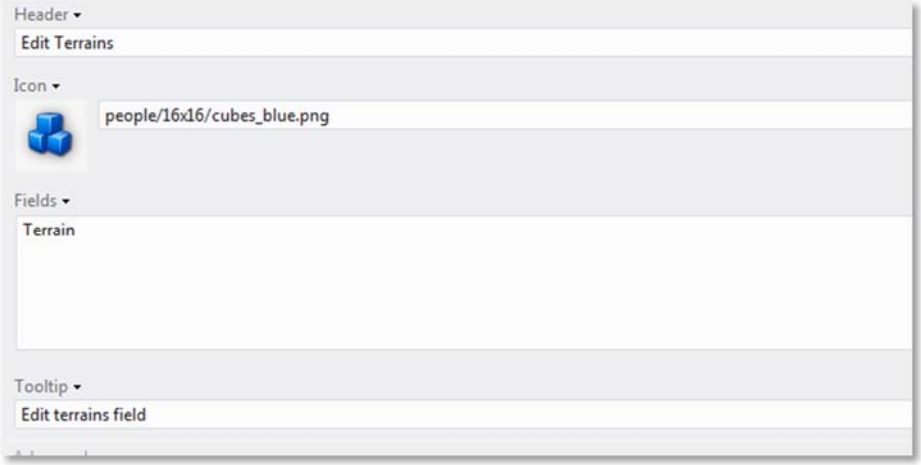
Not aware of datasource (but can be assigned using **DataSource** attribute)

Demo – Edit Frames

In the following walkthrough, we will:

- Create a **Field Editor** button.
- Insert an edit frame into the **Holiday Summary** widget.

1. Using **Sitecore Rocks,** navigate to: */core/sitecore/content/Applications/WebEdit/Edit Frame Buttons*
2. Duplicate the **Default** folder and rename it to *Holiday Summary.*
3. Open the **Holiday Summary** item and change the **Default Title** field to *Holiday Summary***.**
4. Delete all except the **Edit** item, which is based on the **Field Editor Button** template.
5. On the **Edit** item, change the **Heading, Fields** and **Tooltip** as follows.



6. Open the **bascore-widget-holiday-summary.ascx** sublayout: */layouts/BaseCore/widgets/basecore-widget-holiday-summary.ascx.*
7. **Surround** the entire definition list **HTML** (see code sample below) in the list with an **<sc:EditFrame />.**

8. Set the **Buttons ID** to the ID of the **Holiday Summary folder** item you created in the **core** database**.**

Code Sample – Edit Frame

```
<sc:EditFrame ID="EditFrame1" Buttons="/sitecore/content/path/to/buttons" runat="server">
<dl>
<!-- ... --></dl>
</sc:EditFrame>
```

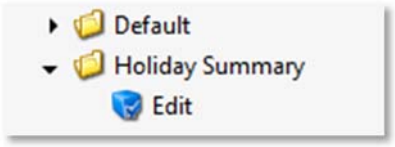
9. **Save** and **deploy** the solution.
10. Switch to **Page Editor** mode and navigate to a holiday page, for example, ***Battle of the Hills.***
11. Switch on **Editing**, but **switch off Designing.**
12. Click the whitespace within the holiday summary list. Your edit frame will appear around the content.
13. Click the **command** on the **floating toolbar.** Your fields will appear in a pop-up window.

Apply – Topic 7.4 – 15 min

Edit Frame

Lab A. Add an Edit Frame

1. Using **Sitecore Rocks**, navigate to **/core/sitecore/content/Applications/WebEdit/Edit Frame Buttons.**
2. Duplicate the **Default** folder and rename it to **Holiday Summary.**
3. Open the **Holiday Summary item** and change the **Default Title** field to **Holiday Summary.**
4. **Delete** all except the **Edit** item, which is based on the **Field Editor Button** template.



5. Open the **Edit** item.

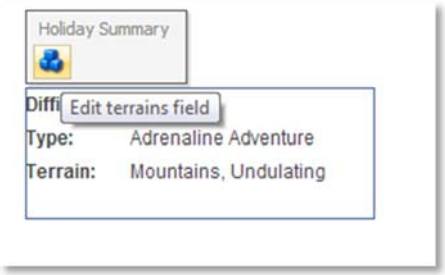
6. Change the **Heading, Fields** and **Tooltip** as follows.

7. Switch to **Visual Studio.**
8. Open the **bascore-widget-holiday-summary.ascx** sublayout - /layouts/BaseCore/widgets/basecore-widget-holiday-summary.ascx.
9. Surround the entire definition list HTML (see code sample below) with an <sc:EditFrame />.
10. Set the **Buttons** property to the **path** of the Holiday Summary folder item you created in the core database.

Code Sample – Edit Frame

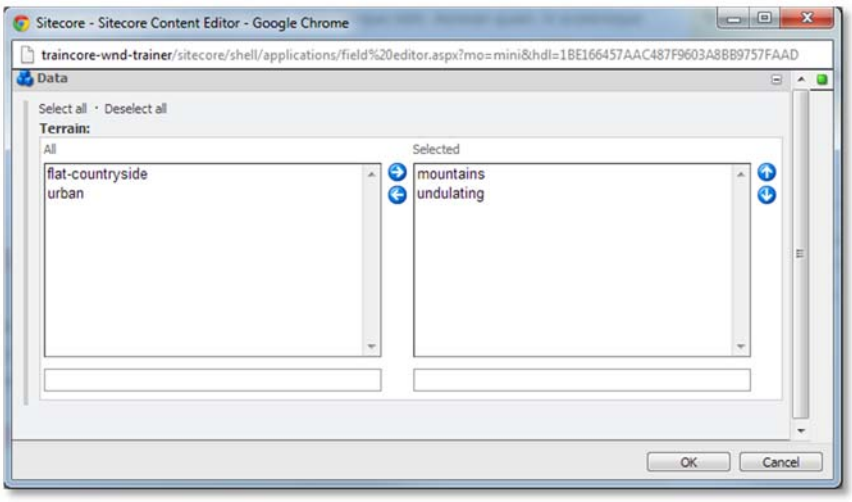
```
<sc:EditFrame ID="EditFrame1" Buttons="/sitecore/content/path/to/buttons" runat="server">
<dl>
<!-- ... -->
</dl>
</sc:EditFrame>
```

11. **Save and deploy** the solution.
12. Switch to **Page Editor** mode and navigate to a **holiday page** – for example, Battle of the Hills.
13. Switch on **Editing,** but switch off **Designing.**
14. Click the whitespace within the holiday summary list. Your edit frame will appear around the content.



15. Click the **Cubes** command in the **floating toolbar.**

16. The **Terrain** field opens up in a pop-up window.



Knowledge Check

If the **DataSource** property is not set, where will the edit frame try to retrieve the field values from?

Module 8
Dealing with Your Data

Contents:

- Item Buckets
- Search

Topic 8.1 Item Buckets

Introduction

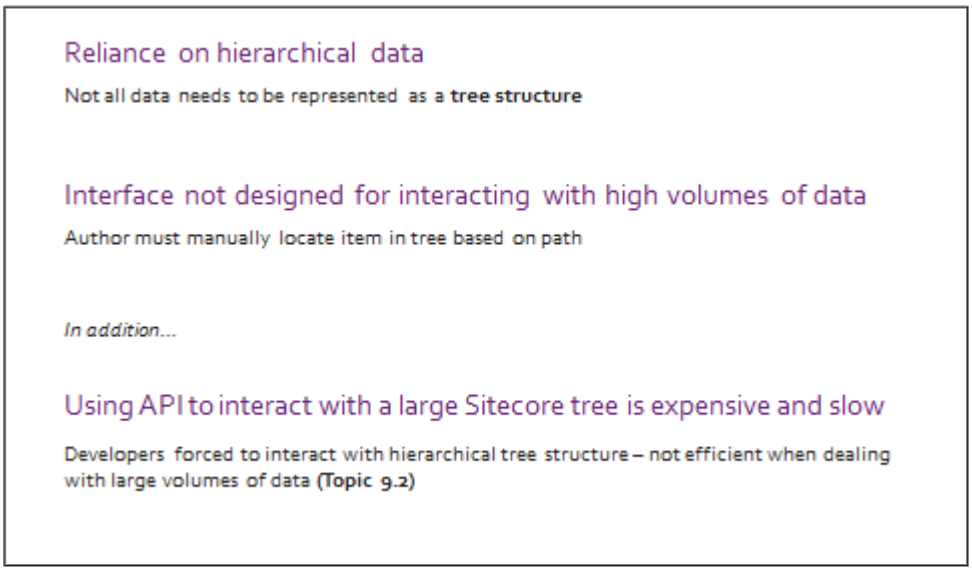
Objectives

By the end of this topic you will be able to:

- Explain how buckets solve UI concerns in large implementations.
- Create buckets and bucketable items.
- Configure bucket settings and facets.
- Use the buckets interface to search for and manage content.

Content

The Scenario



Walkthrough – Using Buckets as an Author

In the following demo, we will:

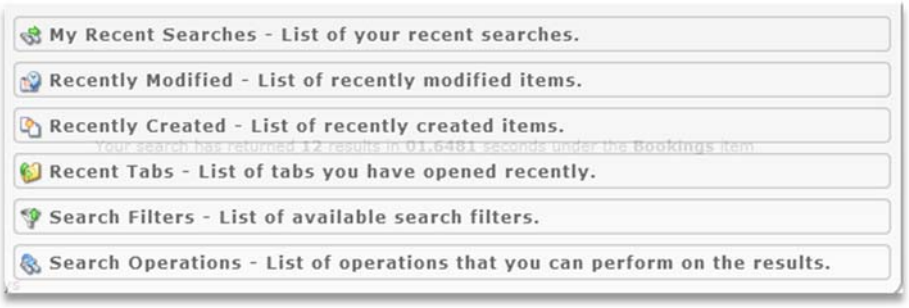
- Use bucket to **search** for bookings.
- Narrow down results using **facets.**
- Edit and create items as normal.
- Use **plain-text search** to search for a widget we know exists, for example, *which-bike*. Use **wildcards, replacements**, and **exact phrases.**
- Change the results view to **Gallery** or **GUID view.**
- Replicate search mechanism in their code – **query an index rather than a tree.**

1. Using the Desktop interface, open the **Content Editor.**
2. Right-click the **quick action bar** and select **Item Bucket.** This will highlight items in the tree that are buckets.
3. Navigate *to /sitecore/Content/sitecore-cycling-holidays/Bookings.*
4. Click the **Search** tab and type * to show all results.
5. Type *mar*ina* to show that **wildcards** work as expected – you can also use **"?"** in place of the asterisk.

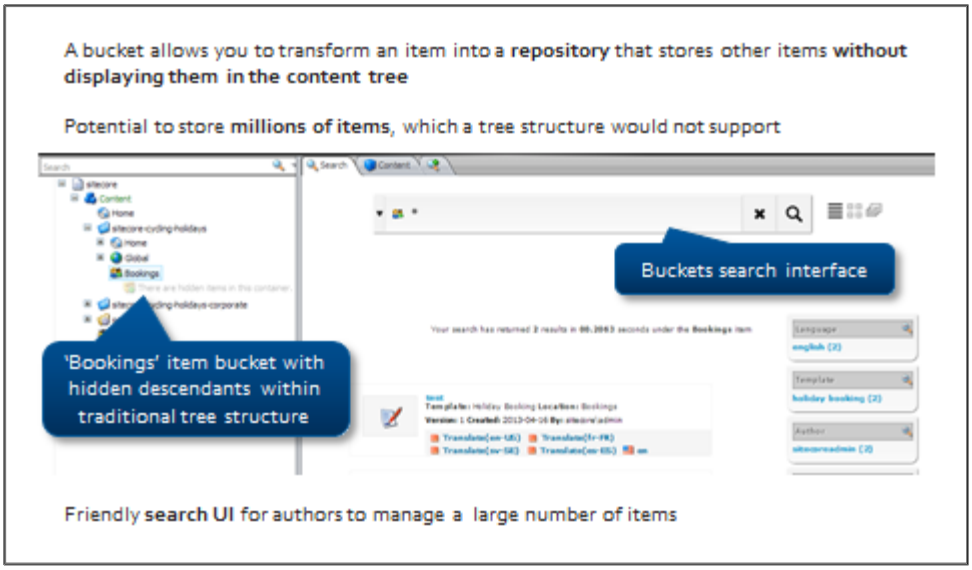
6. Click a search result to open it. Duplicate it to make sure that the functionality is normal. Click the **Search** button to refresh the search and the duplicated item should appear.

   Note the list of filters in the right column on the screen. These are known as *result facets* (for example: by default, Sitecore facets on **Template Name** and **Language**).

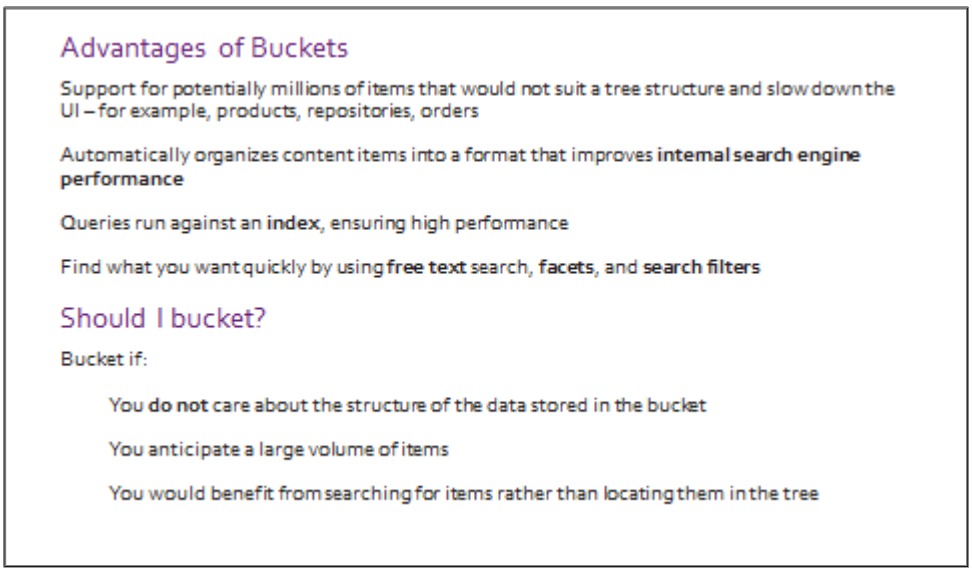
7. In our results list, we have a **Holiday Booked** facet that has been created especially for Traincore to narrow down booking results by holiday package booked. Click **the arrow** on the left-hand side of the search box. Notice that **search filters** and **bulk search operations** become available to you:



What Is a Bucket?



When to Use a Bucket



Tip

To test functionality and performance, you can use the FillDB page found in the admin folder to quickly create large amounts of test data, to test the buckets functionality with sample URL: http://TrainCore/sitecore/admin/filldb.aspx

Demo – Buckets and Bucketable Items

In the following demo, we will:

- Turn reusable content into a bucket.
- Make General Widget items bucketable.
- Change bucket settings.

1. In the **Content Editor**, select the **Reusable Content** item and click the **Configure** tab.
2. Click the **Bucket** button. The options in the Buckets chunk should have changed to **Revert** and **Sync**, and a **Search** tab will appear in the content pane.
3. This item is now a bucket. If you want all items based on this template to be buckets, set it on **standard values.**
4. Navigate to the standard values item of the data template. Click the **Configure > Bucket** option.
5. Create a new item based on this data template. The bucket icon should appear in the gutter next to it, denoting that it is a bucket.
6. **Enable** standard fields, and scroll to the **Item Buckets** field section. Additional options include limiting which views are allowed for the item (for example, **Gallery** View for image heavy items), or choosing to maintain parent/child relationships when bucketing a pre-existing sub-tree of items.
7. Navigate to the reusable content item and try searching for *. Although you see results, if you expand the reusable content item, its children all appear as normal items because the items have not been made **bucketable.**
8. Select one of the **children** of the reusable content item. Notice the message at the top of the item:

9. Navigate to the item's data template and create a standard values item if one does not exist.
10. On the data template's standard values, find the **Item Buckets** field section, and select the **Bucketable** checkbox.
11. **Save** the item and navigate back to the bucket.
12. Because you have made changes pertaining to the bucket-ability of items in the reusable content bucket, you must sync it. **Note** that you do not need to sync a bucket after making content changes.

    On the **Configure** tab, click the **Sync** button. The bucket's children disappear.
13. On the **View** tab, select the **Buckets** checkbox. Then expand the reusable content item. Note the items are organized by creation time; your searches actually run against an index of these items.

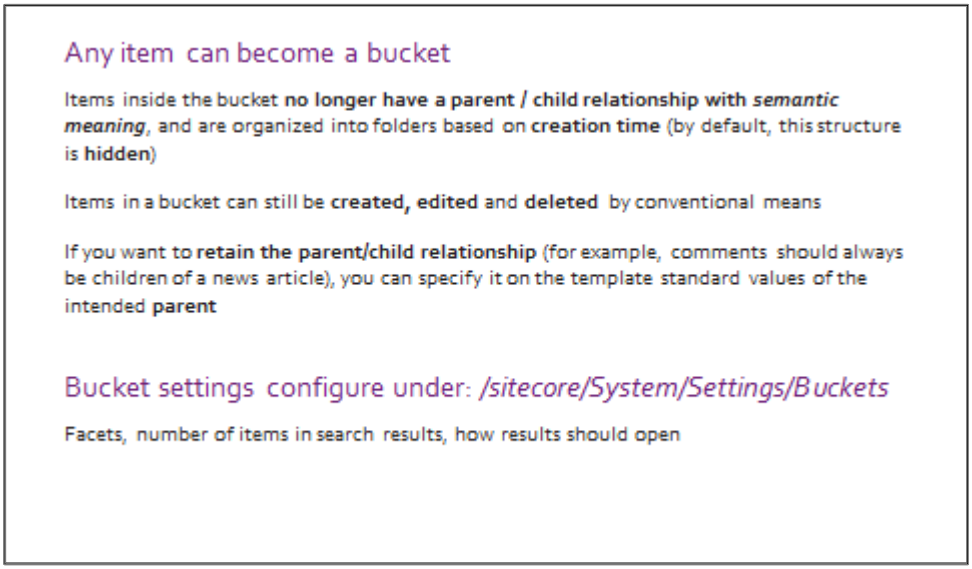
Note

Any item can become a bucket; items in buckets should be bucketable. You can mix non-bucketable items in, but they will not behave in the same way.

Tip

*You can make a particular template a bucket by default by selecting the template's standard values and clicking the **Bucket** command on the ribbon.*

Creating a Bucket



Tip

To preserve the parent/child relationship, navigate to the standard values of the template of the item that will act as the parent (for example: items based on the **News Article** template will be the parent to items based on the **Comment** template.). Ensure standard fields are visible and select the **Lock Child Relationship** checkbox.

Tip

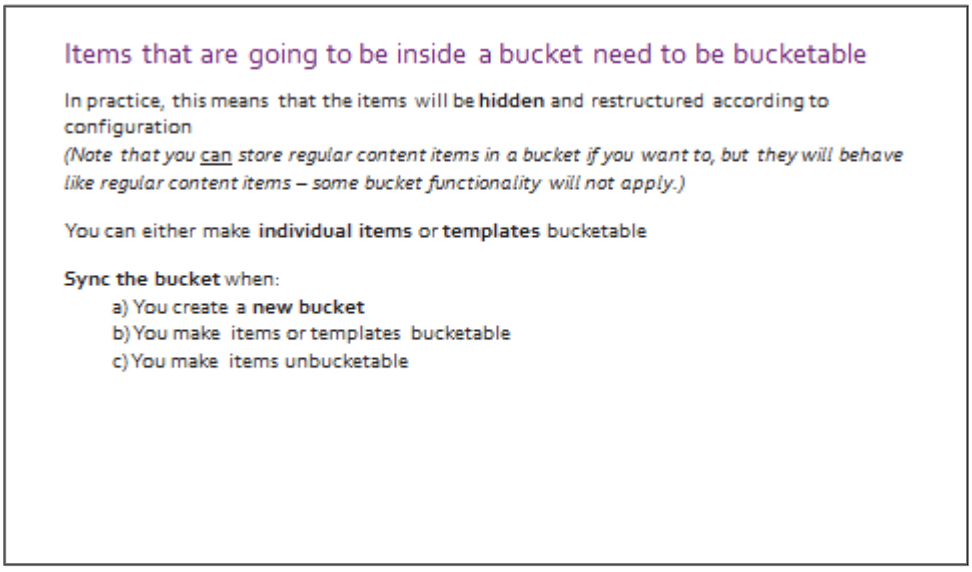
Specify a **Facet Filter class** to facet on only certain items – for example: you may want to exclude some languages.

Tip

To view the folder structure under a bucket, click **View > Buckets** option. This is configurable in Sitecore.Buckets.config:

```
<setting name="BucketConfiguration.BucketFolderPath" value="yyyy\/MM\/dd\/HH\/mm"/>
```

Making Items Bucketable



Demo – Configuring Facets

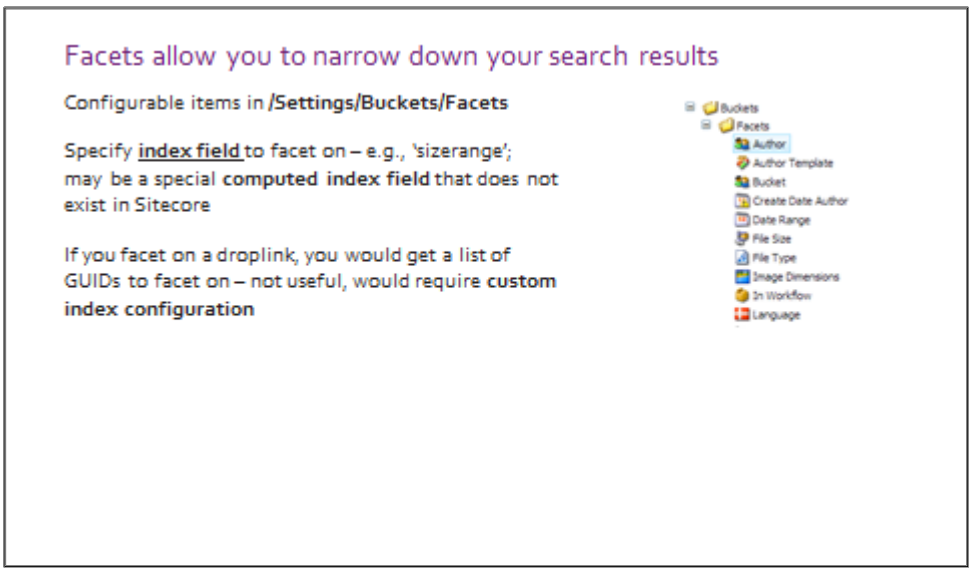
In the following demo, we will:

- View a number of pre-existing facets created by developers.

1. Using the Content Editor, navigate to: */System/Settings/Buckets/Facets.*
2. Navigate to the **File Type** facet. The field is **Extension**.
3. Navigate to a media item in the Media Library; one of the fields is called **Extension**. This is a very simple mapping between **Sitecore** field and **index** field.
4. Navigate to the **Language** facet. **Note** the **Field Name** field; this corresponds to a field in the index, not in Sitecore. Notice that items do not have a *language* field, even in raw values. This is calculated and entered into the index.

   Developers can extend the facet list, but it is not as simple as faceting on any field without doing some pre-processing. How do you facet on a pipe-delimited list of IDs?
5. Navigate to the **Holiday Name** facet. Notice that the field name here is **computedholidayname**. This is another **calculated** field. When the indexer looks at Sitecore items, it looks at every item and sees if it lives under a holiday item, then adds that holiday's name to this custom field.

Search Facets



Apply – Topic 8.1 – 20 min

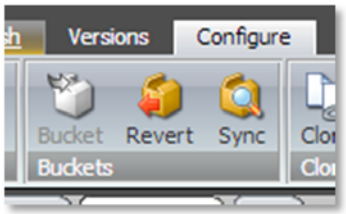
Create a Bucket and Make Items Bucketable

In the following labs, you will:

- Turn items based on the **News Listing** template into buckets.
- Make items based on the **News Article** template bucketable.

Lab A. Turn the News Listing Template into a Bucket

1. Using the **Content Editor**, navigate to the **News Listing** template: */sitecore/templates/BaseCore/Pages/News Listing*. Make sure that you are **viewing standard fields.**
2. Select the template's **standard values**; then click the **Configure** tab > **Buckets** chunk > **Bucket** command to turn it into a bucket.
3. Navigate back to an existing news listing in the content tree: */sitecore/content/sitecore-cycling-holidays/Home/news.*
4. Confirm that the item is now a bucket. The active commands in the Buckets chunk should be *Revert and Sync.*



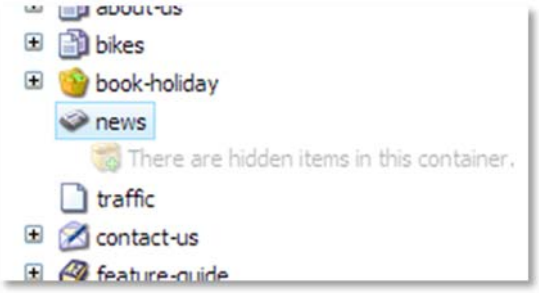
Knowledge Check

Expand the **News** item. Why are the news article items not hidden, even though the news item has been turned into a bucket?

Lab B. Make the News Article Template Bucketable

1. Using the **Content Editor,** navigate to the **News Article** template: */sitecore/templates/BaseCore /Pages/News Article.*
2. Select the template's **standard values** (make sure you are viewing standard fields).
3. Navigate to the **Item Buckets** field section and select the **Bucketable – Can be stored as an unstructured item in an item bucket** checkbox and **save.**

4. Because you have made a change to the template of an item that appears in a bucket, you must **sync** existing buckets (the existing **news** item).
5. Select the existing **news** item in the content tree.
6. Confirm that all children (except ***our-plans-for-the-new-year***, which is based on the Standard Content template) are hidden and that search and faceting work as expected.



Extend

- **Developer's Guide to Item Buckets and Search**
  http://sdn.sitecore.net/Reference/Sitecore%207/Developers%20Guide%20to%20Item%20Buckets%20and%20Search.aspx

Topic 8.2 Search

Introduction

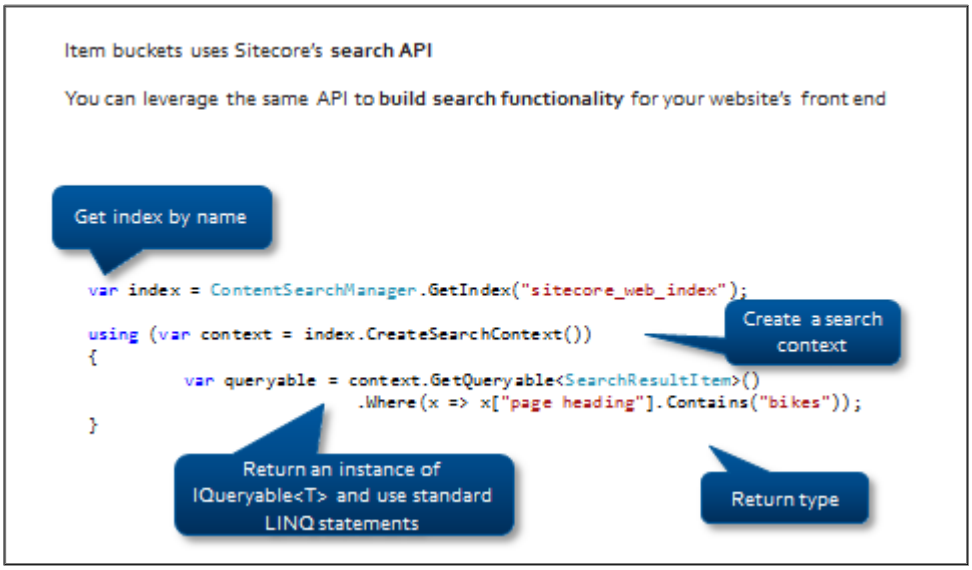
Objectives

By the end of this topic you will be able to:

- Search an index using a LINQ-based search API.
- Create a custom search result type.
- Configure what goes into your indexes, and how they are stored.
- Facet on a set of results.

Content

Simple Search



Demo – Executing a Simple Search

In the following demo, we will:

- Build a simple search and return **all** indexed items.
- Search by the item's *Name property* (Sitecore property).
- Search by item's *Heading* (developer-created property).

1. Drop the simple search code from the preceding slides into a sublayout code-behind file.
2. Attach a break point and note that an IEnumerable<T> is returned. This contains all indexed items.
3. Use .Where() to search for item's whose name contains *bike*:

Code Sample – Sitecore Property

```
.Where(x => x.Name.Contains("bike"));
```

4. Use .Where() to search for the items whose *main content* field contains *bike*:

Code Sample – Developer-Created Property

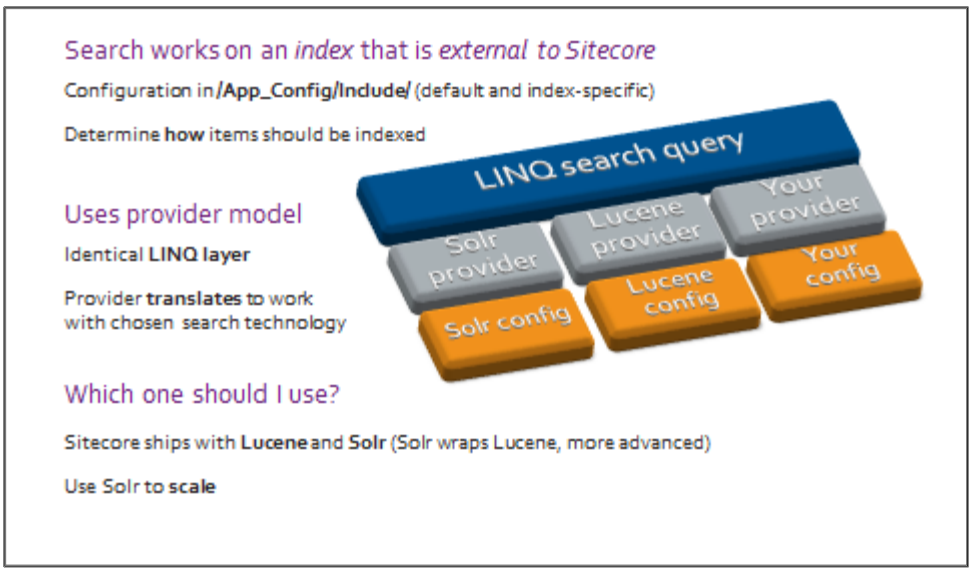
```
.Where(x => x["main_content"].Contains("bike"));
```

5. Use .Where() to search for items whose create date is within the date specified. (**Note** that .*Between* is a Sitecore-specific extension):

Code Sample – Using 'Between'

```
.Where(x => x.CreatedDate.Between(new DateTime(2013, 06, 03), DateTime.Now, Inclusion.Both));
```

Basic Index Configuration



Tip

Use **SwitchOnRebuildLuceneIndex** or **SwitchOnRebuildSolrIndex** to avoid an index directory being deleted while it is in the process of being rebuilt.

http://sdn.sitecore.net/upload/sitecore7/70/sitecore_search_and_indexing_guide_sc70-a4.pdf

Tip

Lucene and **Solr** are search platforms. They are built on the same technology, and the one you choose depends on your requirements. Lucene is best used for integrated, locally managed search, while Solr offers a scalable, enterprise-level solution.

Tip

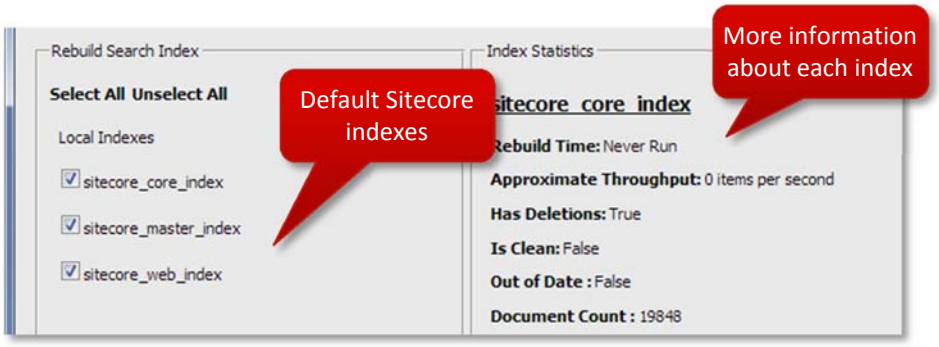
Solr supports distributed indexing, but Lucene does not. This means the search indexes can be offloaded to a separate server and then accessed by all CD servers.

Walkthrough – Index Viewer

In the following walkthrough, we will:

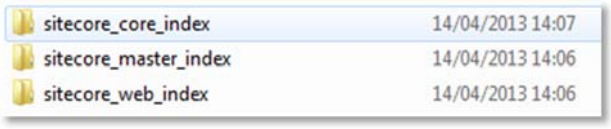
- Use the Index Viewer on the Sitecore Desktop to rebuild the indexes.

1. Using the **Sitecore Desktop**, click *Start* > *Control Panel* > *Indexing* > *Indexing Manage*.
2. Click the **Rebuild** button.

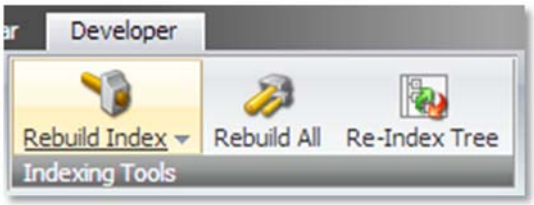


Notice that there is an additional option for Solr: by default, Sitecore defines three indexes – one per database (Master, Core, and Web).

3. Using the file system, navigate to the */Data/indexes folder*. You will see one folder per index.



4. Alternatively, you can enable to **Developer** tab in the Content Editor and use the **Rebuild Index** button there:



Tip

To view raw Lucene indexes, use a free open source tool called Luke: http://code.google.com/p/luke. Luke can also be used on Content Delivery networks to view an index however it can take a write lock on the indexes. Use this for viewing in read-only mode. Also be careful if you are opening a large index on a delivery environment because Luke will load it all into memory. Lastly Luke can open indexes created by different versions of the Lucene library and even the indexes created by Solr.

LINQ Examples

The API implements the **IQueryable<T>** interface and supports **most** options

Standard LINQ	Custom extensions
.Where	.Boost (make this part more important)
.Any	.Page (does skip/take)
.OrderBy	.Between (exclude/include start and end ranges)
.Filter	

Fuzzy search

```
queryable.Where(x => (x["page heading"].Like("Citecoar", 0.8));
```

Pagination

```
queryable.Page(2, 50);
```

Tip

.Filter() and **.Where()** both restrict the result list (and can be used in combination), but .Filter does not affect the scoring/ranking of the search hits.

Custom Search Result Type

You create your own search result type

Map **Sitecore** fields to **index** fields

Empty constructor, public properties

Support for various property types, *not* just strings! (see **Student Workbook**)

Can inherit from SearchResultItem

Standard .NET class

```
public class CustomResultItem : SearchResultItem
{
    [IndexField("page heading")]
    public string Heading { get; set; }
}
```

Heading property mapped to 'page heading' field

```
queryable.Where(x => (x.Heading.Like("Bike", 0.8));
```

Query using properties

Tip

Sitecore provides a starting point for a search result item in: Sitecore.ContentSearch.SearchTypes.SearchResultItem. Inheriting gives you properties like: .Template, .Language, .ID, and so on.

The **.Paths** property on SearchResultItem lets you narrow your search by path in the Sitecore tree.

Tip

Some standard fields are already indexed and stored (see Advanced Configuration). If you want to add a standard field as a property to your result item, use the properties in Sitecore.Search.BuiltinFields when defining the IndexField. This way you can avoid having to remember standard field names.

Examples of supported search result property types	
.NET built-in floating-point types	DateTime
String	GUID
Sitecore ID	Sitecore Short ID

Demo – Paths, Pagination and Search Result Types

In the following demo, we will:

- Create a custom search type called **CustomSearchResult** - include a number of different fields from base templates.
- Inherit from Sitecore's base **SearchResultItem** class.
- Refactor search to query properties rather than strings.
- Limit result set using .Paths.
- Paginate results.

1. Create a custom search type called CustomSearchResult. Include a number of different fields from the Base template, such as *Page Heading*.
2. Inherit from Sitecore's base SearchResultItem class.
3. Fields with a **space** need the IndexField decoration, for example: [IndexField("page heading")].
4. Refactor simple search to use your new custom class. You will now be able to query by properties rather than string fields.
5. Use .Paths property inherited from SearchResultItem to restrict the search location.
6. Use .Page() to paginate result set.

Returning a Result Set

```
A query returns an IQueryable<T>
var query = context.GetQueryable<SearchResultItem>();
int count = results.Count();

Sitecore gives you a .GetResults() item that contains results and count

var results = context.GetQueryable<SearchResultItem>()
                     .GetResults();

var facets = results.Facets.Categories;
int total = results.TotalSearchResults;
var list = results.Hits.Select(x => x.Document);
```

Tip – Advanced Use of Facets

For a more advanced example of how faces can be used, download the Autohaus demo from the Sitecore Marketplace: https://marketplace.sitecore.net/en/Modules/Autohaus.aspx

Best Practice

When you output a list of results, you can iterate through the list of SearchHit object and use the values in the .Document to avoid having to hit the database for item information. However, this will only work if your field is stored rather than just indexed.

Facets

```
Use .FacetOn() to get a list of facet categories (e.g. Template Name)

var results = context.GetQueryable<SearchResultItem>()
                        .FacetOn(x => x.TemplateName)
                        .GetResults();


Each category has a number of values, with a Name (e.g. News Article)
and an Aggregate (e.g. 10)

var facetCategories = results.Facets.Categories;


Feed back into a .Filter()
```

Tip - Security

Huge improvements to security are being made in later updates to Sitecore 7. Until then, refer to Stack Overflow for options: http://stackoverflow.com/questions/16683487/indexing-sitecore-item-security-and-restricting-returned-search-results.

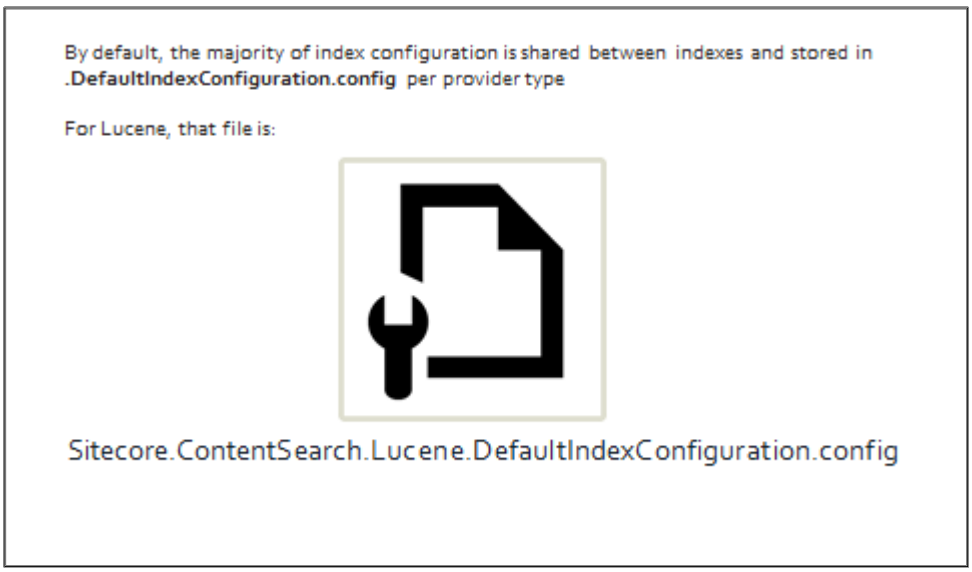
Demo – GetResults() and Facets

In the following demo, we will:

- Return the SearchResults object.
- Set count, hits, and facets to variables and inspect.

1. Use the **GetResults()** method to return a results object rather than an IEnumerable.
2. Return facet categories, a list of **total search results** (regardless of pagination, this saves you from having to do multiple queries), and a collection of hits.

Index Configuration

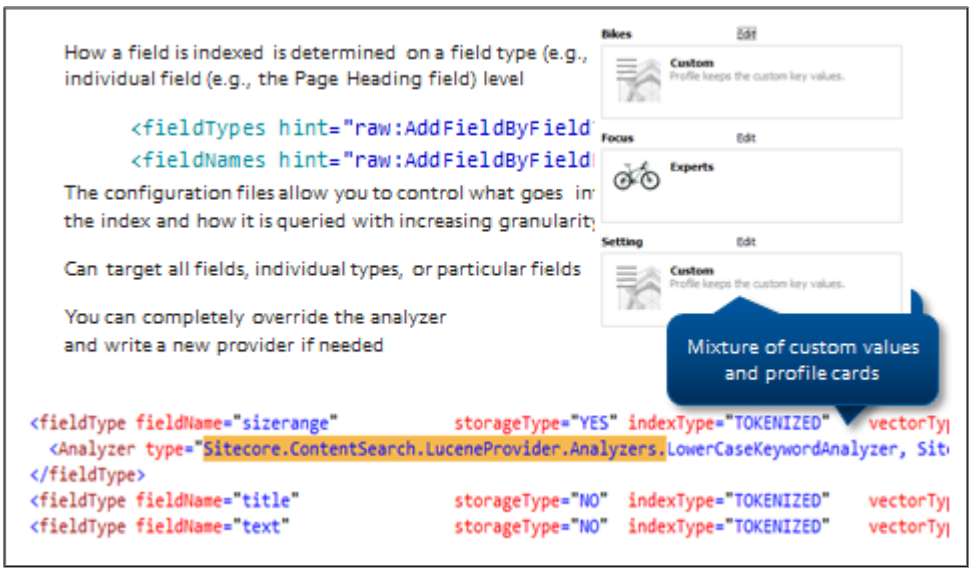


Which Fields Should Be Indexed?

Important

It is essential that you fine-tune your index to include only the fields that you require. Many unnecessary standard fields are excluded by default. As your solution scales, an index that has not been tuned will become difficult to manage and take a long time to generate.

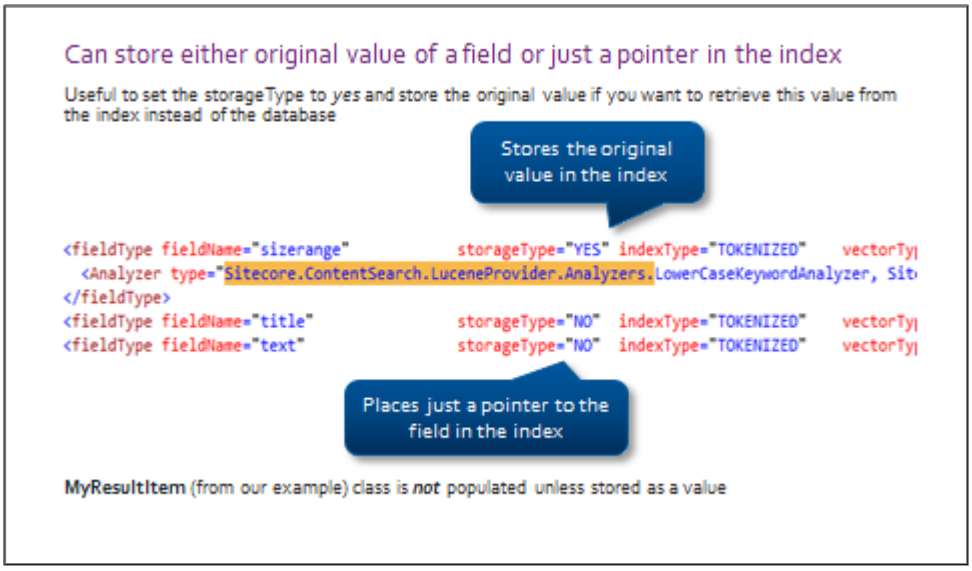
How Should Fields Be Indexed?



Tip
The DefaultIndexConfiguration.config contains comments that explain what each attribute does and what the options are (for example, indexType, vectorType).

Tip
You can index media items (and their contents in the case of PDFs) as well – see blog post for more information:
http://www.sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog/Posts/2013/04/Sitecore-7-Indexing-Media-with-IFilters.aspx

Store or Index?



Computed Fields

Tip
You can also use Virtual Fields that evaluate on request. They are useful if you need to perform a calculation on the fly rather than retrieve a stored value.

Tip
Sitecore ships with a tool that allows you to test the scalability of your search providers and the way your indexes have been set up. For more information, see *The Developer's Guide to Item Buckets and Search* (previous topics Extend).

Apply – Topic 8.2 – 70 min

Refactor News Listing

Currently, we are outputting a list of news articles using the .GetDescendants() method. In the following labs, you will:

- Refactor the news listing code to retrieve data from an index using the search API.

Lab A. Output News Articles Using Sitecore Search API

1. In **Visual Studio**, open the **basecore-news-listing.ascx.cs** file (under */layouts/BaseCore/content/basecore-news-listing.ascx.cs*).
2. In order to take advantage of the **search API**, make sure you include the following namespaces:

Code Sample – Sitecore Property

```
using Sitecore.ContentSearch;
using Sitecore.ContentSearch.SearchTypes;
using Sitecore.ContentSearch.Linq;
```

3. Using **Visual Studio**, create a new class called ***NewsResult***. Either create it in /layouts/BaseCore or in a new folder under BaseCore in the Training.Utilities project; this is your custom search result class.
4. **Inherit** from the Sitecore.ContentSearch.SearchTypes.SearchResultItem base class.

5. Add the properties specified in the following table:

Property name	Property decoration
PageHeading	[IndexField("page heading")]
PageSummary	[IndexField("page summary")]

6. Use the code example below as a guide:

Code Sample – Search Class Property

```
[IndexField("meta description")]
public string MetaDescription { get; set; }
```

Tip

If you want to include a system field in your custom search result class, consider using the BuiltinFields namespace to decorate the properties rather than typing the field name out as a string:

```
[IndexField(BuiltinFields.Version)]
string Version { get; set; }
```

7. Using **Visual Studio,** open the **news listing sublayout code-behind** file (.cs).

   Previously, news articles were retrieved using **.GetDescendants()**. In this refactor, we want to produce exactly the same result by querying an index. A partially completed code sample has been provided in the **student resource folder** (*WND Labs > Module 8 > Topic 8.2 > Lab A > basecore-news-listing.ascx.cs)*. We recommend that you refer back to the materials on how to construct a query.

8. **Retrieve the web index** – either by retrieving it **by name** (sitecore_web_index), or by using a SitecoreIndexableItem. (This object is constructed by passing a Sitecore item in as a parameter). See the code example below. For ease, retrieve the index **by name**.

Code Sample – Index by Name

```
Sitecore.ContentSearch.ContentSearchManager.GetIndex("sitecore_web_index");
```

Code Sample – SitecoreIndexableItem

```
SitecoreIndexableItem indexableItem = new SitecoreIndexableItem(item);


var index = Sitecore.ContentSearch.ContentSearchManager.GetIndex(item);
```

9. **Create** a **search context** from that index.
10. **Use** the **GetQueryable()** method on the search context object to **retrieve** an **IQueryable.**
11. **Use LINQ** statements to **return** only items that are **descendants** of the **current news listing and** are **based on** the **News Article** template.

Sitecore API .Paths
To limit the section of the content tree that results are returned from, use the Paths property on the base class; this will return only items that have a particular ID in their list of ancestors (for example, **Home** would have the ID for the **Content** and **sitecore** items):

x.Paths.Contains(item.ID)

12. **Order by** the **CreatedDate** property (found on the SitecoreSearchResult base class).
13. At the end of the LINQ statement, use the **GetResults()** method to return a SearchResults object. This should come after any .Where() and .OrderBy(). The following example shows a query with a single .Where() using .GetResults():

Code Sample – Search Class Property

```
var query = context.GetQueryable<SearchResultItem>()
                .Where(x => x.Name == "Bikes")
                .GetResults();
```

14. The SearchResults object has a property called **.Hits**. It is a collection of SearchHit objects, each of which has a **.Document** property. The .Document property is a NewsResult object.
15. Select out a list of NewsResult objects:

Code Sample – Search Class Property

```
.GetResults();results.Hits.Select(x => x.Document)
```

16. Bind results to a repeater, delete the existing repeater and use the sample provided in the **student resource folder** (*WND Labs > Module 8 > Topic 8.2 > Lab B > basecore-news-listing.ascx).*
17. Make sure you **change** the **ItemType attribute** on the **repeater** to your **custom results class.**
18. Use the format in the file to output properties of your objects. Notice the colon after the # symbol**. All data should come from the index**; you should not need to retrieve the item from Sitecore – including the **.URL** property.

Important

As of Update-2, the .URL property requires some modification. Ask your instructor for more information.

Knowledge Check

At this point, all properties that are retrieving not standard fields will be blank. Why is this? Hint: think about the storage type configuration.

---

19. In the Visual Studio solution, open */App_Config/Include/Basecore/Basecore.Search.config*.
20. In the **raw:AddFieldByFieldName** section, ensure that **Page Heading**, and **Page Summary** are listed.
21. Make sure the **storageType** attribute is set to **YES.**

22. In the **Content Editor**, use the **Developer** tab to **rebuild** the **web index**. (If you have any unpublished news articles, make sure you publish these items first. Otherwise, they will not be added to the web index).
23. **Save** and **deploy** the solution.
24. Navigate to the News Listing page on the site and confirm that news articles are being output **in date order.**

Tip
If you are seeing more than one version of an item, it might be because you have more than one language version of each specified. Narrow it down by adding a .Where() to your query that only returns items from the context language (ordinarily found in the Regional ISO Code field on the language definition item, but for the purposes of this exercise you can use the item name to save time).

Extend

- **Sitecore Search and Indexing Guide**
  http://sdn.sitecore.net/Reference/Sitecore%207/Sitecore%20Search%20and%20Indexing%20Guide.aspx
- **Search Scaling Guide**
  http://sdn.sitecore.net/Reference/Sitecore%207/Sitecore%20Search%20and%20CMS%20Scaling%20Guide.aspx
- **Sitecore 7 Development Team Blog**
  http://www.sitecore.net/Community/Technical-Blogs/Sitecore-7-Development-Team.aspx
- **Sitecore 7 Hangouts**
  http://www.youtube.com/user/SitecorePM

Module 9
Recommended Practices

Contents:

Topic 9.1 Working with Media

Introduction

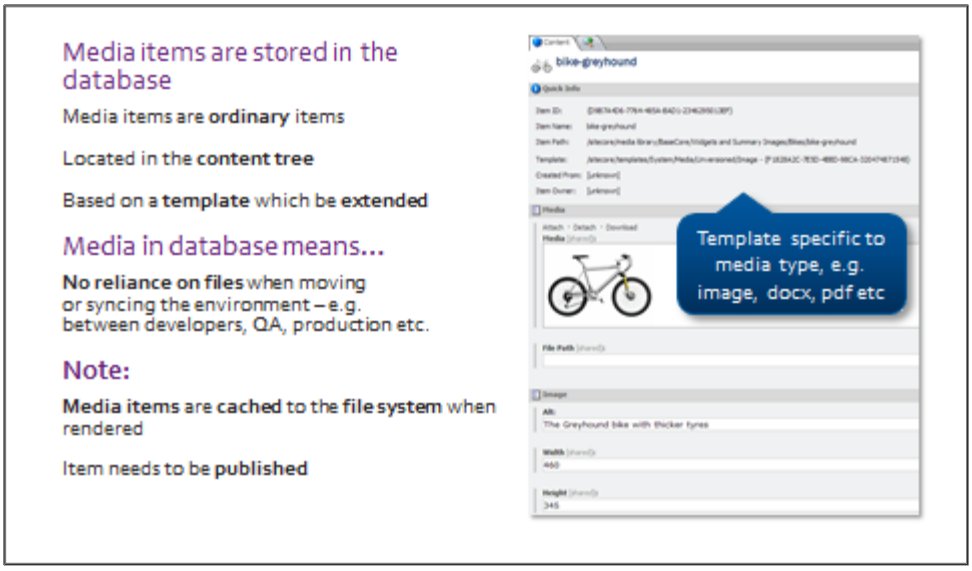
Objectives

By the end of this topic you will be able to:

- Understand how media gets stored.
- Define what settings can be changed in the Web.Config relating to media items.
- Describe how to change media extensions from .ashx to the true file ending.

Content

Storing Media

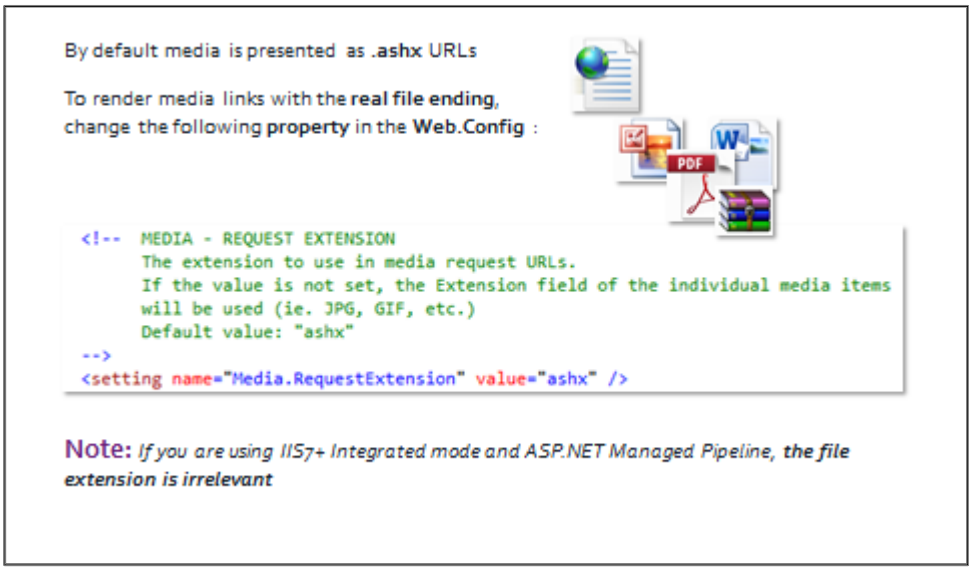


Tip

There is a file upload folder on in the website folder. If you enable this folder in the web.config, it will monitor media items dropped there. If files do get added, then they appear in the media library. Keep in mind when you deploy your solution that if you are storing media files on the file system, they need to be replicated to your new environment. We recommend that you store media in the database for that reason. However, it is useful for storing large files (video library) on disk because that way you don't bloat the database and the media database files (MDF) remain lean. Storing media in the database sometimes can be inefficient and, depending on the size, might actually hurt performance.

Mapping

File Extensions



Walkthrough – Changing File Extensions

In the following walkthrough, we will:

- Change the file extension in configuration.

1. Go to **Traincore.**
2. Inspect the element for an image - **.ashx**
3. Change the Media.RequestExtension value in the **web.config**.
4. Inspect the element again. The true extension is revealed.

Topic 9.2 Caching and Performance

Introduction

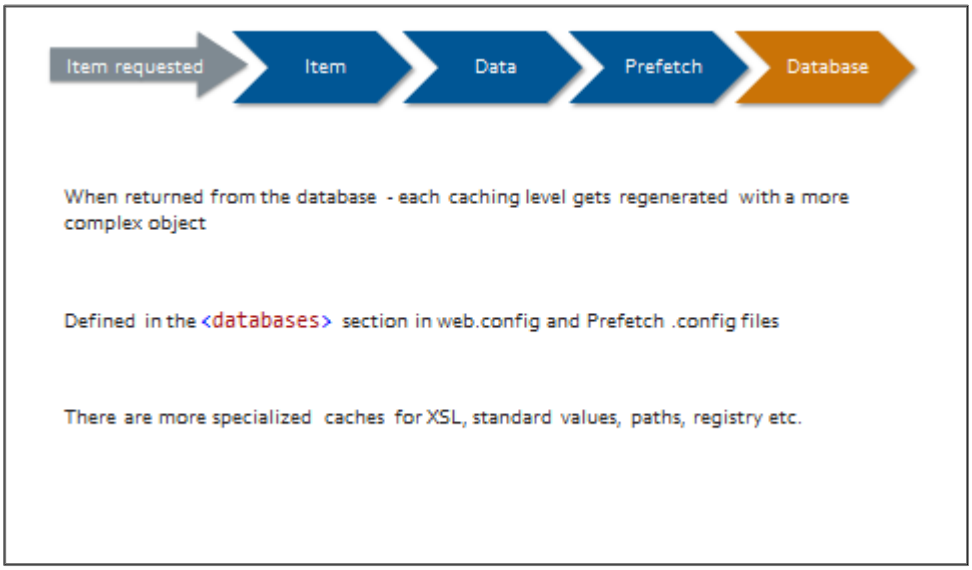
Objectives

By the end of this topic you will be able to:

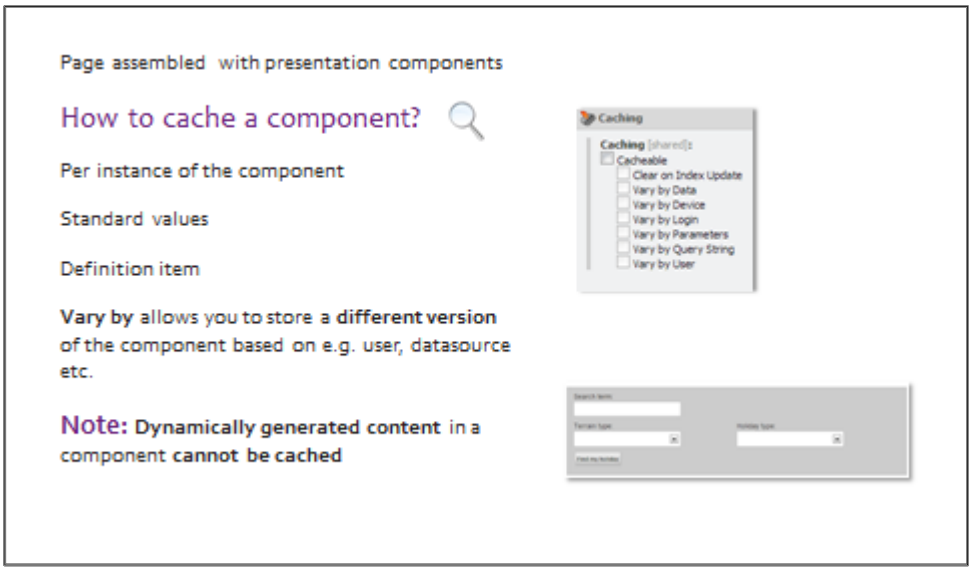
- List the item cache layers.
- Name an action that clears HTML cache.
- Discuss prefetch cache.
- Discuss where you can clear all caches.
- State where cache settings are defined.
- Describe various caching options.
- Explain the purpose of Profile and Debug mode in the Page Editor.

Content

Item Cache



HTML Cache

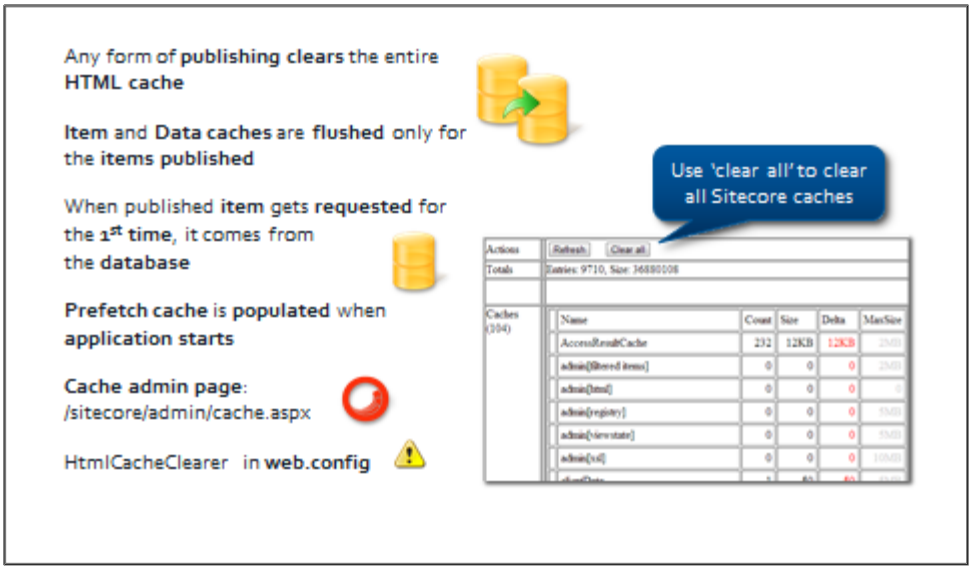


Tip

HTML caching is defined on individual components. This is another reason to make sure you componentize your page.

Caching by parameters/user/data means that a version of your component can be cached unless those parameters change. You might want to cache a component for all users in the US, but cache is different HTML for UK users. Alternatively, if your component accepts various datasources, you can cache that component with its different datasource, which is then added to the index of the cache. Then when the component is requested again, it is pulled from the cache with its relevant datasource.

Clearing Cache



Important

If you have a separate site in the web.config, make sure the name of the site is listed as a site in the HtmlCacheClearer event handler in the web.config:

```
<event name="publish:end">
  <handler type="Sitecore.Publishing.HtmlCacheClearer, Sitecore.Kernel" method="ClearCache">
    <sites hint="list">
      <site>website</site>
    </sites>
  </handler>
</event>
```

Walkthrough – Clearing Cache

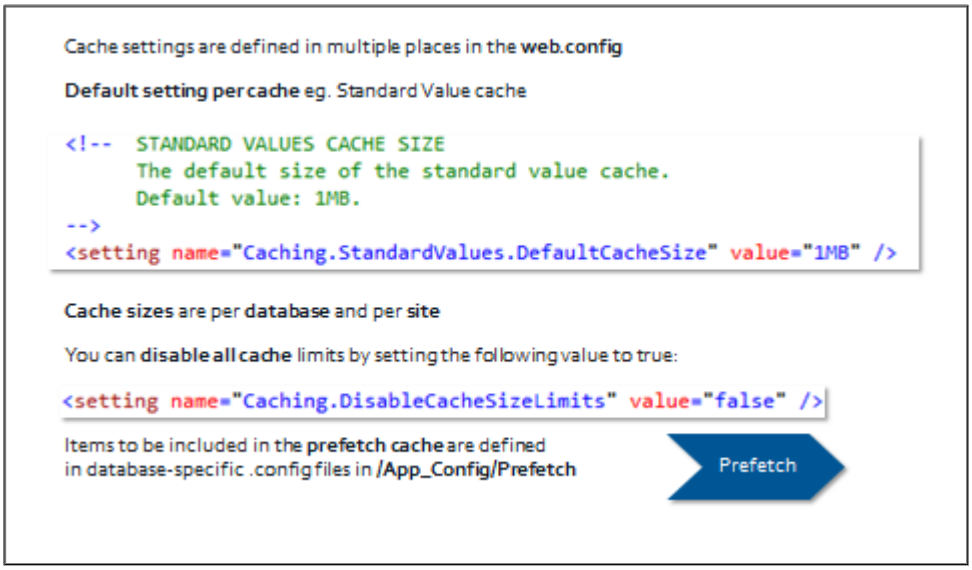
In the following walkthrough, we will:

- Navigate to http://traincore/sitecore/admin/cache.aspx.
- Notice maximum cache sizes are listed in the right-hand column.
- Demonstrate that caches can be cleared using the **Clear All** button.

Set up HTML cache for a component

1. Navigate to http://traincore/sitecore/admin/cache.aspx.
2. Notice that maximum cache sizes are listed in the column on the right side of the window.
3. Demonstrate that caches can be cleared using the **Clear all** button.

Cache Tuning and Configuration



Tip

See **cache reference guide** for more information:
http://sdn.sitecore.net/Reference/Sitecore%206/Cache%20Configuration%20Reference.aspx

Tip

Sitecore Rocks caches interface – using Sitecore Rocks, right-click on an instance and in Commandy, type *cache*. You get a handy interface for viewing and configuring different cache settings. For a mini tutorial see: http://www.sitecore.net/Community/Technical-Blogs/Trevor-Campbell/Posts/2013/02/28-Days-of-Sitecore-Rocks-Manage-Part-3.aspx

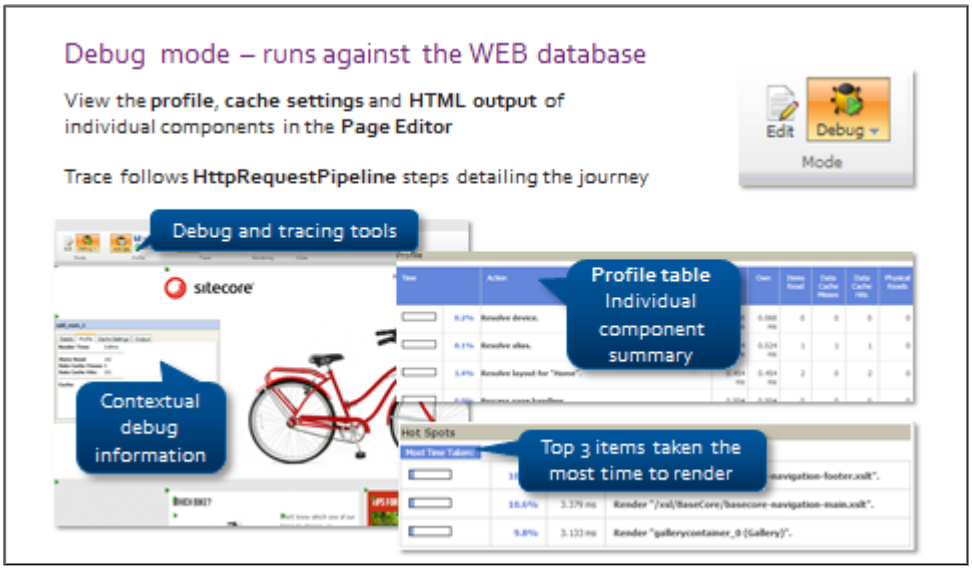
Demo – Various Cache Settings

In the following demo, we will:

- Look at the various cache settings available in the web.config.
- Look at various cache interfaces.

1. Look at the various cache settings in the web.config:
   - Default settings
   - Cache sizes specified in the <databases> section
   - Cache sizes specified on the <site> nodes or below the <sites> section
2. Look at various cache configuration files in */App_Config/Prefetch*.
3. Look at the statistics page.

Profiling and Debugging



Tip

Experience CMS Performance Tuning Guide:

http://sdn.sitecore.net/Reference/Sitecore%207/CMS%20Performance%20Tuning%20Guide.aspx

Experience CMS Diagnostics Guide:

http://sdn.sitecore.net/Reference/Sitecore%207/CMS%20Diagnostics%20Guide.aspx

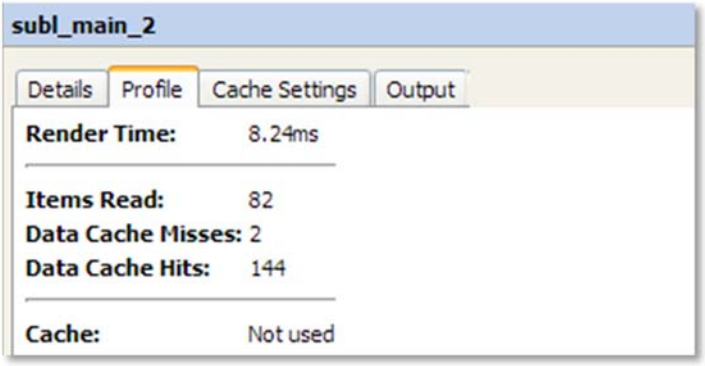
Walkthrough – Use Profiler and Debugger

In the following walkthrough, we will:

- Look at debug mode.
- Enable and disable profiling and tracing.
- Look at the profile table and trace.
- Visit */sitecore/admin/stats.aspx*.

1. Log in to the **Sitecore Desktop** and start the site in debug mode by clicking **Sitecore > Debug.** **Note** that the ribbon at the top of the page allows you to enable/disable profiling and tracing:

2. **Debug** mode allows you to see the profile, cache settings and output of individual components (Yet another reason to split your page into components that can be easily isolated and profiled):



3. The information about individual components is summarized in the profile table at the bottom of the page.
4. The bottom of the page also shows you the trace. Notice that the steps that are executed are the ones defined in the httpRequestPipeline (resolve site, resolve language, resolve device, and so on).
5. Aggregated statistics about component load times are also available on the ***/sitecore/admin/stats.aspx*** page. Total time represents the total time it has taken to load the control across all invocations.

Topic 9.3 Publishing

Introduction

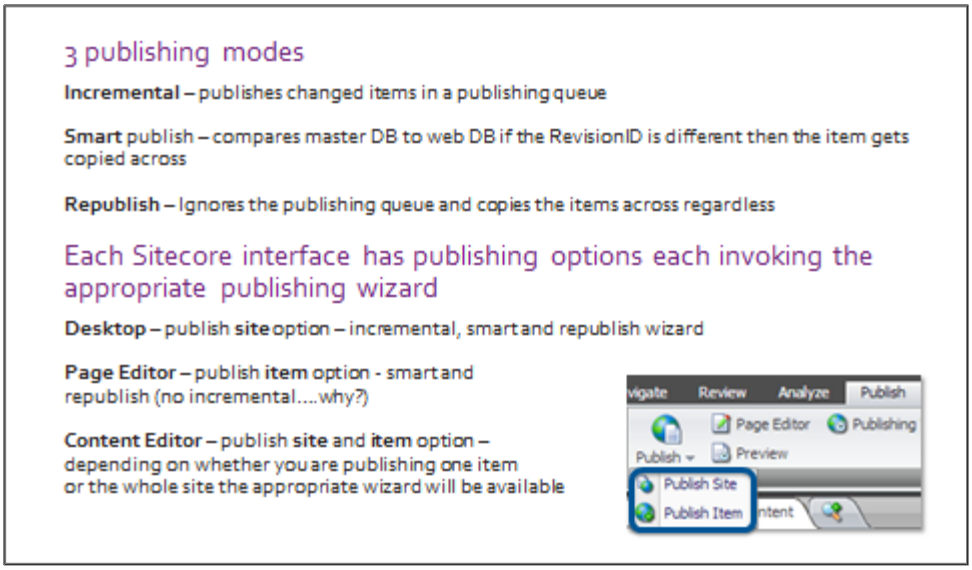
Objectives

By the end of this topic you will be able to:

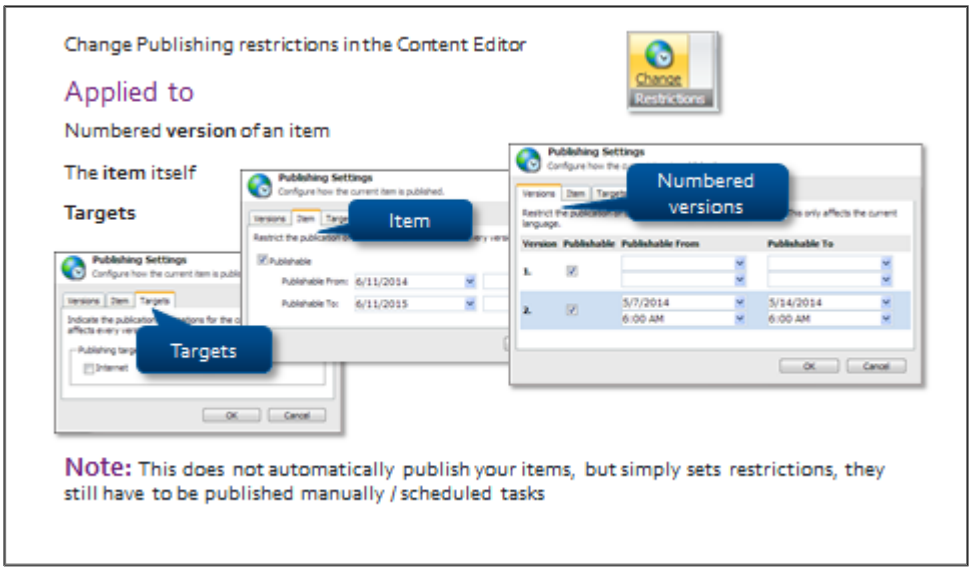
- Describe the different publishing modes.
- Name three elements that publishing restrictions apply to.
- Discuss recommended practices for publishing strategies.

Content

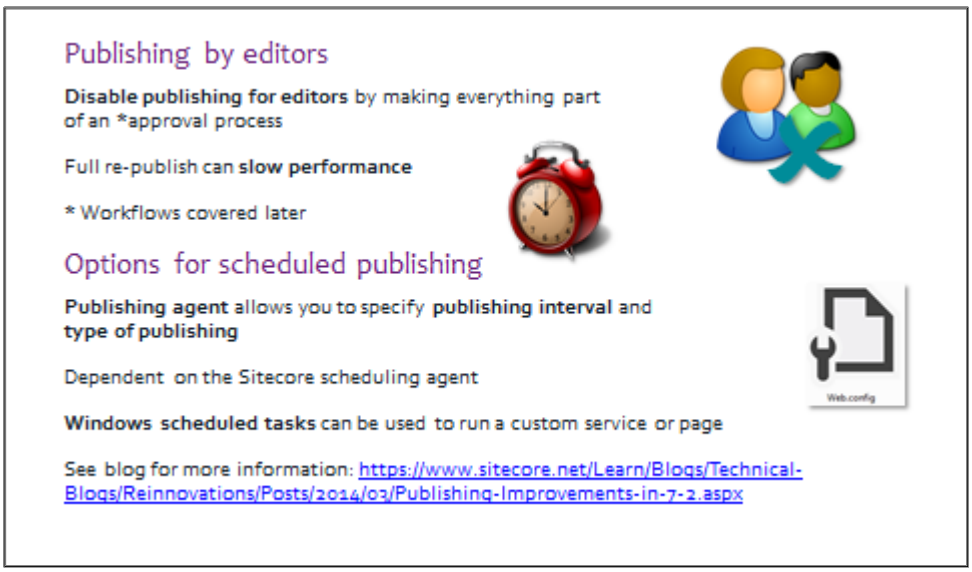
Types of Publishing



Publishing Restrictions



Publishing Strategies



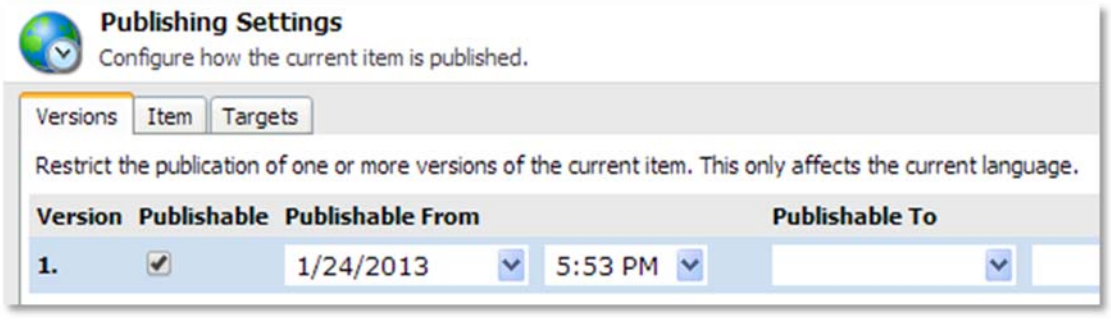
Demo – Publishing Options and Restrictions

In the following walkthrough, we will:

- Set up publishing restrictions using Sitecore Rocks.
- Set up publishing restrictions using the Content Editor.

1. In the Content Editor, select a content item such as **terms-and-conditions** under: */sitecore/sitecore-cycling-holidays/Home*.
2. On the ribbon, select **Publish tab > Change restrictions**:

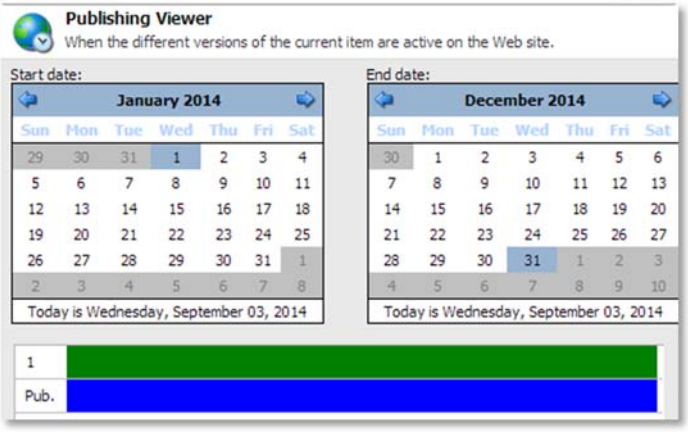
3. Look at the three tabs – Version, Item, and Target:



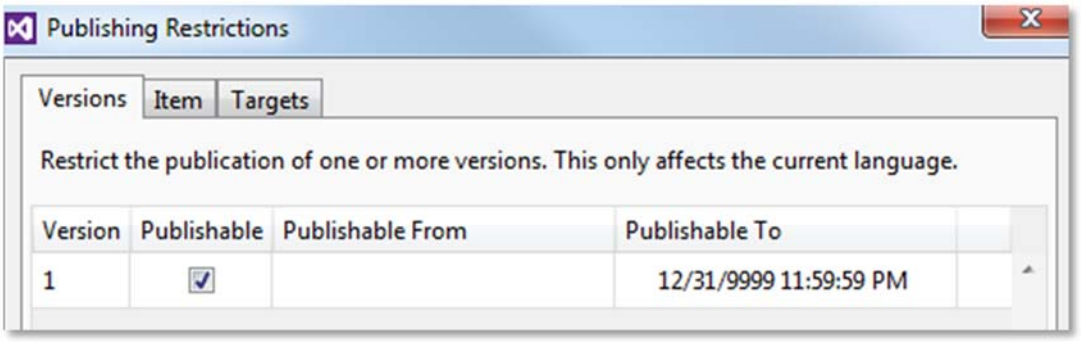
If you clear the ***Publishable*** checkbox on the **Versions** or **Item** tab, that item or version will be removed when the item is next published.

If you set a **Publishable From** and **Publishable To** date and time, the item or version will be *publishable* between those dates and times only.

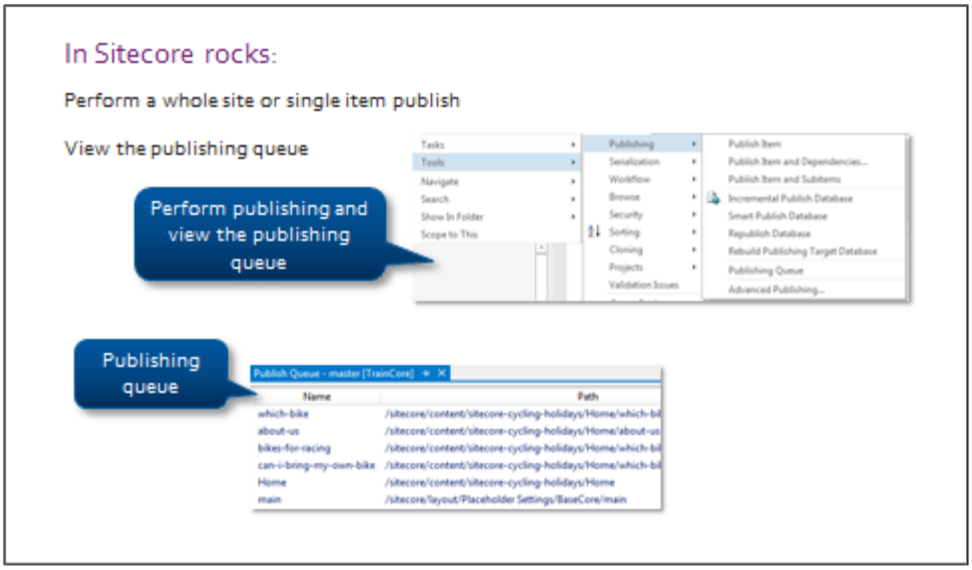
4. Go back to the **Content Editor**. On the **Publish tab,** click **Publishing Viewer**. This dialog shows a graph of when a particular version was available on the live site:



5. Switch to **Sitecore Rocks**. Right-click the item that you worked on in the Content Editor, and select **Tasks>Set Publishing Restrictions**. A similar interface to the one displayed in the Content Editor will appear:



Sitecore Rocks and Publishing



Tip

See the Content API Cookbook for information on how to publish an item using the API:
http://sdn.sitecore.net/upload/sitecore6/64/content_api_cookbook_sc64_and_later-a4.pdf

Extend

- **Publishing improvements in Sitecore 7.2**
  https://www.sitecore.net/Learn/Blogs/Technical-Blogs/Reinnovations/Posts/2014/03/Publishing-Improvements-in-7-2.aspx
- **Publishing through Sitecore Rocks**
  http://www.sitecore.net/unitedkingdom/Community/Technical-Blogs/Trevor-Campbell/Posts/2013/02/28-Days-of-Sitecore-Rocks-Publishing.aspx

Topic 9.4 Installing and Scaling Sitecore

Introduction

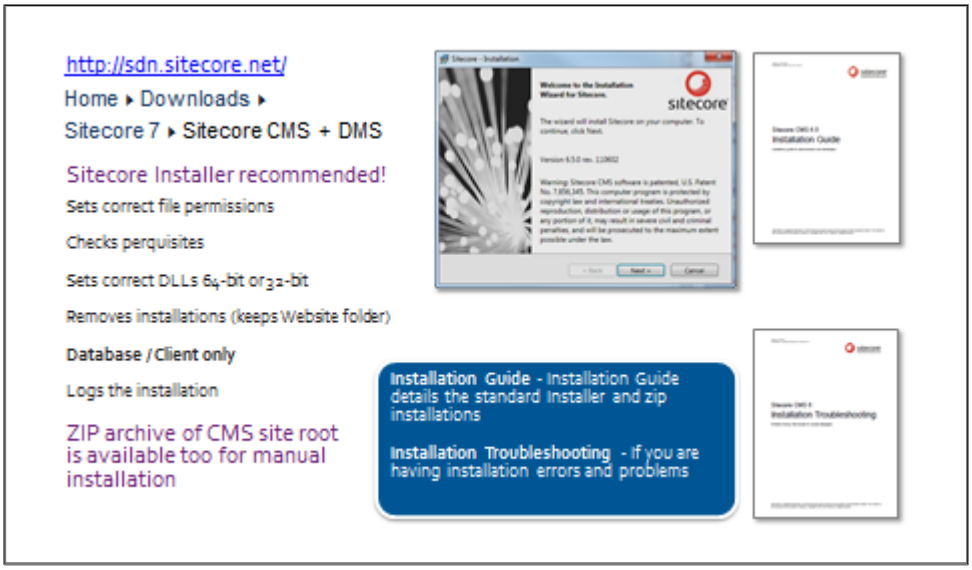
Objectives

By the end of this topic you will be able to:

- Discuss the installer's functionality.
- Name two tools that can be used to install Sitecore using scripts.
- Describe the three system components that get installed.
- Name three folders that get installed.
- Discuss what these folders contain.
- Draw out the production/live environment as recommended by Sitecore.
- Name a guide to help with scaling a solution.

Content

Installer or Manual .ZIP Files



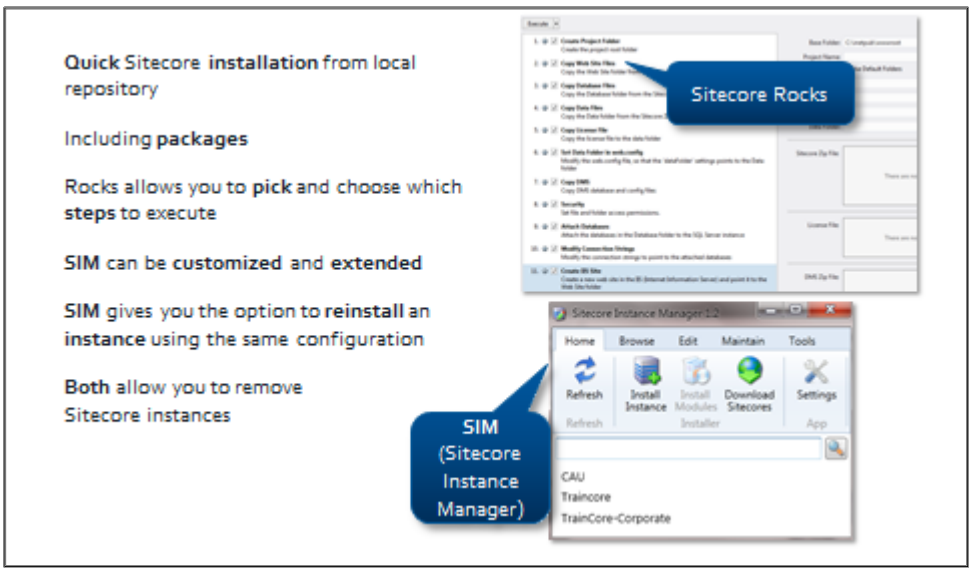
Tip

Using the installer, you can install the databases only on the DB server. The client installation must be performed separately, so you will run the installer twice.

Important

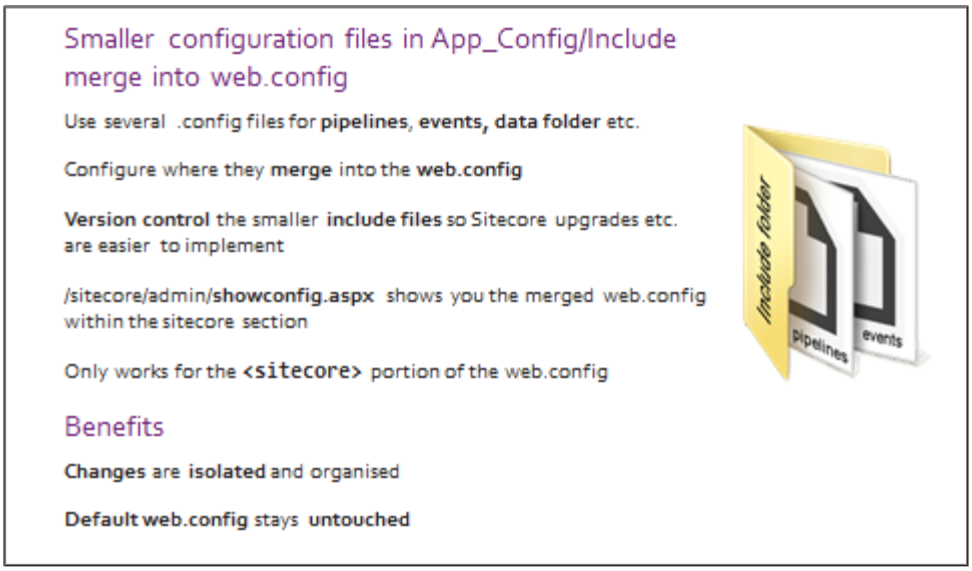
When you have used the .exe for installation, you should use add/remove programs to uninstall it. If you just delete the files and databases and so on, you will leave entries in your registry.

Scripts (SIM or Sitecore Rocks)

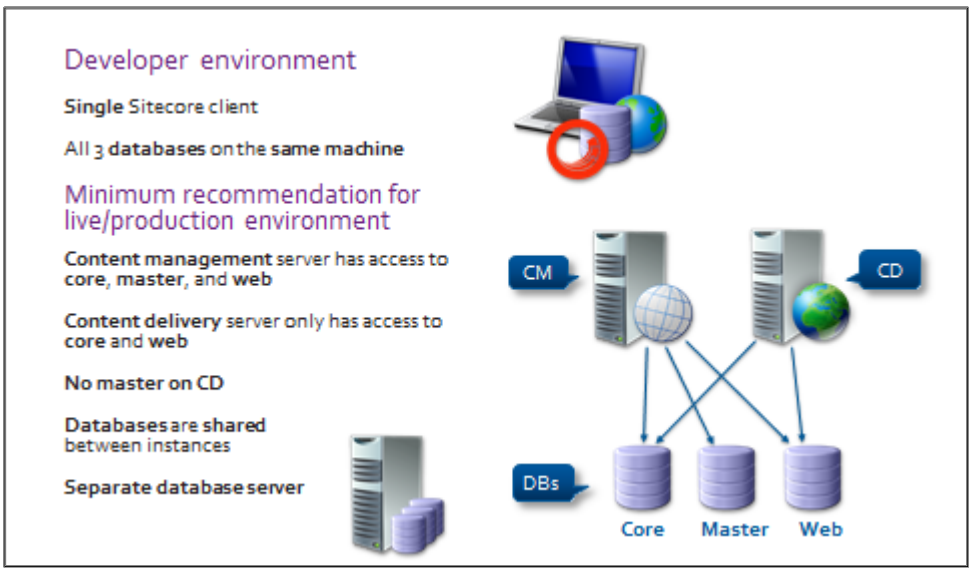


- **SIM download and documentation:** http://marketplace.sitecore.net/en/Modules/Sitecore_Instance_Manager.aspx
- **Sitecore Rocks:** http://marketplace.sitecore.net/Modules/Sitecore_Rocks.aspx

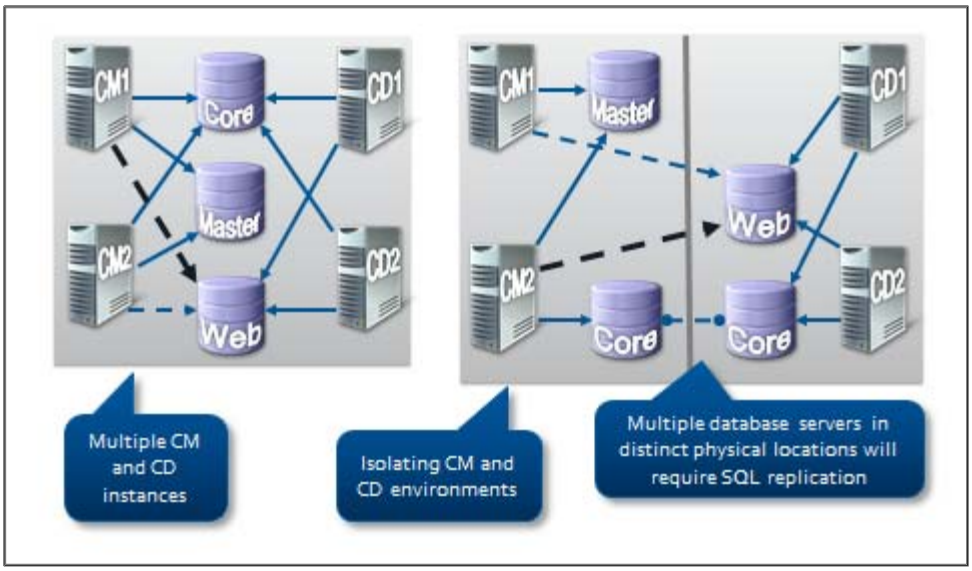
Sitecore's Include Folder



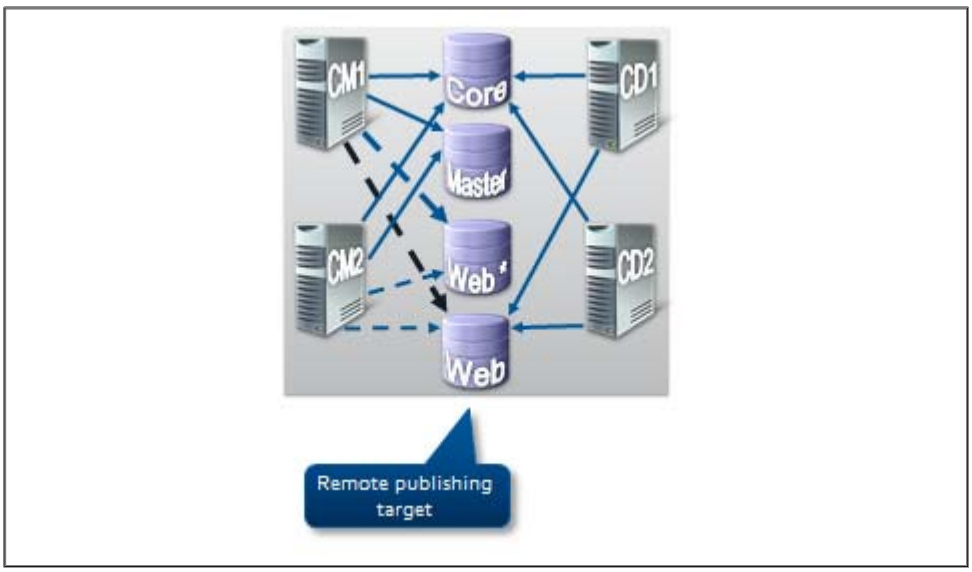
Scaling Infrastructure



Complex Scaling Scenarios



More Complex Scaling Scenarios



Important

Requirements depend 100% on the size of your solution, the expected traffic and on whether or not you are using personalization and analytics features.

Installation Guides

http://sdn.sitecore.net/Products/Sitecore%20V5/Sitecore%20CMS%206/Installation.aspx

Troubleshooting
http://sdn.sitecore.net/Products/Sitecore%20V5/Sitecore%20CMS%206/Installation%20Troubleshooting.aspx

Scaling guide
http://sdn.sitecore.net/Reference/Sitecore%206/Scaling%20Guide.aspx

Topic 9.5 Team Development and the Development Environment

Introduction

Objectives

By the end of this topic you will be able to:

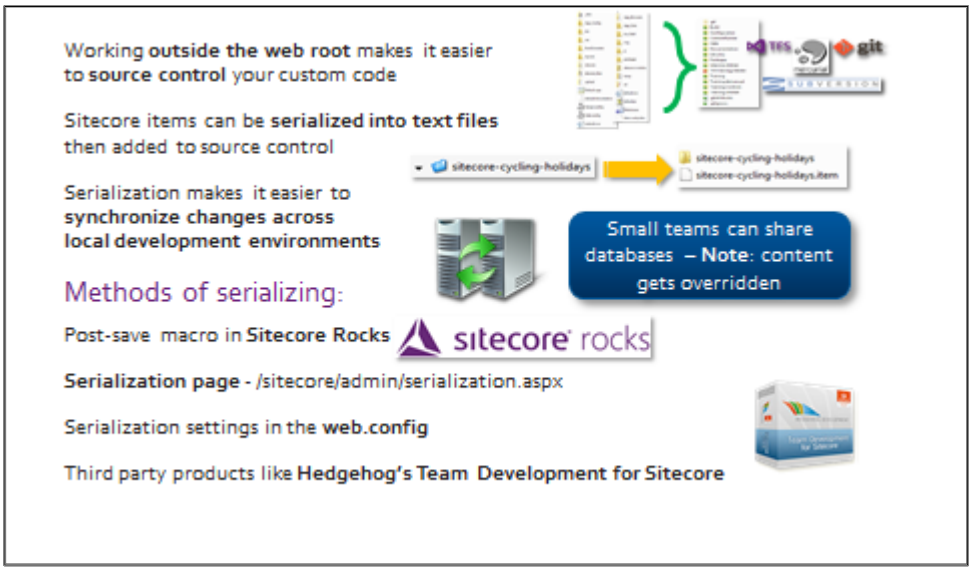
- List the two development environments and their uses.
- Describe how to sync your files and content across environments.
- Name three tools used for serialization.
- Describe packages.
- Name the ASPX used to for upgrading Sitecore.

Content

Setting Up a Solution

Development model	Benefits	Drawbacks
Inside the web root: 	• Solution setup is quick and straightforward • If your team is using the code-beside development model, you will be able to see your changes almost immediately	• No clear separation of ownership • Difficult to transform configuration files • Development sites may contain code-beside files or other unnecessary artifacts that may alter site behavior inadvertently
Outside the web root: 	• Clear separation between the solution files and Sitecore-owned files • Ability to transform configuration files within local developer environments without changing the default config files • Better suited for multi-site solutions • Solutions can be bundled easily for shipment • Development sites remain clear of code-behind files	• Changes to code typically need to be compiled and copied to the web root in order to see the changes and debug, which can also trigger your application to recycle • Initial solution setup can be complex as you create and employ custom post-build actions

Source Control and Serialization Options



Important

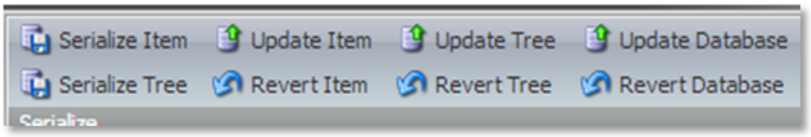
Serialization may not be appropriate when deploying changes to a staging or production environment because it overwrites items.

Walkthrough – Serializing Items

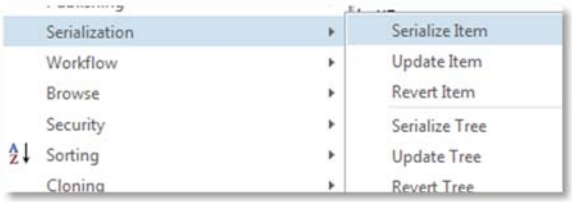
In the following walkthrough, we will:

- Serialize the entire database using http://traincore/sitecore/admin/serialization.aspx.
- Serialize items using the Developer tab in the Content Editor.
- Serialize items using Sitecore Rocks.

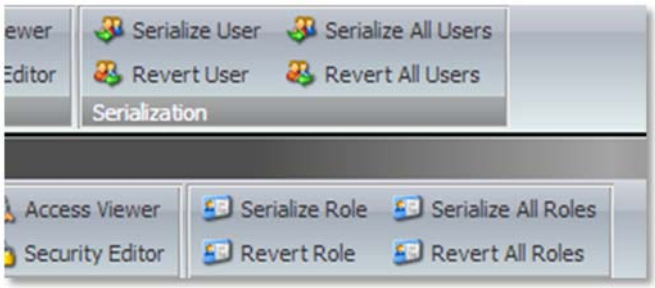
1. Navigate to the serialization page – http://traincore/sitecore/admin/serialization.aspx and serialize one of the databases in its entirety.
2. Navigate to the **/Data/serialization** folder and note that a folder with serialized items has been created as a result.
3. Log into the Content Editor and demonstrate that databases, individual items, or item trees can be serialized from the **Developer** tab:



4. Using Sitecore Rocks, demonstrate that items can be serialized by right-clicking and selecting **Tools > Serialization**:



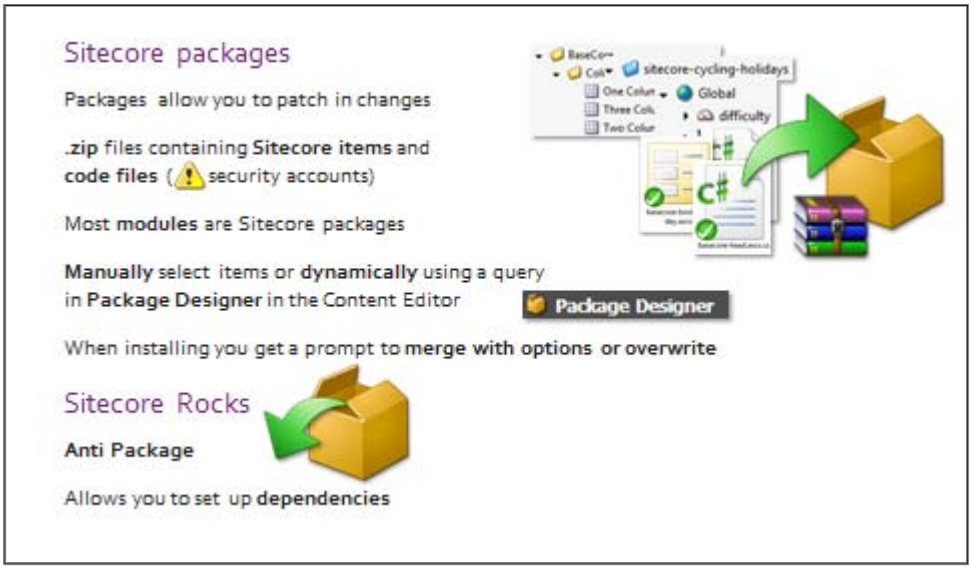
Important
Serializing the database will not serialize users and roles. However, you can serialize them from the Role Manager and User Manager interfaces on the Sitecore Desktop.



Tip
You can change the folder that items are serialized to by changing the following setting in the web.config:
<setting name="SerializationFolder" value="$(dataFolder)/serialization" />

Serialization guide:
http://sdn.sitecore.net/Reference/Sitecore%207/Serialization%20Guide.aspx
Team Development for Sitecore:
http://www.hhogdev.com/Products/Team-Development-for-Sitecore/Overview.aspx

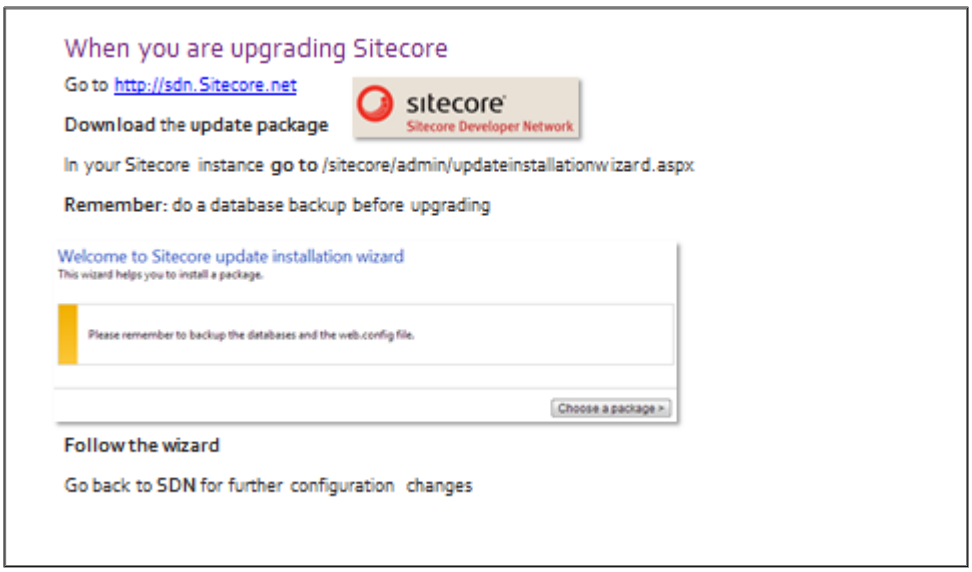
Packages



Tip
See the *Package Designer Administration Guide* if you are using the Package Designer interface on the Sitecore Desktop:
http://sdn.sitecore.net/Reference/Sitecore%207/Package%20Designer%20Administrator%20Guide.aspx

Important
When you package security accounts, all passwords are lost and have to be reset.

Update Packages



Demo – Create a Package Using Sitecore Desktop

In the following walkthrough, we will:

- Create a package using the Package Designer in the Sitecore Desktop.

1. Log into the Sitecore Desktop and click **Sitecore > Development Tools > Package Designer.**
2. Items can be added statically or dynamically using a query (for example, all items created in the past four days).
3. You can preview package contents using **Preview > Lookup.**
4. Generate a .zip file to create a file in the file system.

Tip
You can change the folder that packages are built to by changing the following setting in the web.config:
<setting name="PackagePath" value="$(dataFolder)/packages" />

Walkthrough – Create a Package Using Sitecore Rocks

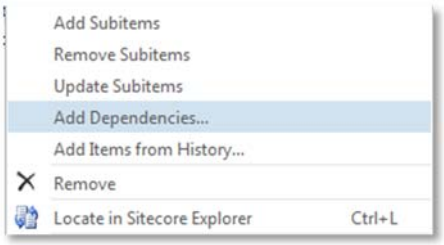
In the following walkthrough, we will:

- Create a package using Sitecore Rocks.

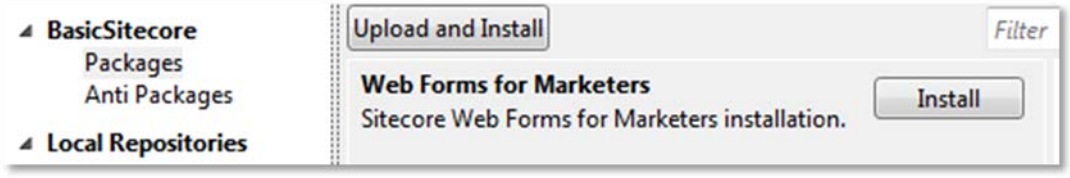
1. In **Solution Explorer**, right-click the **Training** solution and select the **Add > New Item** option.
2. Select **Sitecore** from the **Visual C#** menu on the left side of the screen. Then select the **Sitecore Package** option, and give it a name.
3. Connect it to the **Traincore** instance when prompted.
4. **Click** on the **Item** tab then **drag and drop** items from the tree into the package.
5. **Drag** and **drop** files from the **/Website** folder in Sitecore Rocks into the **package** and notice that they are added into the **File** tab.

   Note: you cannot drag and drop files from your solution, because solution files are independent of the Sitecore instance.

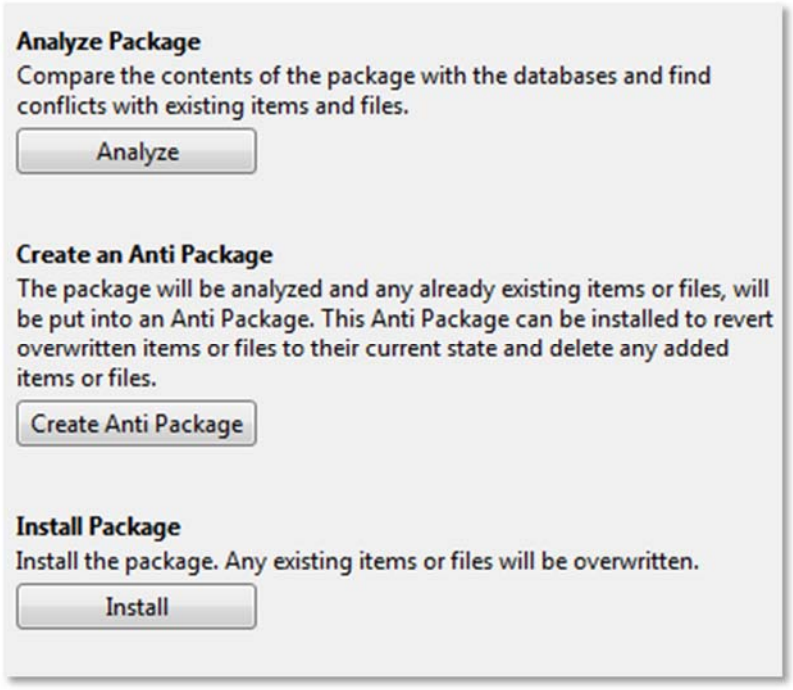
6. Notice that right-clicking on an item gives you options to **add, remove,** and **update** its children:



7. You also have the option to **add dependencies**. For example, if you add an item, you may want to add its template, associated renderings, and potentially any other items it references.
8. **Build** a package and **open Windows Explorer** when prompted. Notice that a **.zip** file has been created in the same location as your solution. However Sitecore stores packages in the **Data folder** of the Sitecore installation.
9. To install a package, right-click the **Sitecore instance** in Sitecore Rocks and select **Manage Packages…**
10. Click **Upload** and **Install** to upload a package to the instance:



11. When the package appears, click **Install**. You are given the option to **Analyze** and **Create Anti-Package** before installing:



Anti-packages allow you to **remove** the files and items that were installed, and **restore** the files and items that were changed.

Apply – Topic 9.5 – 5 min

Install a Package

Lab A. Install a Package Using Sitecore Rocks

Using Sitecore Rocks, install the package you just created in the walkthrough.

1. To install a package, right-click the **Sitecore** instance and click the **Manage Packages…** option.
2. Only packages in your package repository will be visible. Either use the **Repositories** button to add a repository with a package file, or copy the package file into the existing repository (the **Data folder**, in your Sitecore installation).
3. Notice that you can generate an anti-package that functions as an uninstaller.

Extend

- **Sitecore NuGet Packages**
  http://vsplugins.sitecore.net/Sitecore-NuGet.ashx

Topic 9.6 How to Deal with Deployment

Introduction

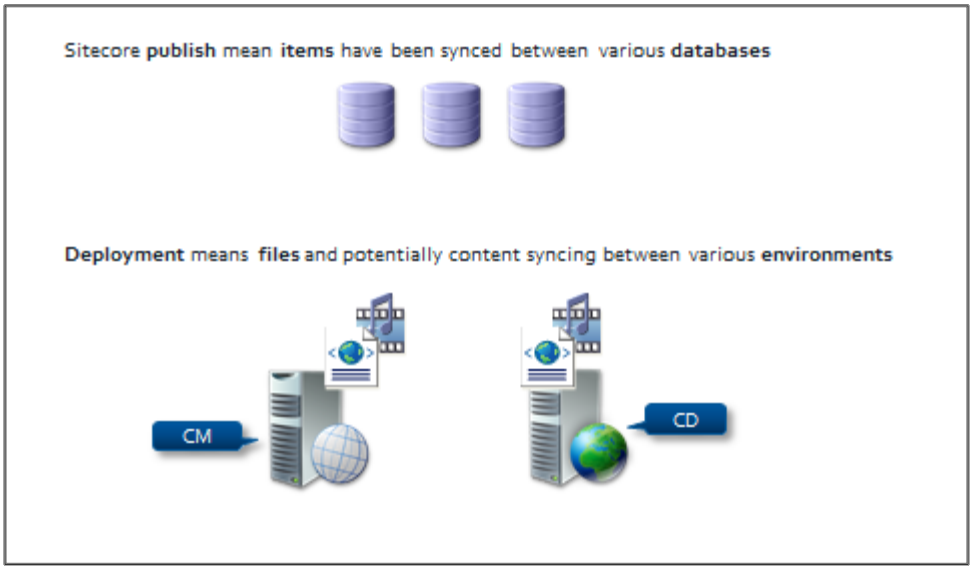
Objectives

By the end of this topic you will be able to:

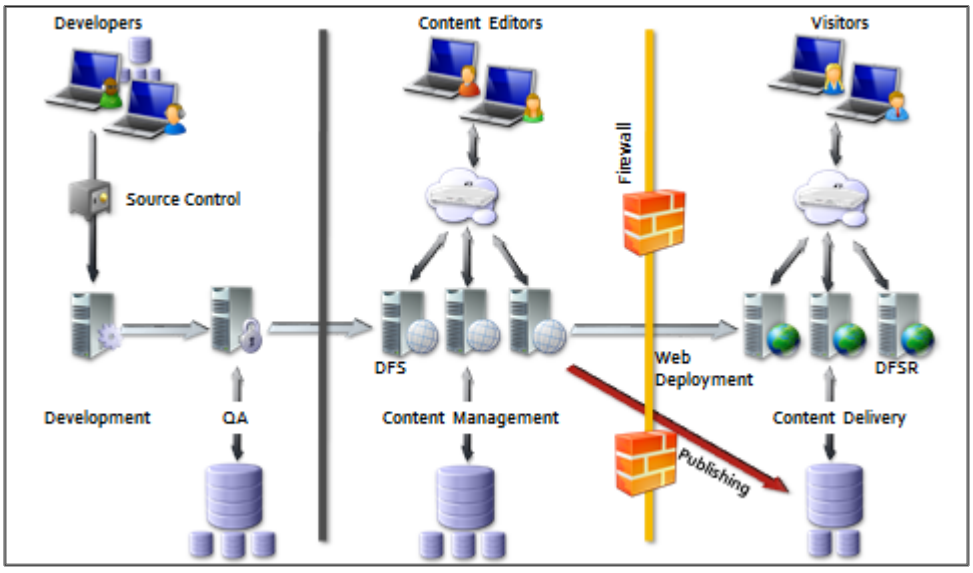
- State the difference between publishing and deployment.
- List some points to remember when deploying to an authoring environment.
- Discuss options for reverse content promotion.

Content

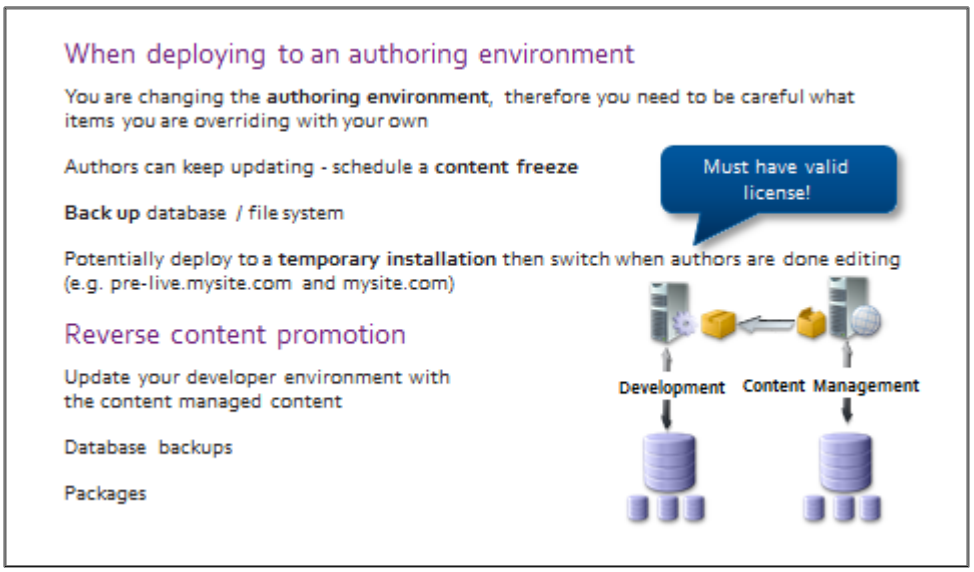
Difference Between Publishing and Deployment



Deployment Between Environments



Points to Remember



Topic 9.7 Basic Security

Introduction

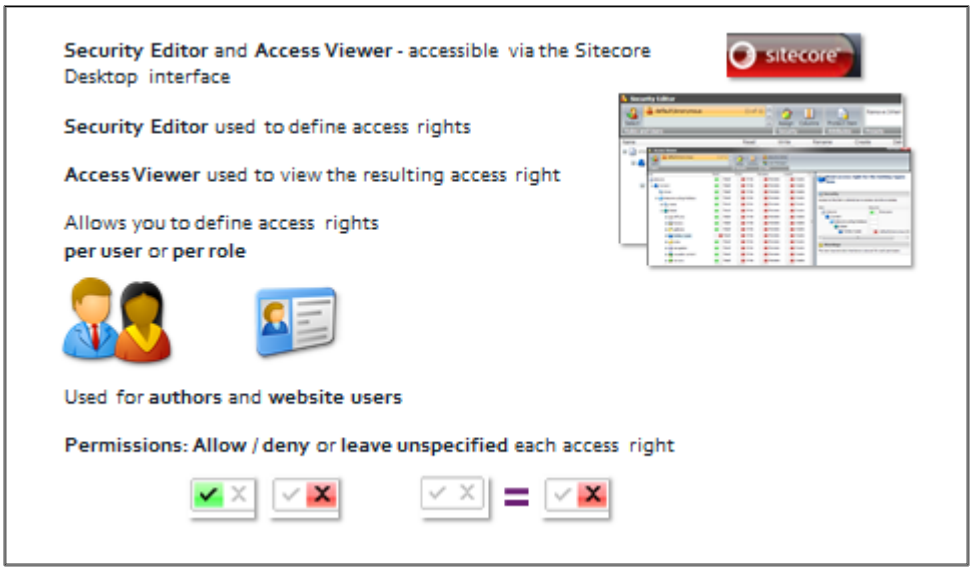
Objectives

By the end of this topic you will be able to:

- Name the application for defining access rights.
- Name the application for viewing the resolved access rights.
- List two entities that access rights can be assigned to.
- Describe the three permissions that are applied to access rights.
- Discuss how to override access rights.

Content

Security Editor and Access Viewer

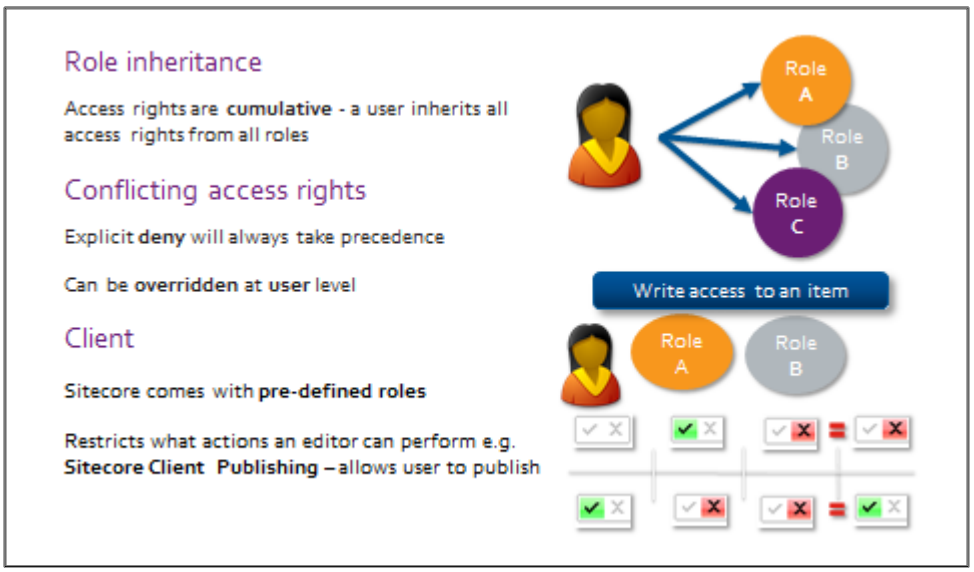


Tip

For more information about setting up roles and access rights, please refer to the **Security Administrator's Cookbook**: http://sdn.sitecore.net/upload/sitecore7/70/security_administrators_cookbook_sc70-a4.pdf

You can also attend Sitecore's the **Security Administrator course**: http://www.sitecore.net/services-and-support/training/administrator-training/ssa-sitecore-cms-security-administrator.aspx

Roles



Recommended Practices

Define a role per task (for example, news editor or image library manager) and avoid defining access rights directly on users. That way, it is easier to maintain a team of editors as their job responsibilities change.

Extend

- **Security reference**
  http://sdn.sitecore.net/Reference/Sitecore%206/Security%20Reference.aspx
- **Security API Cookbook**
  http://sdn.sitecore.net/Reference/Sitecore%206/Security%20API%20Cookbook.aspx

Topic 9.8 Workflow

Introduction

Objectives

By the end of this topic you will be able to:

- Describe the purpose of a workflow.
- Discuss what a workflow consists of.
- State when an item is publishable.
- State three workflow recommendations.

Content

Business Use Case

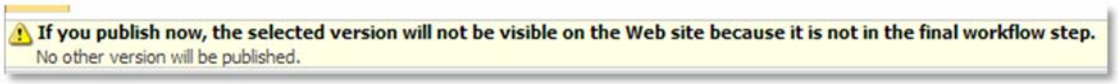
It may be illegal to have certain content online at certain points in time forcing authors to create new versions of the content. Or with some sensitive material, authors may have to have different groups in the organization approve the material before it can go live. Workflows allow you to set one source of responsibility for these issues.

Demo – Using a Workflow as an Author

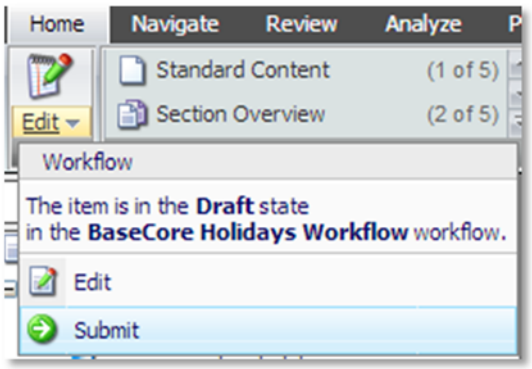
In the following demo, we will:

- Push an item through workflow.

1. Using the Content Editor, right-click on the */sitecore/content/sitecore-cycling-holidays/Home/holidays* item and insert a New Holiday. Name it *All-around-the-world*.
2. Notice that a message appears at the top of the content pane:

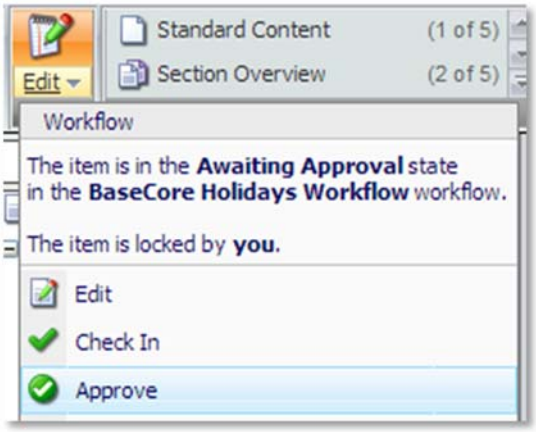


3. Click the arrow under the **Edit** button on the ribbon, and select **Submit**:



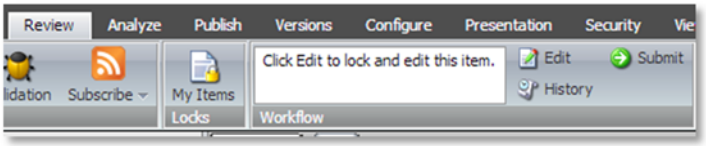
4. Enter a comment when prompted.

5. This is a two-step workflow. Right-click on the arrow under the **Edit** button a second time, and select **Approve**. Write a comment when prompted. This item will now go into a **final workflow state,** and is then publishable. In this particular case, entering into the final workflow states triggers an automatic publish of the item.



Tip

You can also use the **Workbox** interface to manage bulk manage items in workflow. This interface is available on the Sitecore Desktop, or as a button at the bottom of the Content Editor. You can also use the **Review** tab in the Content Editor:

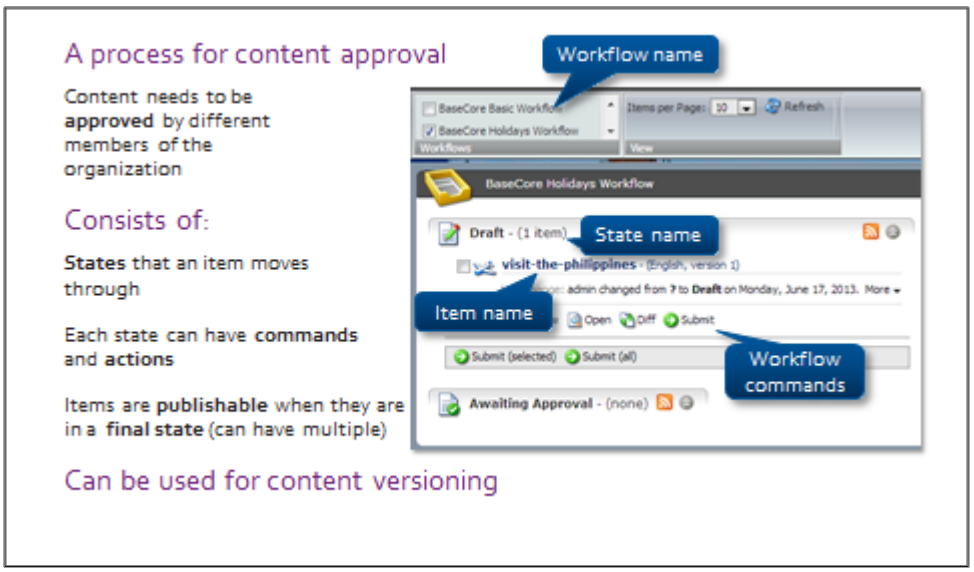


Best Practices

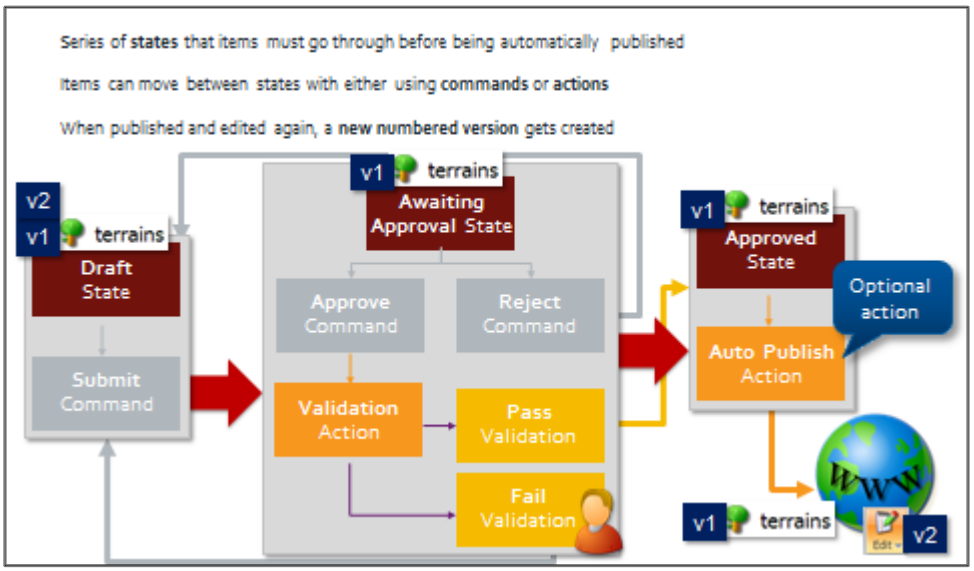
Workflow is useful for two reasons in particular; it maintains a **history of changes,** allowing authors to roll back to a previous version, and it **prevents accidental publishing of content**. Only content in a final state will appear on the live site. We recommend that you:

- Specify workflows on the Standard Values of a template.
- Use workflows from the beginning for all the content.
- Use clean up scripts to remove unnecessary, older numbered versions.

What Is a Workflow?



Workflow



Module 10
Marketing Functionality

Contents:

- Introduction to the Customer Experience Platform (XP)
- Engagement Value and Goals
- Profiling and Personalization

Topic 10.1 Introduction to the Customer Experience Platform (XP)

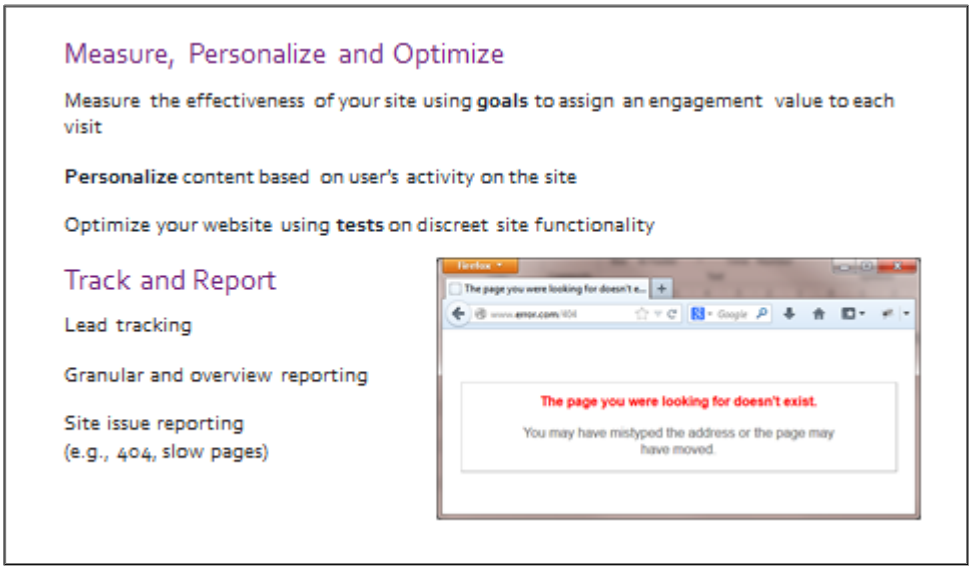
Introduction

By the end of this topic your will be able to:

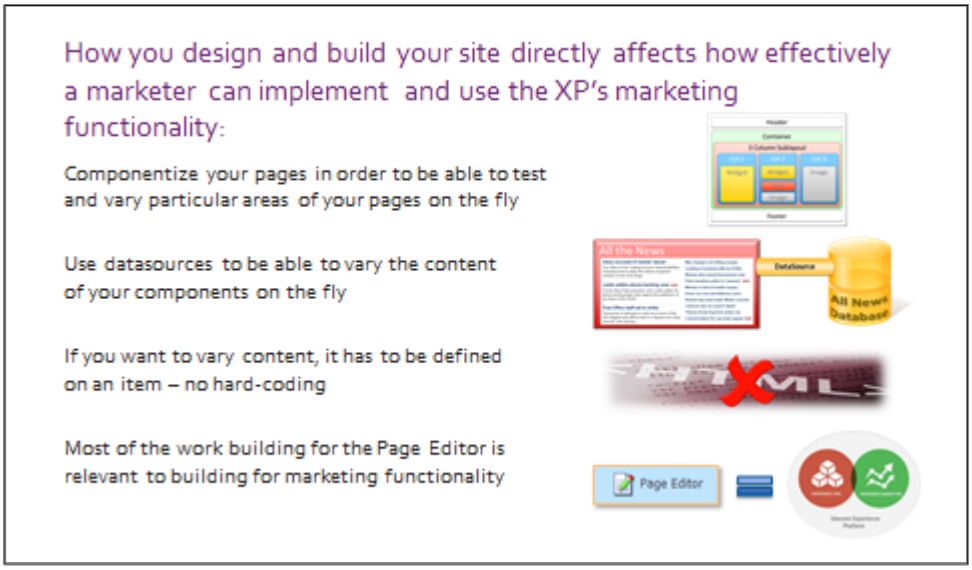
- State the three main features of the Customer Experience Platform.
- List two types of reports.
- Describe why componentization and the use of datasources are integral to using the XP.
- Discuss why building for the XP is much easier than retrofitting features later on.

Content

Marketing Functionality



Building for the XP



I'm Not Using It yet

Even if your not using the marketing functionality just yet, **build to support it in future:**

The implementation of digital marketing capabilities is often a **Phase 2** consideration – be prepared for it. **Retrofitting is much harder than planning ahead**

If you are building to support Sitecore's marketing functionality, you are building to support the **Page Editor** – you will save yourself time and get a lot of marketing features without having to do any extra work

Topic 10.2 Engagement Value and Goals

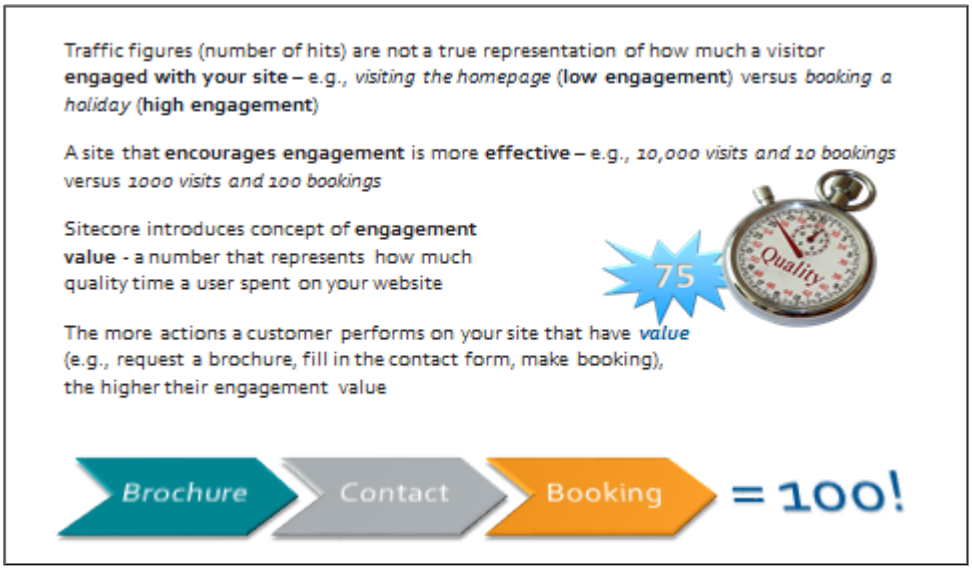
Introduction

By the end of the topic you will be able to:

- Describe the difference between value and traffic.
- Create goals and assign them.
- View the Dashboard and Engagement Analytics reports.
- Set up and run multivariate tests.

Content

Value vs. Traffic



Goals

Best Practice

Deciding what **your** goals **are** and how much they are worth is crucial to the collection of meaningful data. Customers are strongly encouraged to attend the XP Scoping Workshop:

http://www.sitecore.net/unitedkingdom/Support/Consulting-Services/Business-Optimization-Services/CEP-Scoping-Workshop.aspx.

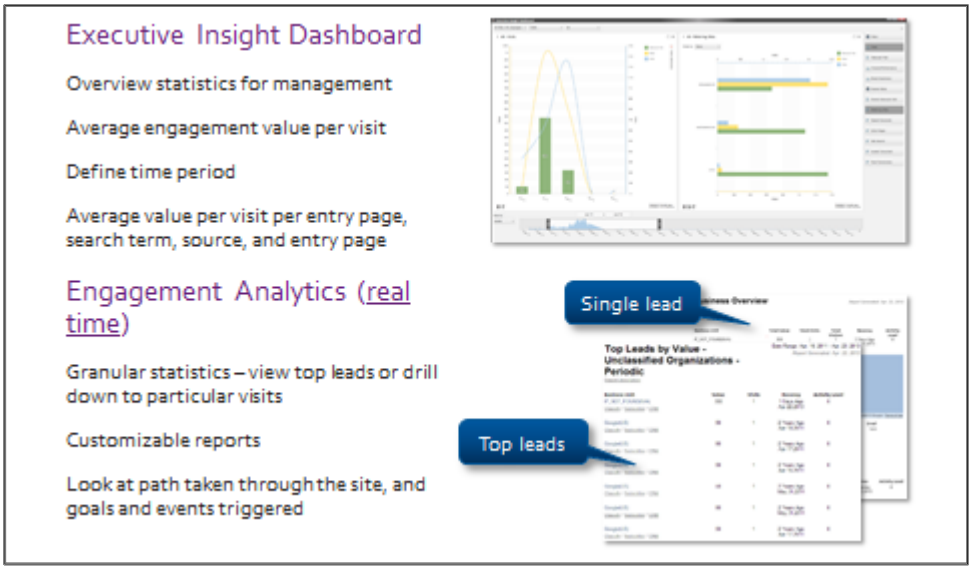
Demo – Create and Assign a Goal

In the following demo, we will:

- Using the Marketing Center, create a goal called *Book a holiday*.
- Give it a value of *50*.
- In the Content Editor, locate the booking page and assign the goal using the **Goals** button on the Analyze tab.

1. Using the **Marketing Center**, create a **goal** called ***Book a holiday.***
2. Give it a **value** of **50.** The entire **Sitecore Cycling Holidays** site is geared toward selling cycling holidays, so this is the most important task an author can perform.
3. On the **Review** tab, **deploy** the goal, (it is automatically part of an approval process and needs to be published. You will learn more about this in module 10).
4. In the **Content Editor**, locate the **booking thank-you** page and assign the goal using the **Goals** command on the **Analyze** tab.
5. Open **another browser**, for example, Internet Explorer, and book a holiday.
6. You will be redirected to the ***thank-you*** page after a successful booking.

Reporting



Demo – Executive Insight Dashboard and Engagement Analytics

In the following demo, we will:

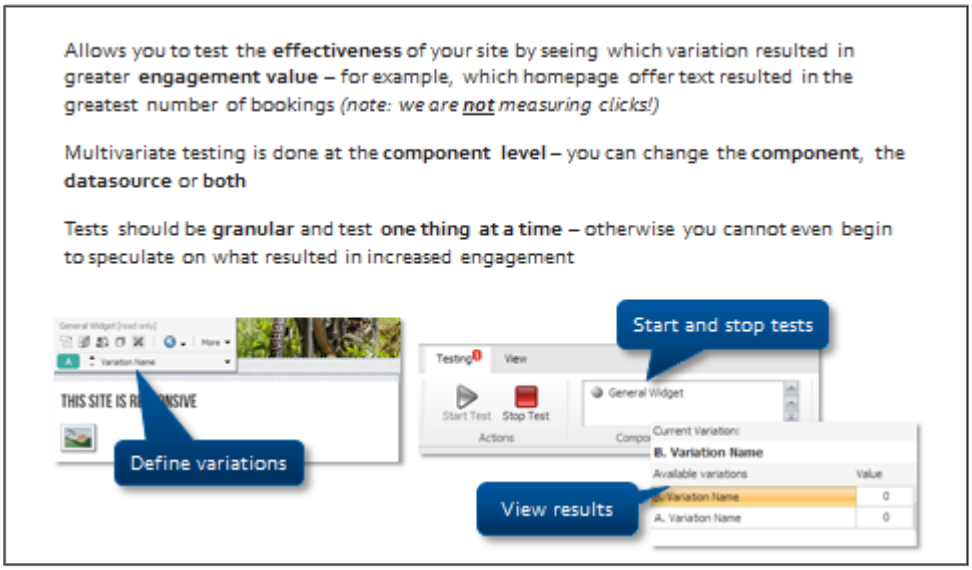
- View the Executive Insight Dashboard.
- View the Engagement Analytics reports.

1. Open the **Executive Insight Dashboard** and select the **Visits** checkbox on the right side of the screen. The resulting graph shows the average value per visit for the time period specified.
2. **Note** that you can view by year, month, week or day.
3. **Note** the other overview statistics that the Dashboard provides: average value per visit by entry page, search key word, source and referring site.
4. By contrast, **Engagement Analytics** reports are much more granular. Expand the **Reports** item. Click the **Sales > Top Leads by Value > Unclassified organizations** options. (Make sure that you have selected a wide date range).
5. **Note** that you can apply filters, subscribe to a particular lead, or view more details about that lead, including drilling down to individual visits.
6. Locate the session where you completed the holiday booking goal. **Note** that the visit has an engagement value associated with it.
7. Engagement Analytics reports also provides statistics on particular events related to site health (for example, common mistakes and not found URLs).

Tip

The Dashboard collects data over time and only displays data when the number of visits recorded exceeds the MimimumVisitsFilter (set to 50 by default) that was set in: \sitecore\shell\Applications\Reports\Dashboard\Configuration.config

Testing



Tip

You need to think about testing before you start building. Given that you can change the datasource or component, think about the following examples:

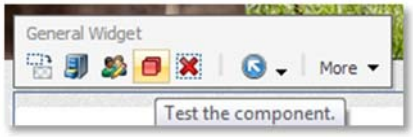
- *If you want to vary color or design, change the component. On the Sitecore Cycling Holidays website, color and design is, in part, determined by parameters. This means we need to set up a separate testing component if we want to test design.*
- *If you want to vary content that is not part of the main content of a component such as button text or form labels, these values need to be defined on your datasource. For example, you might have a 'form' component that accepts a 'form' item on the Sitecore Cycling Holidays website. Our forms use values from the Domain Dictionary, so we would need to set up a separate testing component.*
- *If you expect to be doing a lot of multivariate testing, make sure you componentize your page and keep all testable values in a data source. The functionality relies entirely on being able to change components and datasources.*

Demo – Setting Up a Multivariate Test

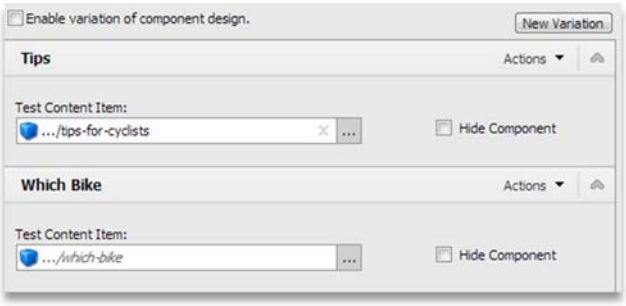
In the following demo, we will:

- Set up a multivariate test on a General Widget component.

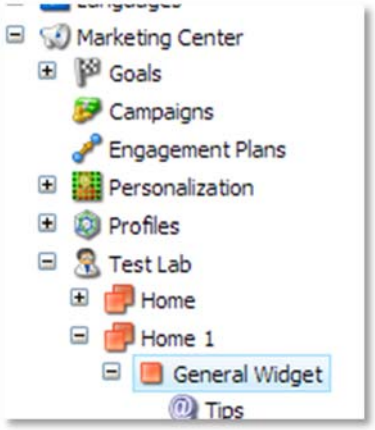
1. Open the homepage in **Edit mode** and make sure the **Designing** checkbox is selected on the **View** tab.
2. Select the component you wish to test. Click the **Test the component** button on the context menu.



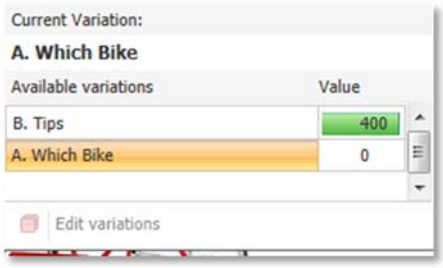
3. Add a new variation (It will default to what is already selected). Then add a second variation and only change the datasource of the component.



4. Notice that you can scroll between the variations.



5. **Smart publish** the page and start the test by navigating to the **Testing** tab and clicking the **Start Test** button. An item will be created in the tree at this point. There are extended properties on this item (such as, whether to make the variation *sticky* or *random*).
6. Look at the homepage in different browsers, using different sessions to emulate different visits (for example, in Internet Explorer, select ***File > New Session***). **Note** that you now receive a variation of the component at random.
7. In one of the sessions, complete the goal that was set up earlier in the module.
8. Go back into editing mode and look at the results of the test. You may have to recycle the application pool to see the value.



9. Stop the test and you will be given the option to pick the winning component.

Important

Test results are only used to determine which component to select. They are *not* saved.

Apply – Topic 10.2

Creating and Assigning Goals

Lab A. Create and Assign a Contact Page Goal

1. Using the **Sitecore Desktop**, open the **Marketing Center.**
2. **Create** a new goal called ***Viewed contact page*** and assign it a max value of ***5.***
3. On the **Review** tab, **deploy** the goal.
4. Using the **Content Editor**, assign the goal to the contact page by selecting the item, clicking the **Goals** option on the **Analyze** tab, and then selecting that goal you created.

Topic 10.3 Profiling and Personalization

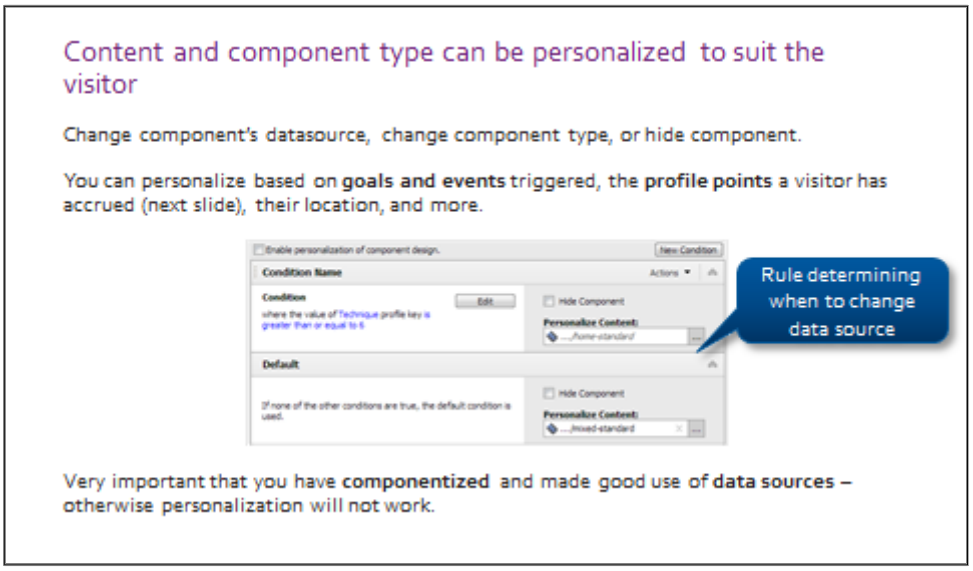
Introduction

By the end of this topic you will be able to

- Set up profile dimensions and keys.
- Describe how a profile card aggregates profile keys within a dimension.
- Assign profile cards to items.
- Set up a custom profile card.
- State the difference between a profile card and a pattern card.

Content

Personalization



Best Practice

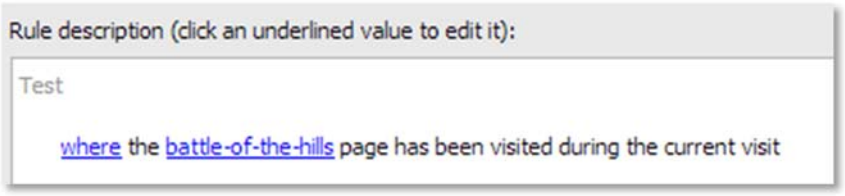
Encourage authors to make content profiling a *routine* **task** or risk ending up with 40,000 items that require profiling. Make sure that a meaningful profile taxonomy has been created. Defining dimensions, keys, profile cards and pattern cards is a *business decision not a development decision*.

Demo – Personalize Based on Page Visited

In the following demo, we will:

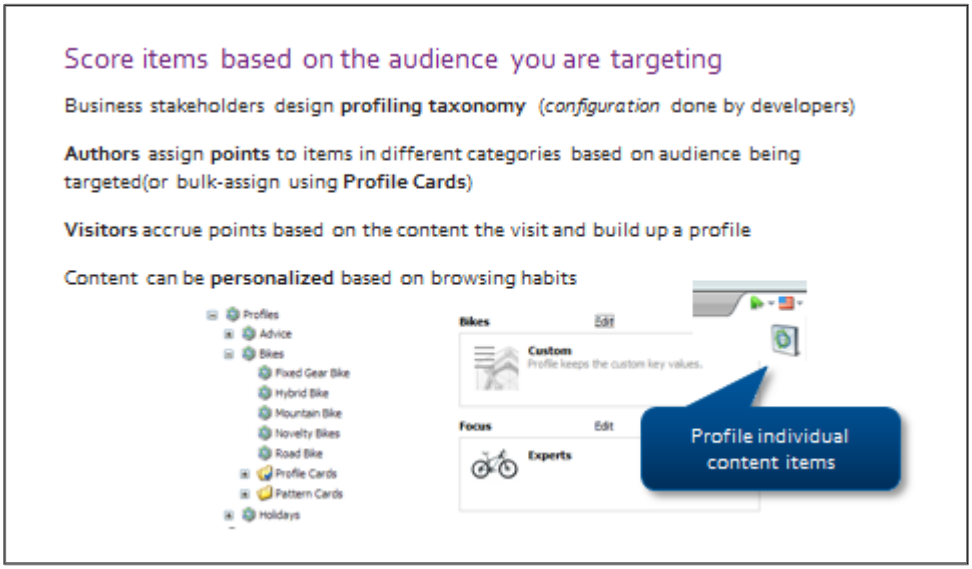
- Set up a personalization rule that changes a homepage General Widget component datasource to *Expert Holidays* if the *Battle of the Hills* site has been visited.

1. Using the **Page Editor**, select one of the **General Widget** components on the home page.
2. Click the **Personalization** button.
3. Add a **new condition** – name it **Experts**.
4. Create a rule that says *when Battle of the Hills has been visited*:



5. In the **Personalize Content** dropdown menu, select: */sitecore/content/sitecore-cycling-holidays/global/reusable-content/for-experts.*
6. **Republish** the page, and log out of the Page Editor.
7. Open a **new browser**, such as Internet Explorer. Ensure that the browser cache is cleared.
8. Browse to http://traincore/. **Note** which General Widget components are being displayed.
9. Visit the **Battle of the Hills** holiday page, and then go back to **Home**. The component should now be displaying the **for-experts** content.

Profiling Content Items



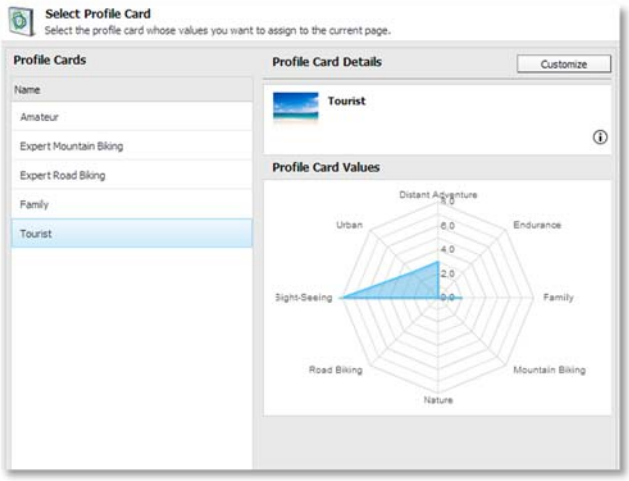
Demo – Profile Content

In the following demo, we will:

- Assign **Tourist** profile card to a holiday in the **Holidays** profiling dimension.
- Assign **custom point values** in the **Bike's** profiling dimension.

1. Open the **Content Editor**.
2. Select the **Battle of the Hills** holiday item under: */sitecore/content/sitecore-cycling-holidays/Home/holidays.*
3. Click the **Analytics icon** in the top right-hand corner of the content pane:

4. Click **Edit** next to the **Holidays** profile dimension.
5. Select the **Tourist** profile card. This bulk assigns certain profile points, which you can see in the right-hand graph:



6. Click **OK**, and then click **Edit** next to the **Bikes** profile dimension.
7. Click the **Customize** button in the top right-hand corner.
8. Set **Mountain Bike** to *10.*
9. Click **OK**, and then click **Close**.

Module 11
Optional Topics

Contents:

- Branch Templates
- Other Item and Template Properties
- Pipelines and Events
- Rules
- Placeholder Overrides

Topic 11.1 Branch Templates

Introduction

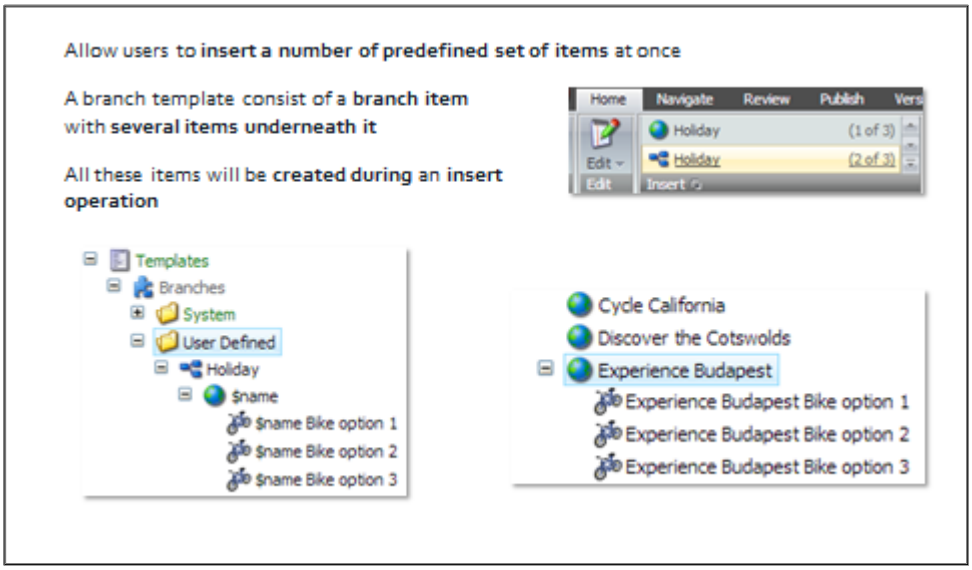
Objectives

By the end of this topic you will be able to:

- Describe how to create a list of child items for an item in one click.

Content

Branch Templates

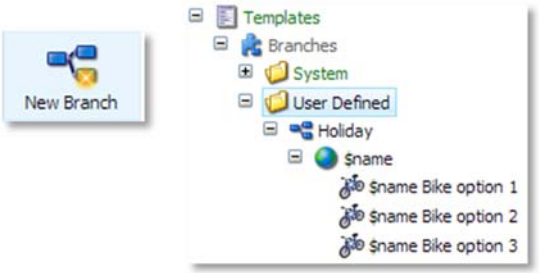


Walkthrough: Creating a Branch Template

In this walkthrough you will learn how to:

- Create branch templates in the Template Manager.
- Assign a new branch template as an insert option for the **Trip Details** data template.

1. Open http://BasicSitecore in Sitecore Rocks. If it does not already exists, create a data template named **Bicycle** and set it as an insert option on the **Trip Details** standard values.
2. To create a new branch template, navigate to */sitecore/templates/Branches/User Defined* (ensure that **Hidden items** is checked in the **View** tab) and select *New Branch*.
3. Select the **Trip Details** data template as your starting point, and click **Create**.
4. Click on the *$name* item that uses a globe icon. Click the **Home** tab. You will see **Bicycle** as an **insert option**. Insert three **Bicycle** items – call them *$name Bike option 1*, *$name Bike option 2* and *$name Bike option 3*.

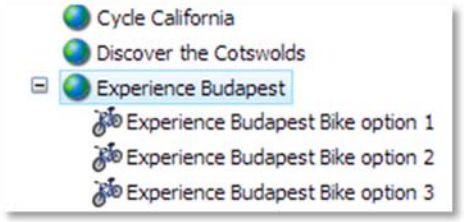


Knowledge Check

What will happen to the $name token when an item is created based on the Trip Details branch template?

---

5. Assign this **Trip Details** branch template as an insert option for the Holidays Section.
6. Let's create an instance of this branch template. Navigate to */sitecore/content/home/Family Holidays item* and, from the insert options, select the **Trip Details** branch.
7. Give the trip a name – such as ***'Experience Budapest'***.
8. A new sub-tree has been created and all $name tokens have been replaced with *Experience Budapest*:



Note

Branch templates can be assigned as insert options on standard values in exactly the same way as regular templates.

Topic 11.2 Other Item and Template Properties

Introduction

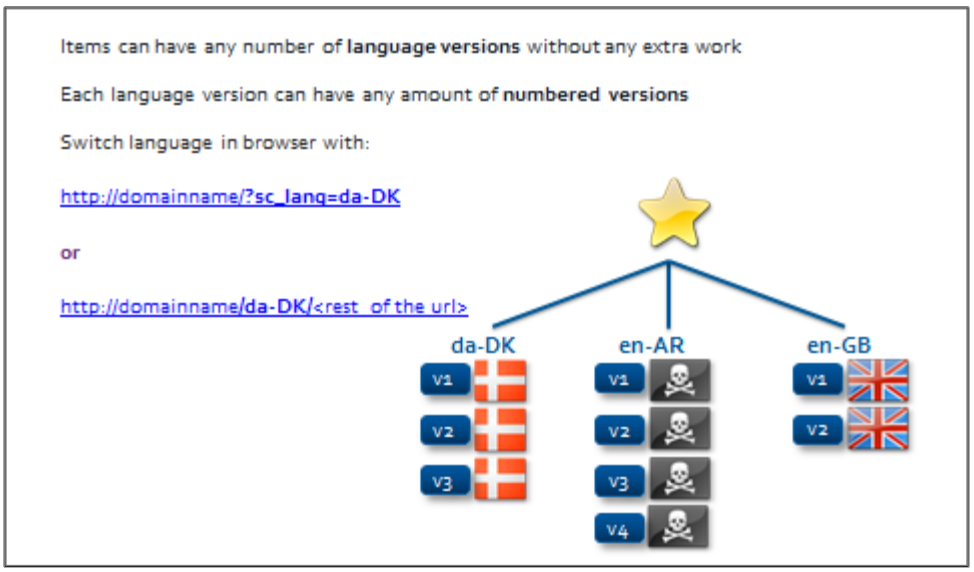
Objectives

By the end of this topic you will be able to:

- Explain language versions.
- Discuss numbered versions.
- State three types of field versioning.
- Explain the purpose of setting the field source.
- Describe where to set field sorting.
- Discuss where to set helper text.

Content

Language and Numbered Versions



Important

For content to exist you need to have a language version and at least one numbered version of the content.

You may have a website where content is needed in ten different languages. Instead of creating ten individual websites for each language or using .NET resource files, you can add languages to the Sitecore implementation without the developer's involvement.

Demo – Languages and Numbered Versions

In this demo you will see:

- That you can have content in different languages.
- How to create a numbered version
- How field versioning works.

The following demos use the http://BasicSitecore instance.

Languages and Numbered Versions

1. Using **Sitecore Rocks,** right-click the **master database** and select *Tools* > *Manage*.
2. Right-click in the **editor** pane and select the **Add New Language… > Danish**.
3. Navigate to the **Explore Holland** item and switch between the languages. Notice there is no content showing for the Danish language because there is no numbered version under it.

4. For the Danish language, you need to add a new numbered version of the content so right-click in the editor pane and select *Versions* > *Add Version*.
5. Switch languages in the browser using *sc_lang query string (?sc_lang=da-DK).*

Field Versioning

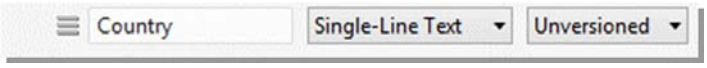
What if travel agents needed a unique ID for every trip details item? You would not want authors to have to retype an ID every single time they create a new numbered version of that item, or a new language. They can easily make mistakes and this is where shared fields come in.

1. In the **Trip Details data template,** create a new field section called ***Holiday Admin*** and a field called ***Holiday ID,*** which is **shared** (single value for all versions in all languages).

2. Enter a default value for the **Holiday ID** field.
3. Navigate back to **Explore Holland** and switch between the **language** and **number** versions, and notice that the field value is the same.
4. A month has passed and you need a new numbered version of the content. Create ***V#2*** in **Danish**.
5. **Switch** between ***DK v#1*** and ***DK v#2*** (same field value).

Business Use Case
What if you have a field with a name of a country that you want to keep the same for all numbered versions in a particular language? For example, the country itself doesn't change, but every language may spell a country name differently. This is where *unversioned fields* come in.

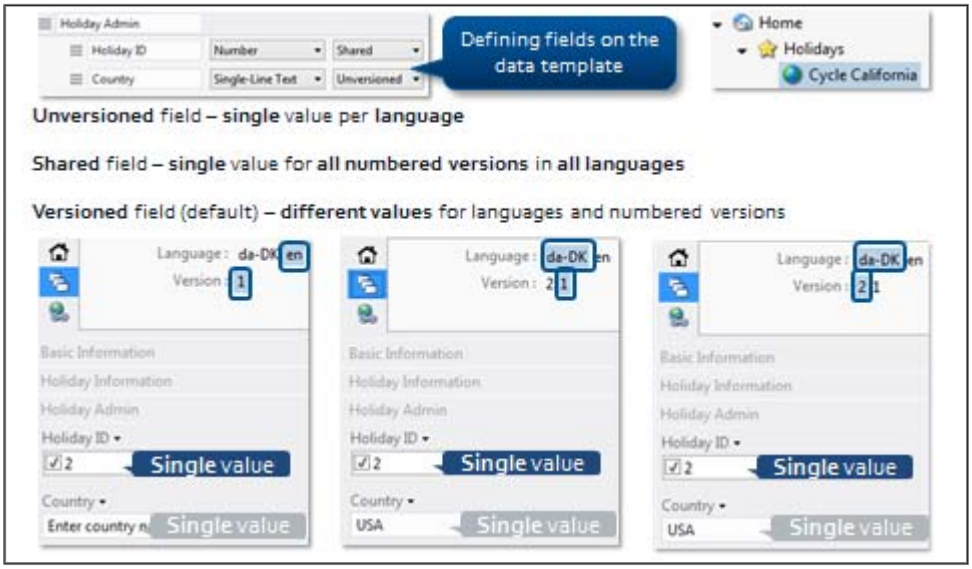
6. In the **Trip Details data template,** create a new field called ***Country***, which is **Unversioned** (single value per language).



7. Enter a default value for the **Country** field.
8. Navigate back to **Explore Holland** and switch between the **language** and **number** versions. Notice the English version is different from the Danish one, which is empty.
9. Fill in the **Danish field** with ***USA***.
10. Switch to ***English v#1*** (different value).

The **Holiday ID** field is the **same** for **all languages** and **all numbered versions,** whereas the **Country** field is **language specific,** same value for each language and each version number under it.

Field Versioning



Tip

If you have an item representing a setting, for example a CSS class, you would want that to be the same across all languages for that item. This is a good candidate for making a field 'Shared' and 'Unversioned' – meaning it's the same across all languages, and there is only one numbered version of that item. **Note** that a shared field is automatically unversioned.

Demo – Helping Authors

In the following demo, we will:

- Sort fields.
- Add help text to fields.
- Change the *Title* of a field to something more intuitive.

This demo uses the http://BasicSitecore instance.

Field Sorting

In this demo you will see that fields have a sorting order and can be reordered for display in the editor pane.

1. In the __Standard Values item for the **Trip Details** data template notice the order of the field sections **Basic Information** and **Trip Information**.
2. To change the sorting order we need to go to the definition of those field sections. Navigate to */sitecore/templates/User Defined/Trip Details/Trip Information*, right-click in the **Editor** pane, and in the context menu, select the **Standard Fields** option.
3. In the standard fields look for the **_Sortorder** field in the **Data** field section.
4. Change the value to **0** (bringing that field section to the top).
5. Double-click */sitecore/templates/User Defined/Base/Basic Information*, and change the value of _Sortorder field to **10** (bringing that field section beneath Trip Information).
6. Open up the **__Standard Values** item for the **Trip Details** data template and notice that your field sections have now changed order.
7. Lastly, right-click again in the **Editor** pane and clear the **Standard Fields** checkbox.

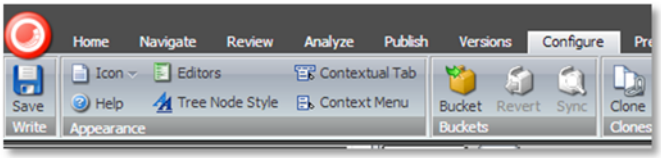
Help Text

In this demo you will see that fields can have helper text added to them to help authors fill in fields.

1. In the **Content Editor**, double-click the Type field definition item */sitecore/templates/User Defined/Trip Details/Trip Information/Price per person* and right-click to view the standard fields.
2. Find the **Help** field section. In the **__Short description** field, type: *This must have two decimal places*.
3. In the **__Long description** field, type: *A validation rule will ensure that this field has two decimal places*.
4. **Preview** a bicycle item or create a new bicycle item. You should see fields amended.

Tip

Using the Contend Editor, a quicker way to get to the help fields is to select the field definition items in the tree. But instead of viewing standard fields, you can click on the Configure tab and the help command. This opens up the Help field section.

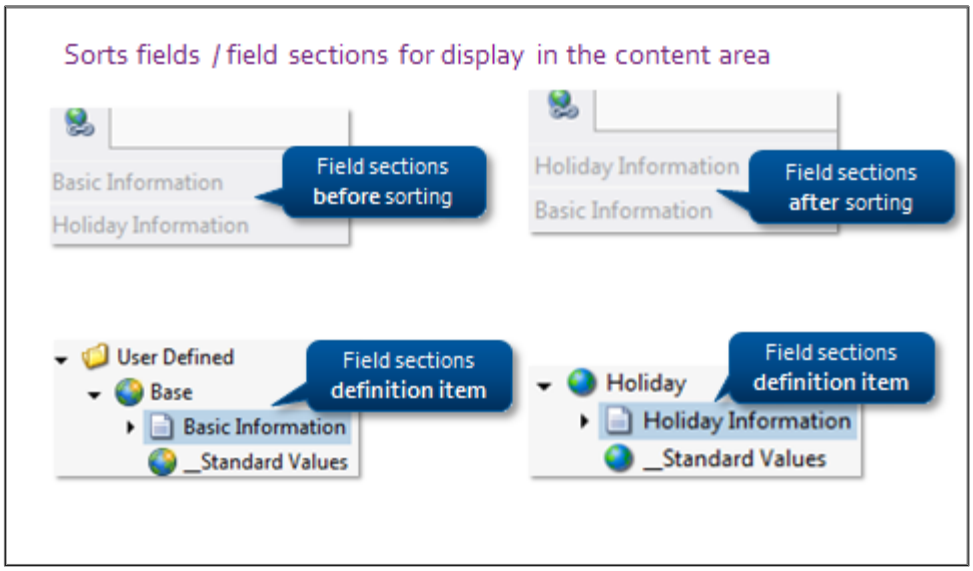


Title Field

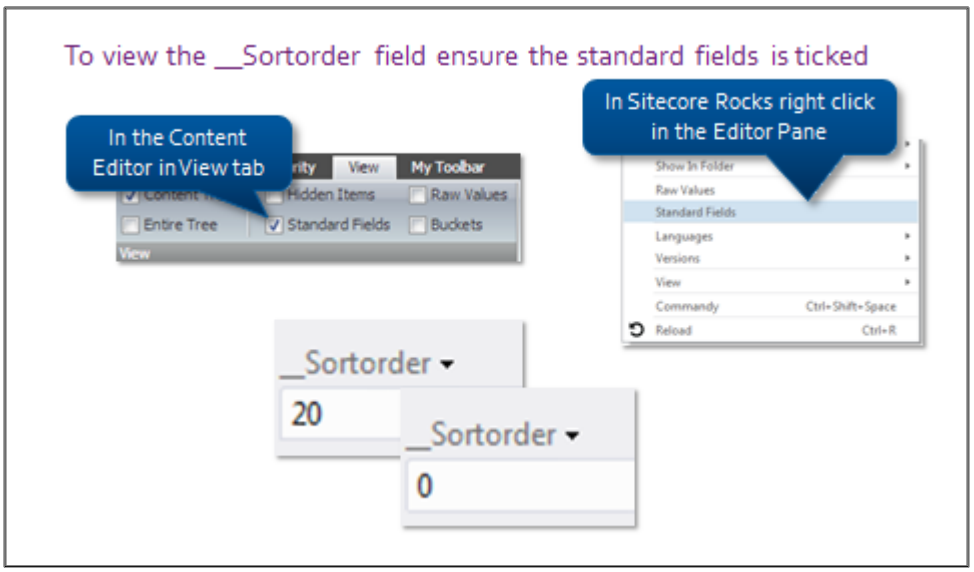
If you want to completely change the field name that is displayed to authors, you can do that in the **Title** field. For example, you can change *Holiday ID* (which developers will still use and see) to *Booking reference number* (which is what authors see).

1. Navigate to */sitecore/templates/User Defined/Trip Details/Holiday Admin/Holiday ID* definition item.
2. In the *Title* field (Data field section) enter *Booking reference number.*
3. Click the **Save** button.
4. Preview any item in the Content Editor.

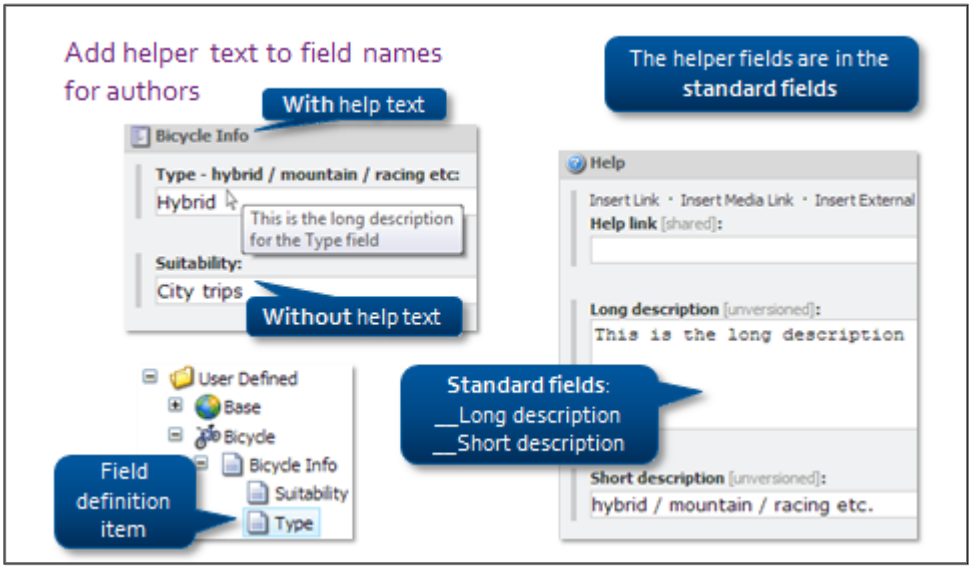
Sorting Fields



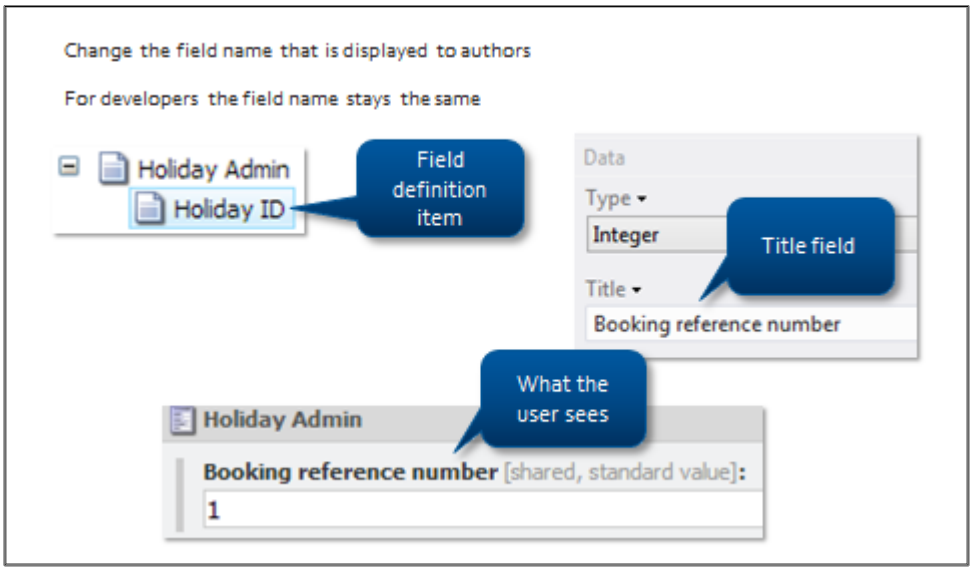
Show Standard Fields



Help Text



Title Field



Tip

You can create default values for language specific items on the standard values. For example, the default page image field for English is image A, and for German it's image B. This is done by creating new numbered version of that language in the standard values.

Apply – Topic 11.2 – 35 min

Working with Versions, Sorting, Help Fields, and Field Source

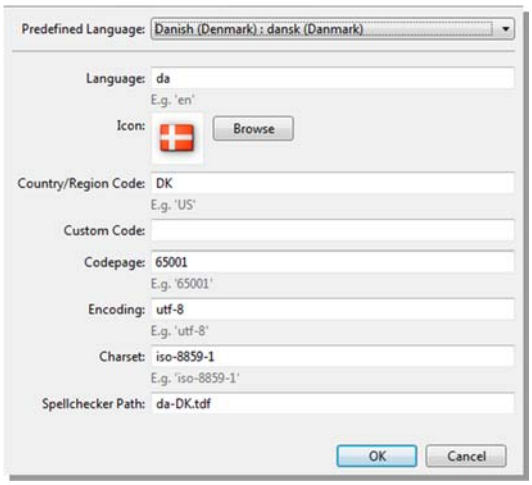
In the following labs you will be:

- Adding a new language.
- Changing field and item versioning.
- Altering the sort order for the Bicycle template.
- Setting helper text for your **Bicycle Information** item.
- Changing the title field of a particular field definition item.

Lab A. Add a New Language and Some Numbered Versions

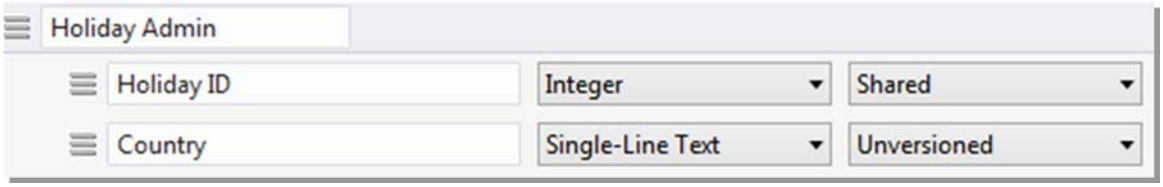
1. Go back to the http://BasicSitecore/ site you created on day one. Expand the **BasicSitecore** instance node in Sitecore Rocks.
2. Using **Sitecore Rocks,** right-click the **master database** and select **Tools** > **Language**.
3. Right-click in the content pane and select the **Add New Language…** option.

4. Select the **Danish** language.



5. Navigate to the **Explore Holland** item and switch between the **English** and **Danish** language.
6. For the Danish language, you need to add a new numbered version of the content. Right-click in the **Editor** pane and select the **Versions > Add Version** option.

7. The fields you get in a new version are blank because the field values have not been defined on the standard values for that particular language. **Enter** some **values** for the fields.
8. In the **Trip Details data template,** create a new field section called ***Holiday Admin*** and two new fields called ***Holiday ID***, which is **Shared** (single value for all versions in all languages), and a field called ***Country*** which is **Unversioned** (single value per language).



9. Enter a default values for the **Holiday ID** and the **Country** fields in the Standard fields.
10. Navigate back to **Explore Holland** and switch between the **language** and **number** versions. Notice that the Holiday ID value is the same between languages, but the Country value in English is different from the Danish one, which is empty.
11. Fill in the **Danish** field with **USA**.
9. Switch to **English v#1** (different value).
10. Create **V#2 in Danish** by right-clicking the **Editor** pane and then select the **Versions > Add Version** option.

11. Switch between **DK v#1** and **DK v#2** (they should be the same value – USA).

Lab B. Field Source

In this lab, you will specify a location of where your authors can select images.

1. In the **Content Editor,** navigate to a content item – such as **Explore Holland** at: */sitecore/content/Home/Family Holidays/Explore Holland*.
2. Change the image and note the browsing location.

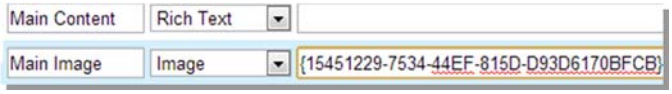
3. Close that and navigate to the **Explore Holland** data template using the **Quick Info** section.
4. Remember **we are inheriting** that image field from the Base data template, so navigate to **Base**.

5. Change the **field source** to be the **location of a folder in media library**.

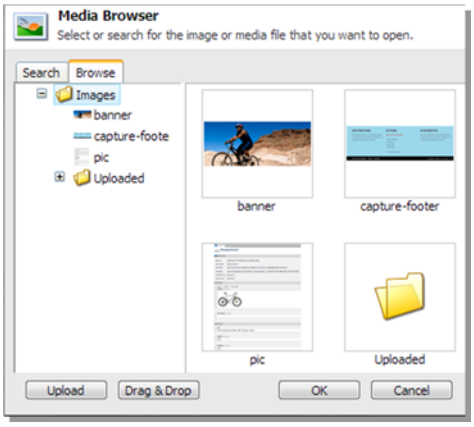
Tip

You can use the item path in the tree or the GUID for field source. There are different field sources depending on different field types. We will cover this later in the course.

6. In the source field type */sitecore/media library/Images* or use the **GUID** of that item.



7. Go back to **Explore Holland** and browse to an image or check the standard values. **Note** the change of location.



Lab C. Changing the Sort Order

In this lab you will change the sort order of the Bicycle fields.

1. View the __Standard Values item for the **Trip Details** data template.
2. **Note** the order of the fields *Holiday ID* and *Country* in the **Holiday Admin section.** We are going to swap their order.
3. Double-click the **Holiday ID** item at: */sitecore/templates/User Defined/Trip Details/Holiday Admin/Holiday ID.*
4. Right-click inside the **editing** pane to view the **standard fields**.
5. Change the **__Sortorder** field to ***20***.
6. Double-click on the **Type** item at: */sitecore/templates/User Defined/Trip Details/Holiday Admin/Country*.
7. Change the **_Sortorder** field to ***200***.
8. Right-click again to clear the **Standard Fields** checkbox.
9. Open up the **__Standard Values** item for the Bicycle data template, and notice that your fields have now changed order.

Tip

You can change the sort order of template sections in exactly the same way. As well as manually updating the sort order fields, you can right-click on items in Sitecore Rocks and choose **Tools > Sorting** to select the options there. The lower the number the higher it appears in the hierarchy.

Lab D. Help Fields

In this lab you will add some help text to your Bicycle Information fields using the **Content Editor**.

1. Using the **Content Editor**, double-click the **Type field definition** item at: */sitecore/templates/User Defined/Trip Details/Holiday Admin/Holiday ID* and on the ribbon, select **View > Standard Fields**.
2. Find the **Help** field section and the **__Short description** field. Enter *This is the ID from your external feed.*
3. Click **Save**.
4. Double-click the **Country** field definition item at: */sitecore/templates/User Defined/Trip Details/Holiday Admin/Country.*
5. Find the **Help** field section and the **__Short description** field, enter *Remember that the country name is translated.*
6. **Save**.
7. Look at an existing bicycle item and you will see the help text appear next to the fields.

Lab E. Title Field

In this lab you will change the way the field name is displayed to authors, however your field definition will stay the same. This is particularly useful in case authors need a longer description for a field but as a developer you still reference a different name/ID.

1. In the **Content Editor**, navigate to: */sitecore/templates/User Defined/Trip Details/Holiday Admin/Holiday ID* definition item.
2. In the **Title** field (Data field section), enter: *Booking reference number.*
3. Click **Save**.
4. **Preview** any item that has that field in the Content Editor.

Topic 11.3 Pipelines and Events

Introduction

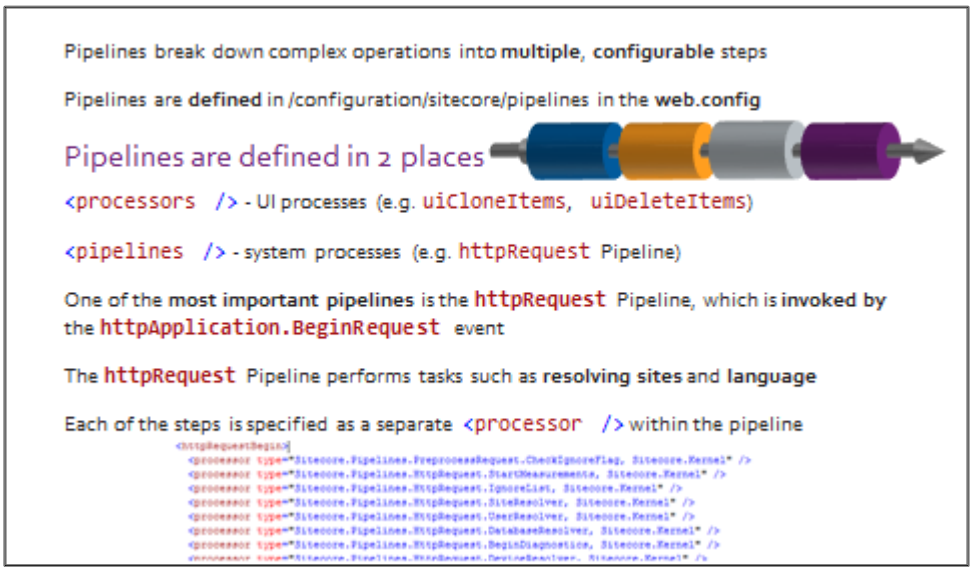
Objectives

By the end of this topic you will be able to:

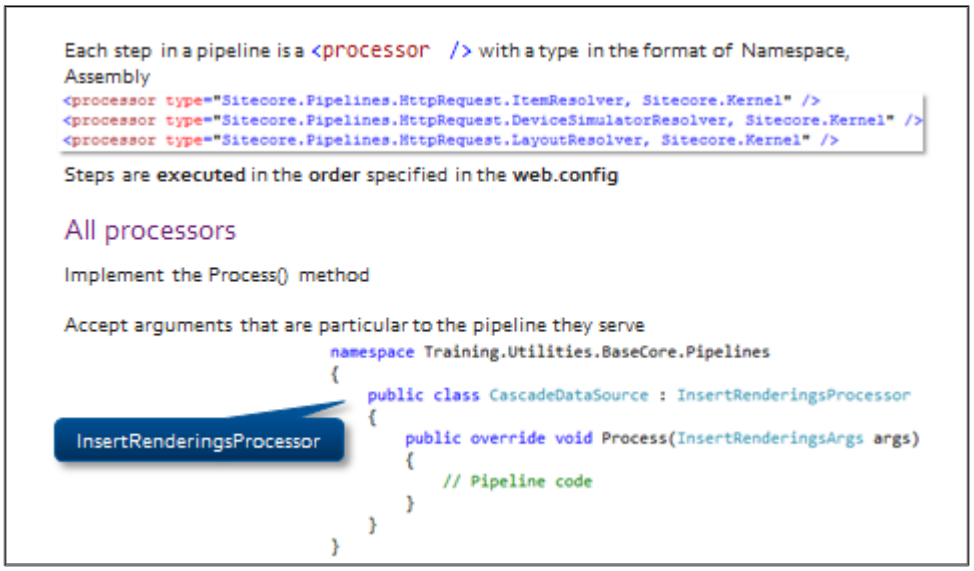
- Define pipelines and events.
- Describe where to configure pipelines and events.
- Describe the type format for the <processor />.
- Name the most important pipeline.
- Name the method that all processors implement.
- State which files are used to patch your processor into the pipeline.
- List three options for creating pipelines.

Content

Introduction to Pipelines



Processor



Creating Pipelines Option 1

There are 3 **options** for using pipelines to customize Sitecore behaviour

Option 1: Add processors to existing pipelines

Locate the processor that you want to extend – e.g. `insertRenderings`

Decide where your step fits in – e.g. if your processor modifies the parameters on a rendering, you want it to appear **after** the renderings have been added to the page – i.e. after: `Sitecore.Pipelines.InsertRenderings.Processors.AddRenderings`

Use **include files** to patch your processor into the pipeline

Determine whether processors in the pipeline inherit from an abstract class – in the case of the `insertRenderings` pipeline, all processors inherit from the abstract `InsertRenderingsProcessor` class

Decide what args the Process() accepts – in this case `InsertRenderingsArgs`

Write your pipeline code in the Process() method

Creating Pipelines Options 2 & 3

Option 2: Override and extend existing processors

Instead of adding extra steps to a pipeline, **replace the processor with your own** by deleting the existing processor, using include files and patching in your own

By either:

Inheriting from that processor giving you access to t… …s in the original `AddRenderings`

```
class MyCustomAddRenderings : Sitecore.Pipelines.InsertRenderings.Processors.AddRenderings
```

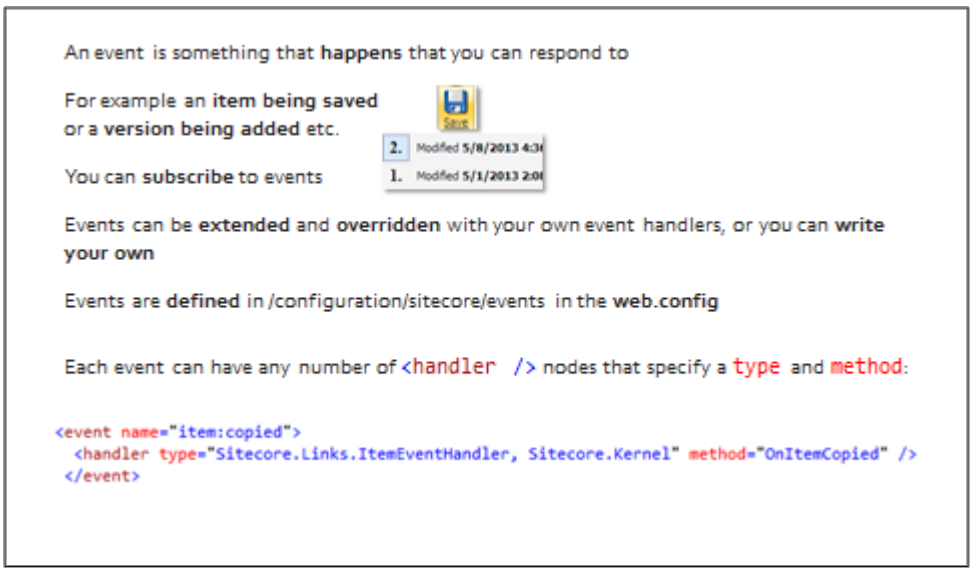
Note: Pipelines may need updating if Sitecore has been upgraded

Option 3: Create new pipelines that you call from your code

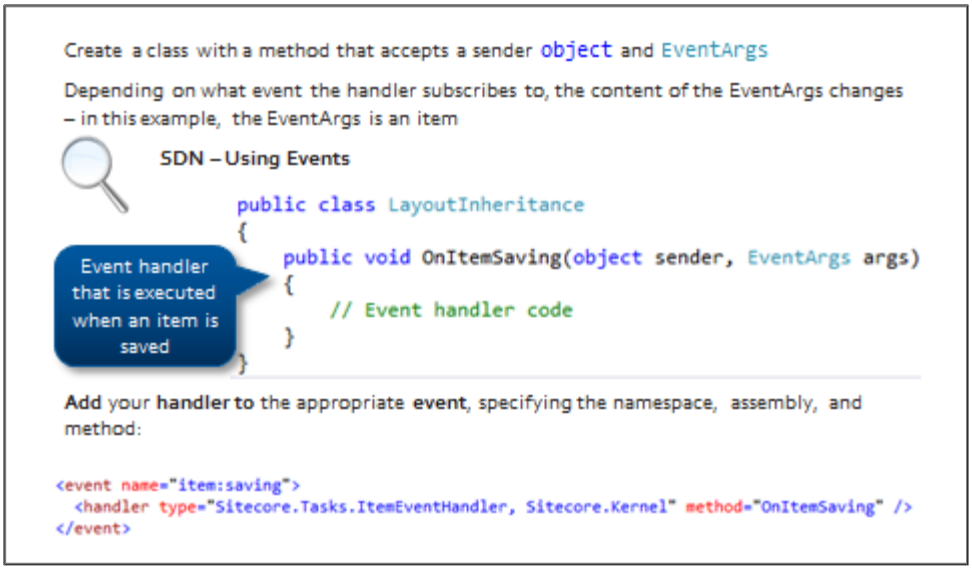
Important

Upgrading Sitecore may change the original code that you are inheriting from, which means that your pipeline may need updating.

Introduction to Events



Subscribing to Events and Using Event Handlers



Tip

For an example of how to create an Insert Options pipeline, see *The Data Definition Cookbook*: http://sdn.sitecore.net/upload/sitecore6/datadefinitioncookbook-a4.pdf

For more information about events, see John West's blog: http://www.sitecore.net/Learn/Blogs/Technical-Blogs/John-West-Sitecore-Blog/Posts/2011/05/All-About-Events-in-the-Sitecore-ASPNET-CMS.aspx

When Would You Use Pipelines or Events?

If a task is not needed in realtime, Sitecore Jobs can be used to perform maintenance rather than packing those tasks in the save or publisher pipeline slowing down the process

Is it a pipeline or an event...

Intercept requests and authenticate the user against an external service **PIPELINE**

When a news item is saved, automatically move it into to the appropriate year / month / day folder structure **EVENT**

Change the way Sitecore resolves URLs **PIPELINE**

When publishing begins **EVENT**

Change how sitecore resolves a language **PIPELINE**

Topic 11.4 Rules

Introduction

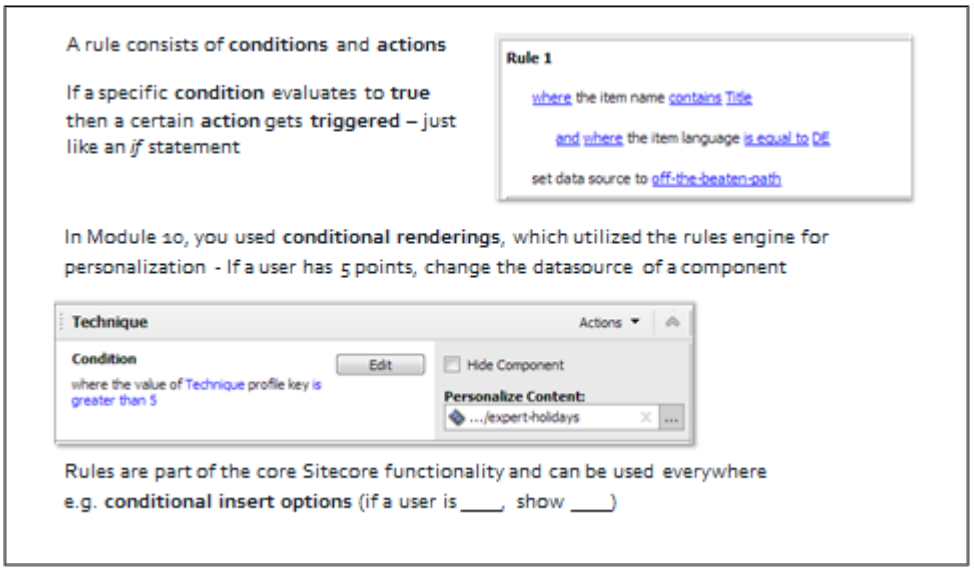
Objectives

By the end of this topic you will be able to:

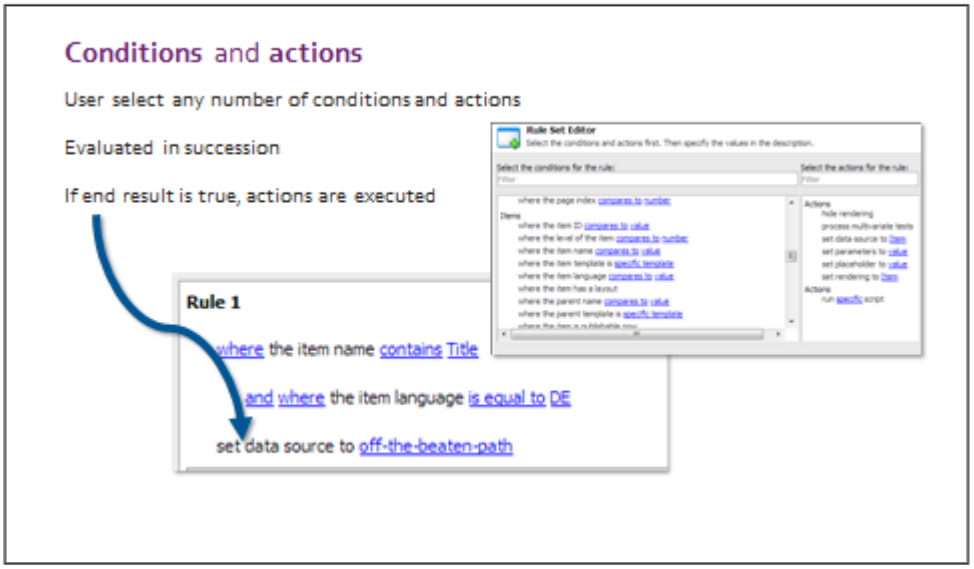
- Define a rule.
- Name the two fields that define a rule.

Content

Sitecore Rules Engine & Rules



How Does a Rule Work for Authors?



Walkthrough: Implementation of Insert Options Rule

In the following walkthrough, we will:

- Create an insert options rule that adds *Simple Link* as an insert option to any item whose name equals *Links*.

1. Create a new Insert Option Rule called ***Link***.
2. In the Name field enter *Links.*
3. In the Rule field enter:
   Where the item name is equal to Links.
   add Simple Link insert option
   (*/BaseCore/Global Content/Simple Link*).
4. Under: */sitecore/content/sitecore-cycling-holidays/Global* create a Links item using the common Sitecore folder.
5. Notice the insert option includes Simple Link.
6. There are many other conditions and actions that are useful regardless of what the purpose or context of the rule is. Those are defined under a common folder: */sitecore/system/settings/rules/common/Conditions/Items*. Find *Item Name*, which is the rule we used in step #3.

Business Use Case

- *Ecommerce website visitors have a shopping cart and based on what they have purchased, you can change what is displayed in a component.*
- *Have a tree structure for news articles, for example, Year, Month, Day. When an author creates a new news article at the top level of this structure. Using a rule, this article can be slotted into its corresponding folder.*
- *Hide Insert options based on what level the tree is on.*
- *You may wish to boost items when indexing data by. For example, you might use a rule to apply a different boost depend on an item's level in the hierarchy: http://www.sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog/Posts/2013/05/Sitecore-7-Boosting-Items-By-Depth-with-the-Rules-Engine.aspx*

Extend

- **Rules engine cookbook**
  http://sdn.sitecore.net/Reference/Sitecore%206/Rules%20Engine%20Cookbook.aspx

Topic 11.5 Placeholder Overrides

Introduction

Objectives

By the end of this topic you will be able to:

- Override a placeholder setting for a particular item.

Content

Demo: Create Placeholder Setting Override

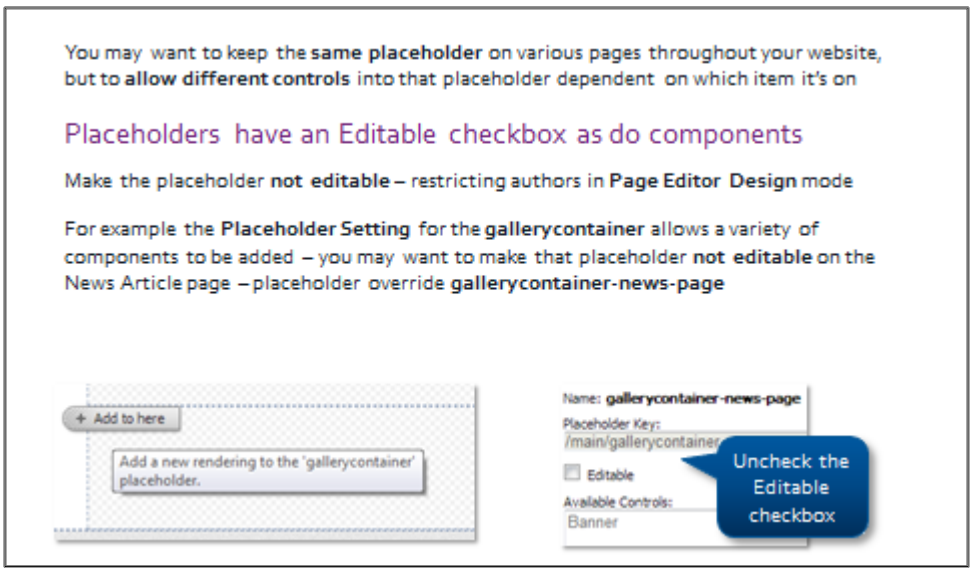
In the following walkthrough, we will:

- Create a new placeholder control in the mark-up.
- Create a **placeholder override,** which in effect is a duplicate of the original placeholder, with different "allowed controls" – per instance of an item. You can also set it up on the standard values so that every item would get this override.

1. In **Page Editor** look at the **Home item** – component called ***Gallery*** in the **gallerycontainer** placeholder.
2. Notice that you can add **banner** and **gallery** component to that placeholder.
3. Go to **another item** and show that you can still add **banner** and **gallery** to the **gallerycontainer** placeholder.
4. Let's say for this page you are not allowed to add a gallery and you only want static image to appear. Navigate to the gallery container placeholder, click **Create New Settings** call it *gallerycontainer-banner-only* and edit the allowed controls to allow **Banner** to be the only component.

5. **Save** the page.
6. Notice the placeholder name changed to: *gallerycontainer-banner-only.*
7. Now when you click **Add to here** you only see **Banner.**
8. **Navigate** back to **home,** show the **old placeholder** with its options.
9. Switch to the **Content Editor** and show the **new placeholder** item that was created in the background**.**

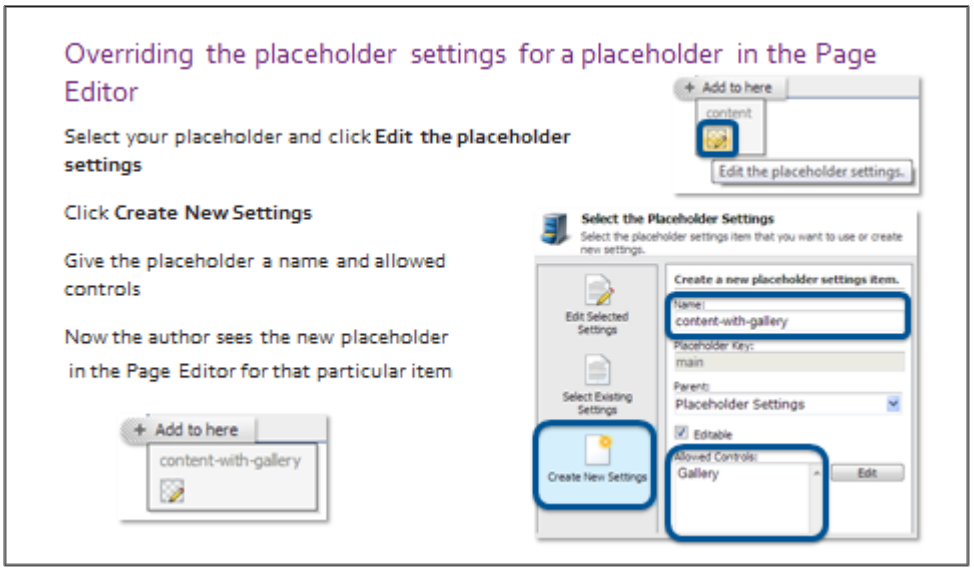
What are Placeholder Overrides Used For?



Tip

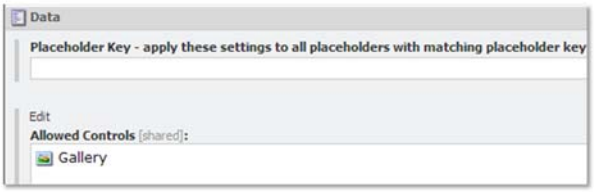
Generally, you tend to specify overrides on the item's template's standard values, but you can specify placeholder overrides per item.

How to Override a Placeholder Setting



In the Content Editor

You get a placeholder settings item with a content-with-gallery blank placeholder key showing the allowed controls



Why a blank key?

This settings item **only** acts as a placeholder override – it does not define a new placeholder key in the markup

If you fill in Placeholder Key, Sitecore will not know which settings item to choose

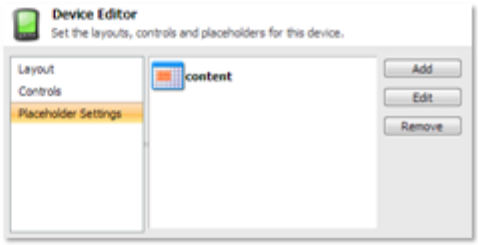
Where to Override Placeholder Settings

On the instance of an item

Create a setting for one page to prevent authors changing components, like the gallery on the Cycling Holidays homepage

On the Standard Values

If you want the **placeholder override** to appear **on all items** instead of just once, you can override placeholders on the template **standard values** using the **Layout Presentation Details** dialog



Apply – Topic 11.5 – 20 min (OPTIONAL)

Create Placeholder Setting Override

It is important to ensure that presentation details are configured correctly for the Page Editor. This involves setting up placeholder settings and placeholder settings overrides.

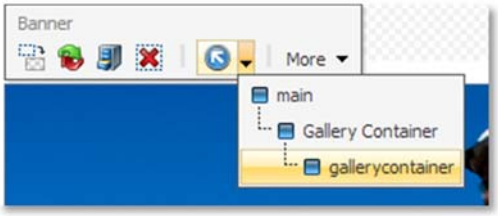
A scenario for overriding a placeholder

- You have created a holidays page, and it has used the global placeholder '*gallerycontainer*'
- For the holidays page, you *only* want to allow authors to put the 'banner' component on the page. But the gallerycontainer placeholder has no restrictions on it; it allows authors to add a Banner and a Gallery component.

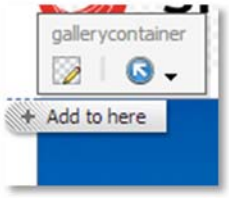
You want to set this restriction for all holiday pages - globally.

A change request has come in to restrict authors to be able to add only the **Banner component** to the **gallerycontainer** placeholder for all the **holiday pages.**

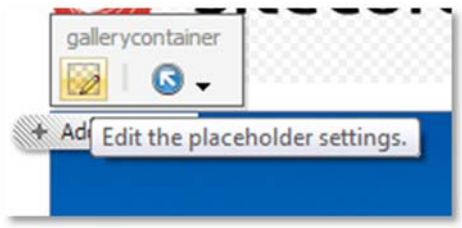
1. Open **Traincore**. In the Page Editor navigate to: */Sitecore-cycling-holidays/Home/holidays/battle-of-the-hills*
2. Make sure you are in **Design** mode.
3. On the holiday page, click the **Banner** component, and navigate to the **gallerycontainer** placeholder using the arrow icon.



4. Click **Add to here.**

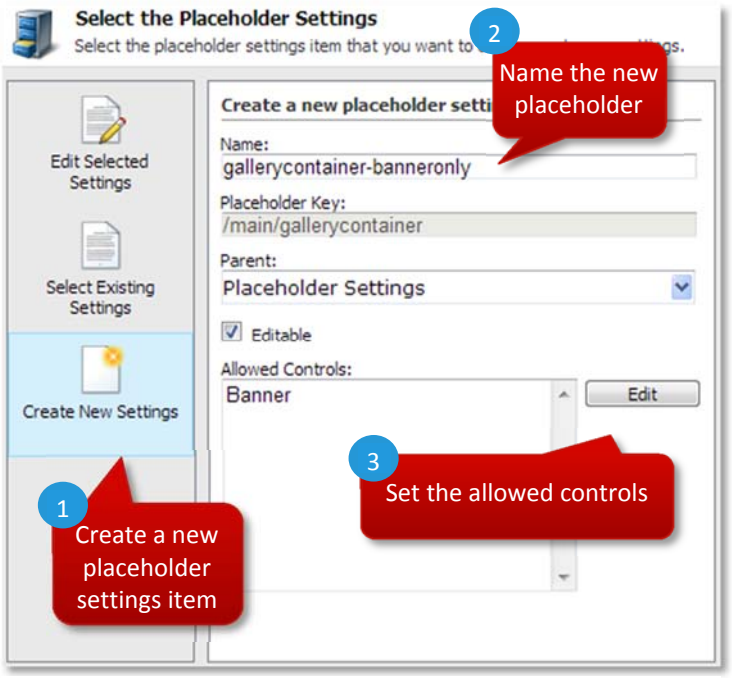


5. As an author you are allowed to add only a Gallery or a Banner component into this placeholder – click **cancel** – think of it as Insert Options for a placeholder.
6. Select the **gallerycontainer** placeholder again and, this time, click the command on the **ribbon** to **Edit the placeholder settings**.

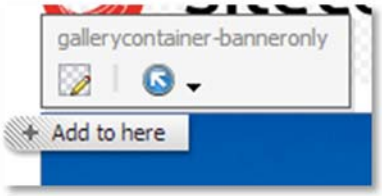


7. Click the **Create New Settings** tab.

8. Name the new placeholder ***gallerycontainer-banneronly.***
9. In the allowed controls, you can specify the **allowed controls** that this **placeholder** can have. Set the **Banner** component as an allowed control. The path is: *Layout/Renderings/BaseCore/Banner* and click **OK**.
10. **Save** the item.



11. Confirm that the placeholder used now is the new one.



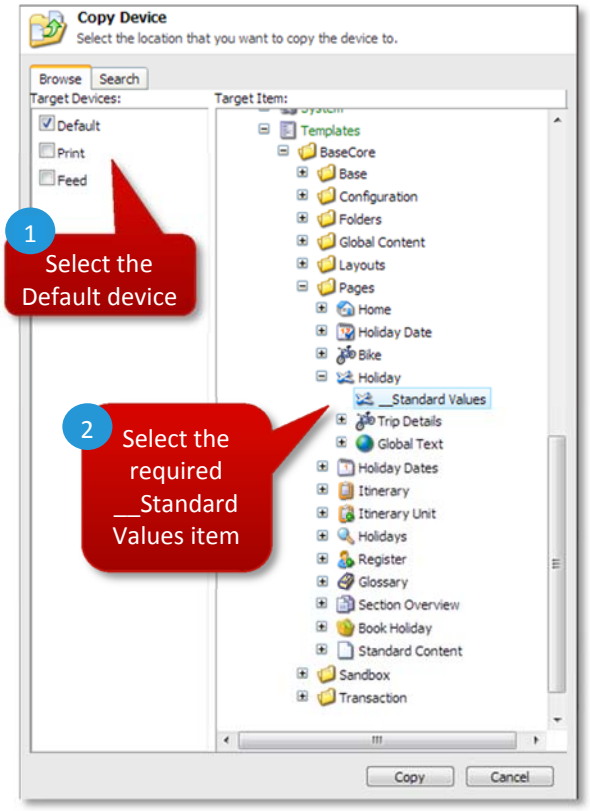
12. Click the **Add to here** button to confirm that only **Banner** is the only allowed control.

Lab B. Copy the Placeholder Override to the Standard Values

1. Stay in the Page Editor. On the ribbon, select the **Advanced** tab and click **Details**.

2. For the **Default** device, click **Copy To** at the bottom of that device.
3. Check the **Default Device** checkbox and select the standard values for that holiday page. The path is: */Templates/BaseCore/Pages/Holiday/__Standard Values.*

4. Click **Copy** then **OK** in the **Layout Details** dialog.



5. If you add a new holiday page, it will automatically get the **gallerycontainer-banneronly** placeholder.

Tip

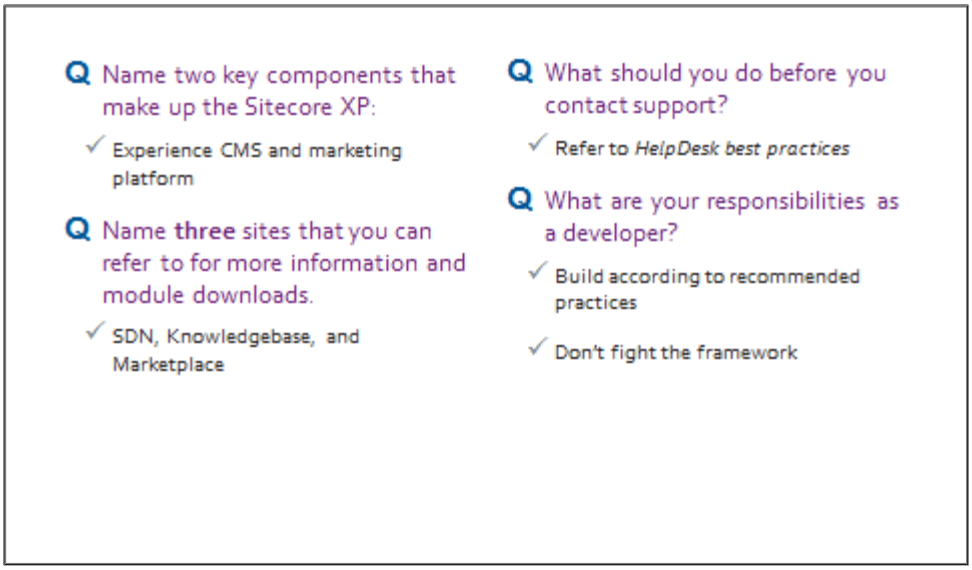
If you want a placeholder override to apply to all items based on the holiday data template, modify the presentation details on the holiday data template's standard values. Add a new placeholder setting that overrides gallerycontainer; this change will then be reflected on all news items.

Review

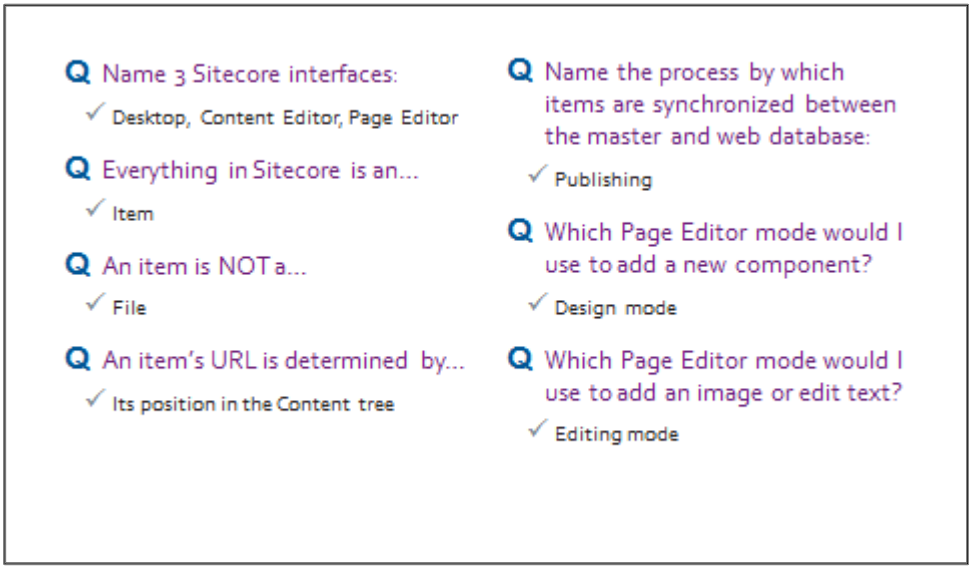
Contents:

Module 1 Sitecore Overview

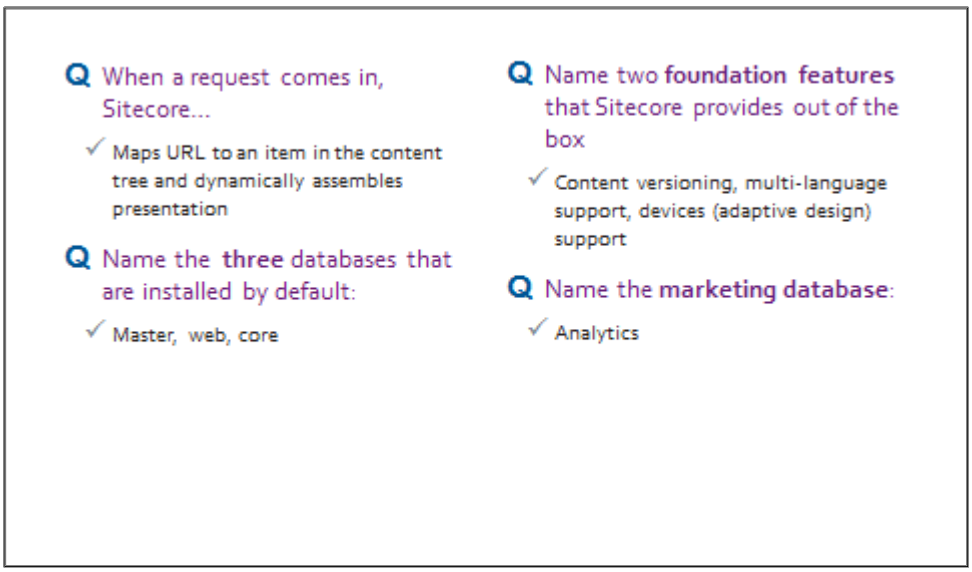
Topic 1.2 What Is Sitecore



Topic 1.3 Sitecore from an Author's Perspective

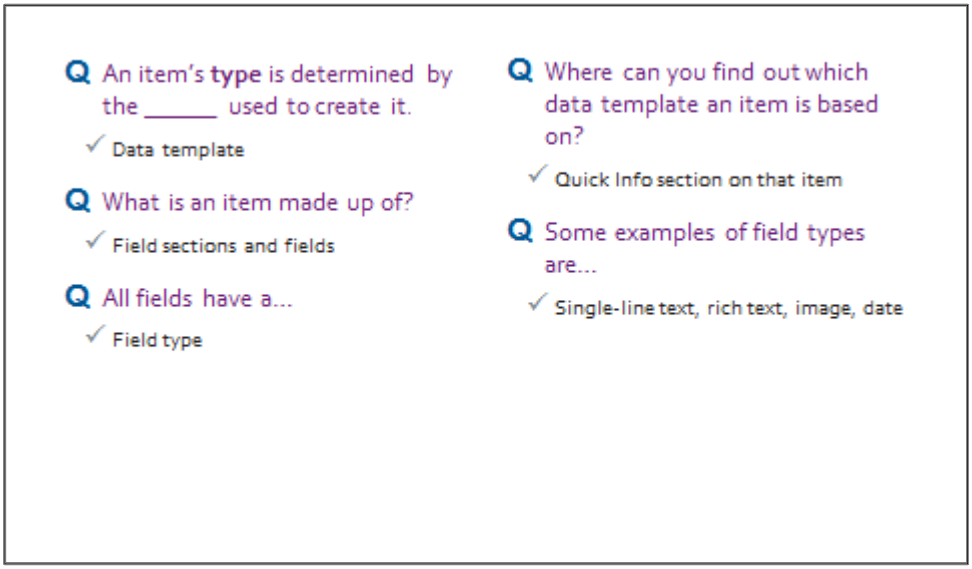


Topic 1.4 Sitecore from an Developer's Perspective



Module 2 Defining Data

Topic 2.1 Defining Data Structure



Topic 2.2 Data Template Inheritance

Q In what scenario would you use data template inheritance?

- ✓ When you have fields that will be reused in multiple data templates

Q Why is it important to think of your data template creation and inheritance structure from the beginning?

- ✓ You should not have to be deleting fields / field sections, values get lost

Q By default what data template do all data templates eventually inherit from?

- ✓ Standard Template

Q What happens if the same field section name is used more than once in a data template's inheritance tree?

- ✓ The field sections will merge into one, and all fields will be listed under this single section.

Q What happens if the same field name is used in two separate inherited data templates?

- ✓ Unlike field sections, actual fields do not merge. Both fields will appear, but one will take precedence over the other.

Topic 2.3 Standard Values

Q Describe the standard values **item**.

- ✓ Always named **__Standard Values**
- ✓ Child item of owning data template
- ✓ Looks like an **instance** of an item – contains all inherited fields

Q What type of settings can be applied to standard values?

- ✓ Default values, tokens, default insert options, *default presentation*, *default workflow state*

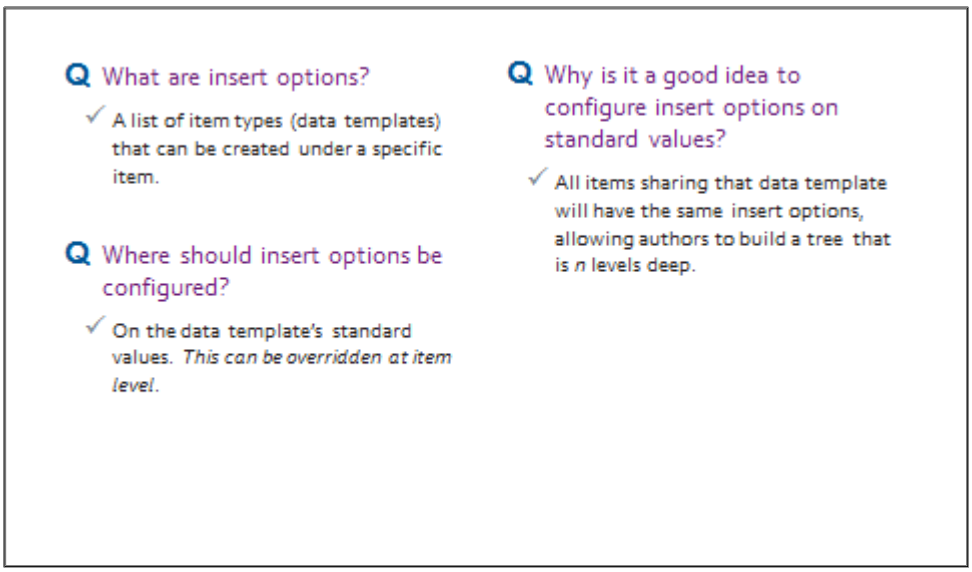
Q How do tokens work?

- ✓ Tokens are replaced when an item is created. Depending on the token, it may be the name of the item or the ID of the parent.

Q Name three tokens:

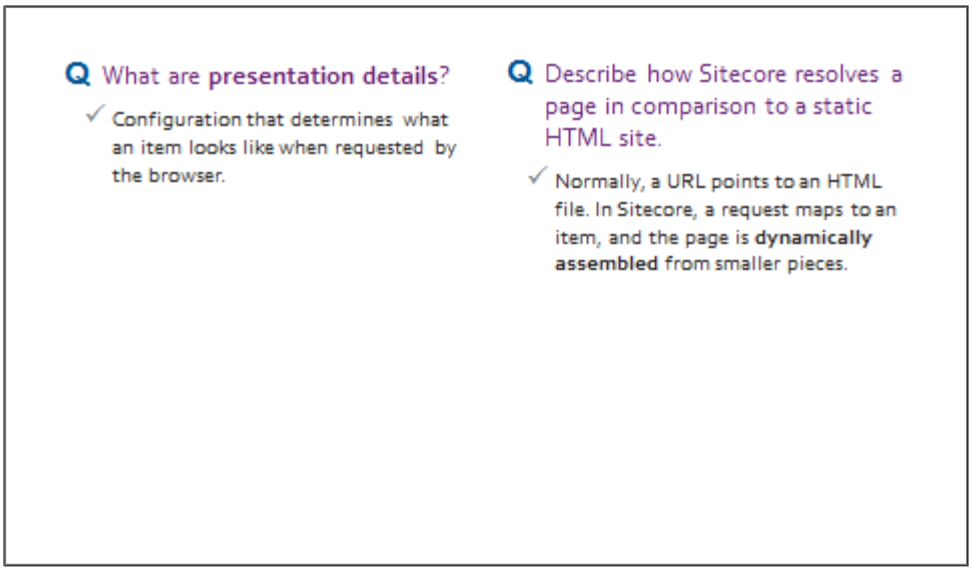
- ✓ $name, $date, $id, $parendid, $parentname, $time, $now

Topic 2.4 Insert Options

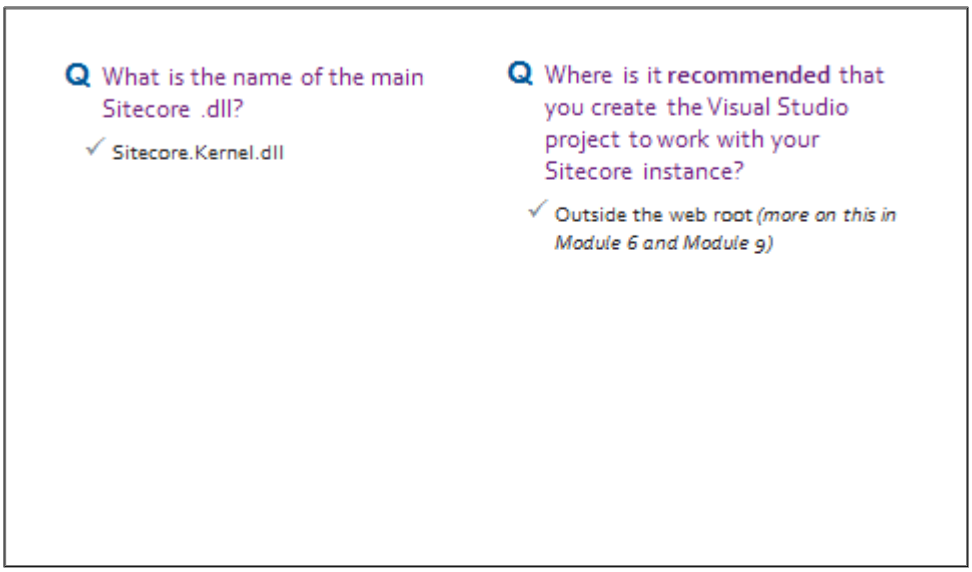


Module 3 Presentation

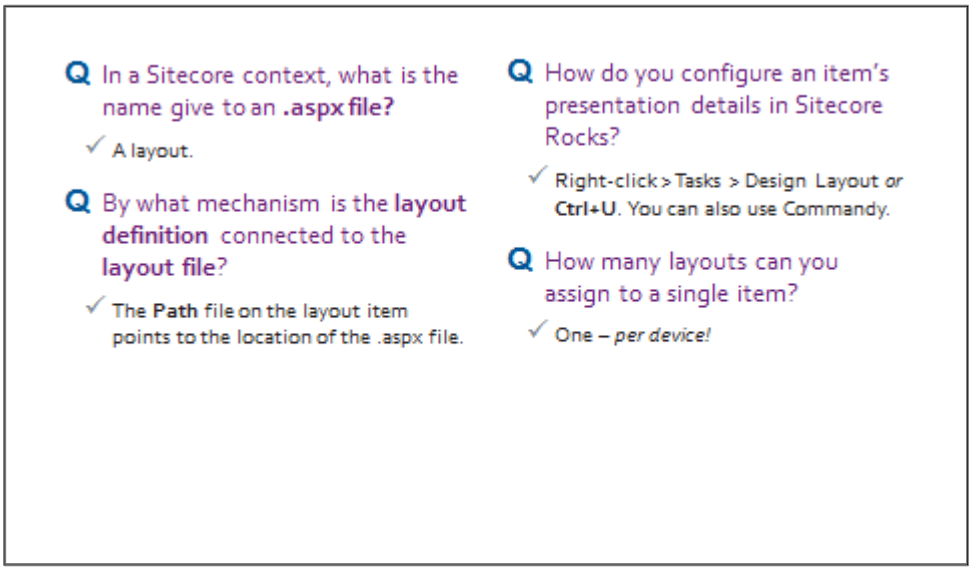
Topic 3.1 Presentation Is Dynamic and Modular



Topic 3.2 Preparing to Build



Topic 3.3 Creating a Layout



Topic 3.4 Creating Components

Q Name 3 different types of Sitecore components

✓ Sublayout, XSLT rendering, web control.

Q What type of file is a Sitecore sublayout?

✓ .ascx

Q A component consists of a file on the file system and...

✓ A definition item in Sitecore

Q By what mechanism are a component's *file* and *definition item* linked together?

✓ By a **Path** field on the definition item.

Topic 3.5 Dynamic Binding

Q What is the name of the Sitecore component that allows you to bind components to a page?

✓ A placeholder.

Q Name the property that you must set on a placeholder to identify it.

✓ Key

Q Describe the process by which a component is bound to a placeholder?

✓ Open **presentation details** > Add Rendering > Select rendering > Edit **PlaceholderKey** property

Q If *all* items of a particular type require a particular component, where should these presentation details be configured?

✓ On the standard values of the data template.

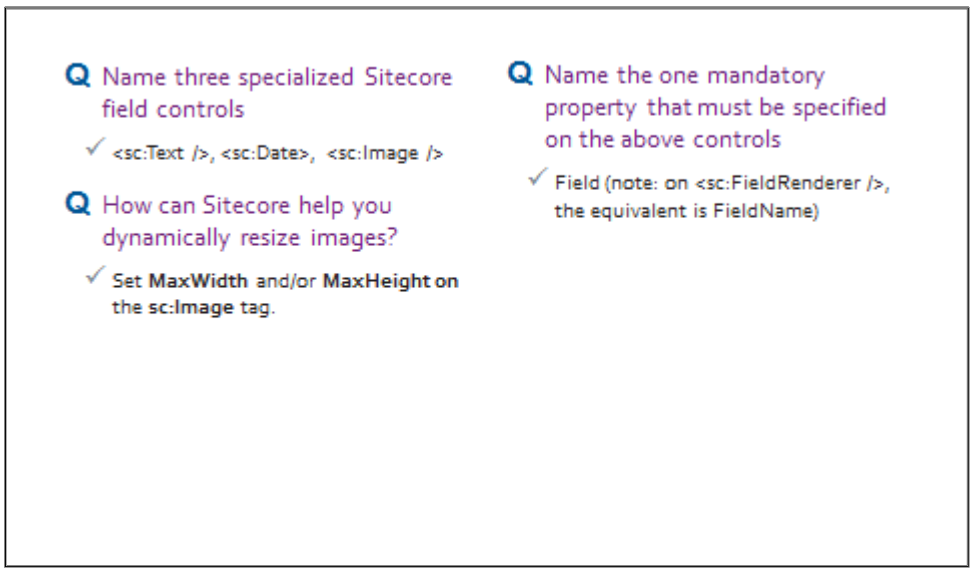
Q What is the control that can be used to statically include sublayouts?

✓ `sc:Sublayout`

Q What type of components *might* be good candidates for static binding?

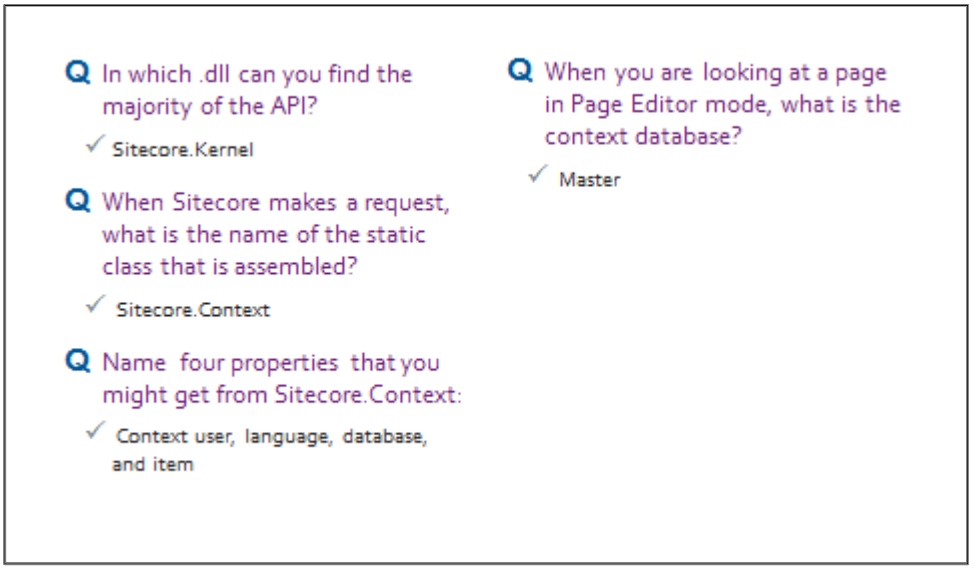
✓ Page scaffolding, like headers and footers.

Topic 3.6 Outputting Content

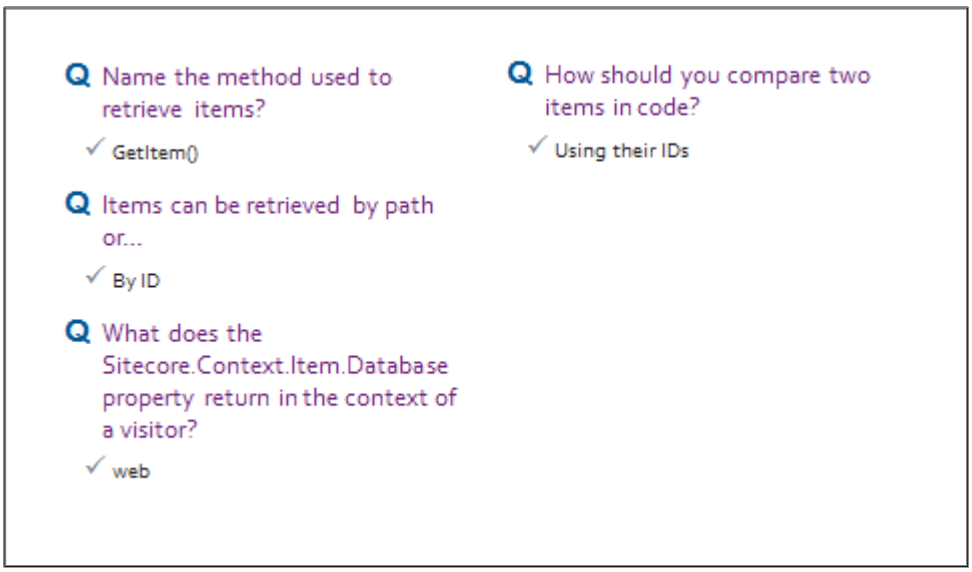


Module 4 Sitecore API

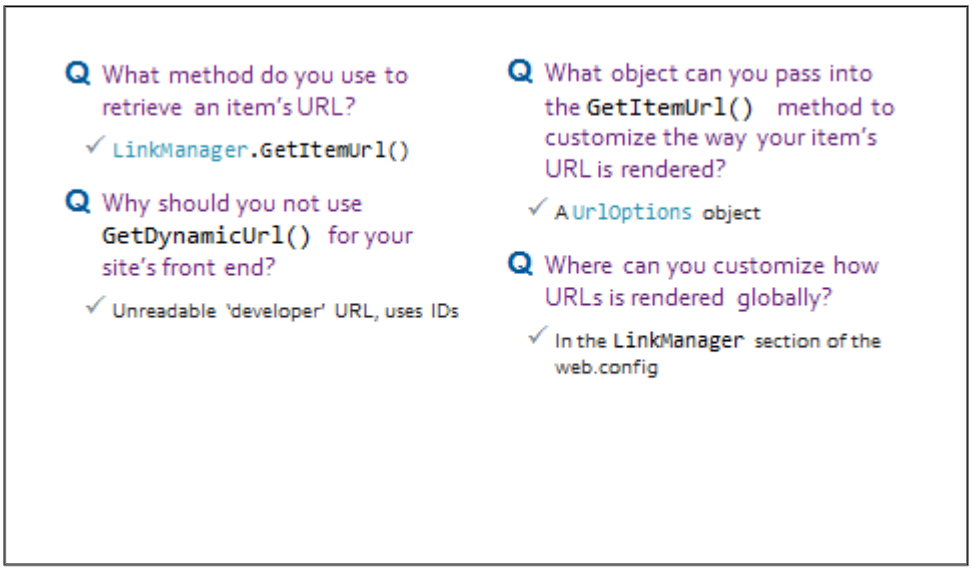
Topic 4.1 Basic API Concepts



Topic 4.1 *(continued)* Retrieving Items



Topic 4.2 Item Links



Topic 4.3 Creating, Deleting and Modifying Items

Q Sitecore security is an extension of...

✓ Standard .NET membership

Q All code is executed in the context of...

✓ The current user

Q What affects whether or not a piece of code will run?

✓ That user's permissions

Q What permissions does extranet\anonymous lack by default?

✓ Permission to create or edit items

Q How can you force a piece code to run even though the currently logged in user does not have the appropriate permissions?

✓ Recommended practice is to use the `UserSwitcher` to run in the context of a user that does have permission. The `SecurityDisabler` may also be used.

Q When creating or editing an item, the code must be executed with the appropriate security rights. Name two ways that this can be done.

✓ UserSwitcher or SecurityDisabler

Q How do you put an item into an editing state?

✓ `Item.Editing.BeginEdit();`

Q Why should you create / edit items in the master rather than the web database?

✓ Web overwritten by publish

Q What property do you use to update?

✓ `Item.Editing.EndEdit();`

Q Why should you not output a field's value straight to the screen?

✓ Not editable in Page Editor and complex fields contain custom XML

Q Which field types are suitable for editing using the `.Value` property?

✓ Simple text fields – e.g. Single-Line Text

Topic 4.4 Working with Complex Fields

Q Why should you never output raw value to screen?

✓ Not editable in Page Editor, complex field types will not make sense – e.g., image

Q Name the best suited field object for the following field types: single-line text, treelist, droplink, general link:

✓ Field, MultilistField, ReferenceField, LinkField

Q What method should you use to render the contents of text, date, image, and link fields to the screen (and why)?

✓ FieldRenderer.Render() – Page Editor support, transforms custom XML, transforms RichText links

Q Why can't you render a multilist field to the page using .Render()?

✓ It contains IDs, not readable content

Module 5 Advanced Presentation Concepts

Topic 5.1 Datasources

Q Setting a component datasource allows that component to _____.

- ✓ Output content from a content item elsewhere in the tree.

Q How can you force a Sitecore control to output the content of an item's datasource?

- ✓ Set `DataSource` property in code-behind
- ✓ Set `Item` property in code-behind
- ✓ Set `DataSource` property on the control

Q What do parameters allow you to do?

- ✓ Allows properties to be set per instance of a component

Q Which property do you use to retrieve parameters?

- ✓ `.Parameters` property

Q What utility converts parameter lists to a **NameValueCollection**?

- ✓ `WebUtil.ParseUrlParameters()`

Q Why would you set the datasource on a component on the item itself?

- ✓ When your component and its datasource appear on one page

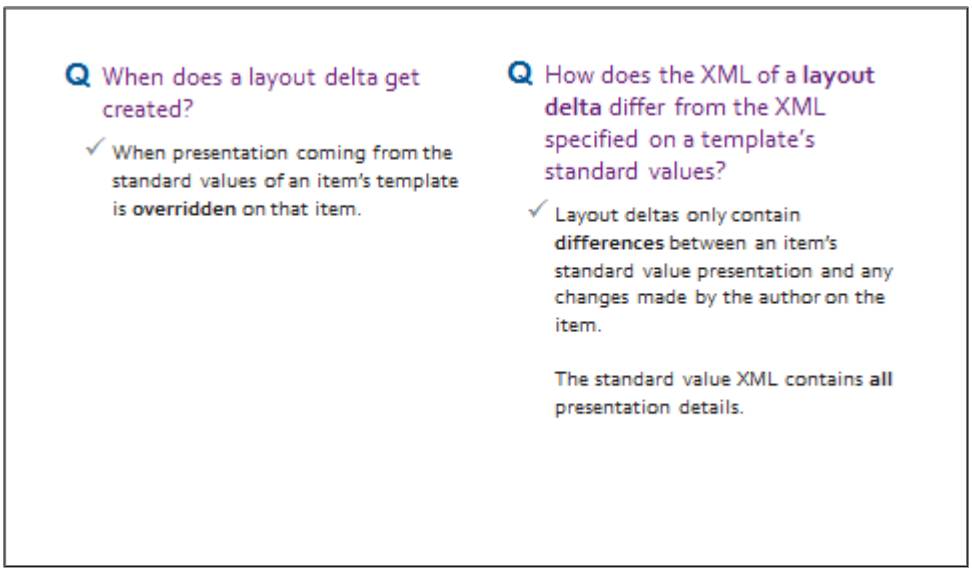
Q Why would you set the component and the datasource on the standard values?

- ✓ When you want all items to have that component with that particular datasource

Q Why would you set the component on the standard values and override the datasource on the item?

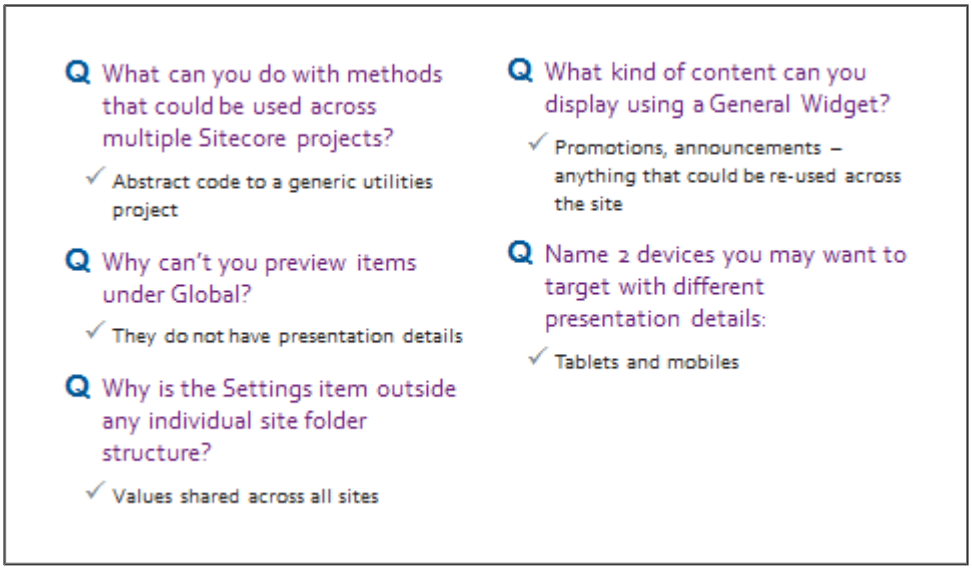
- ✓ When your component has to appear on every page but has a different datasource each time

Topic 5.2 Layout Deltas

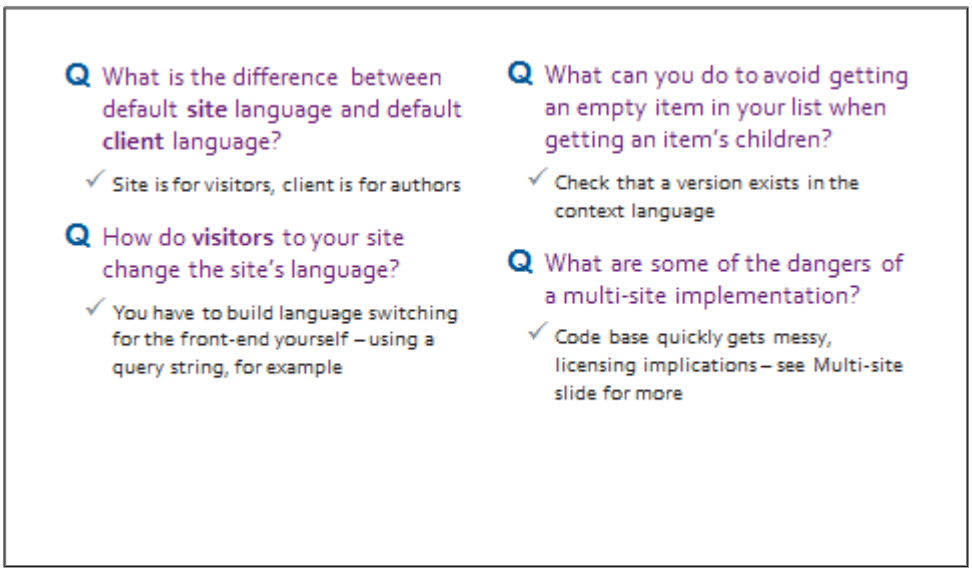


Module 6 Real World Solutions

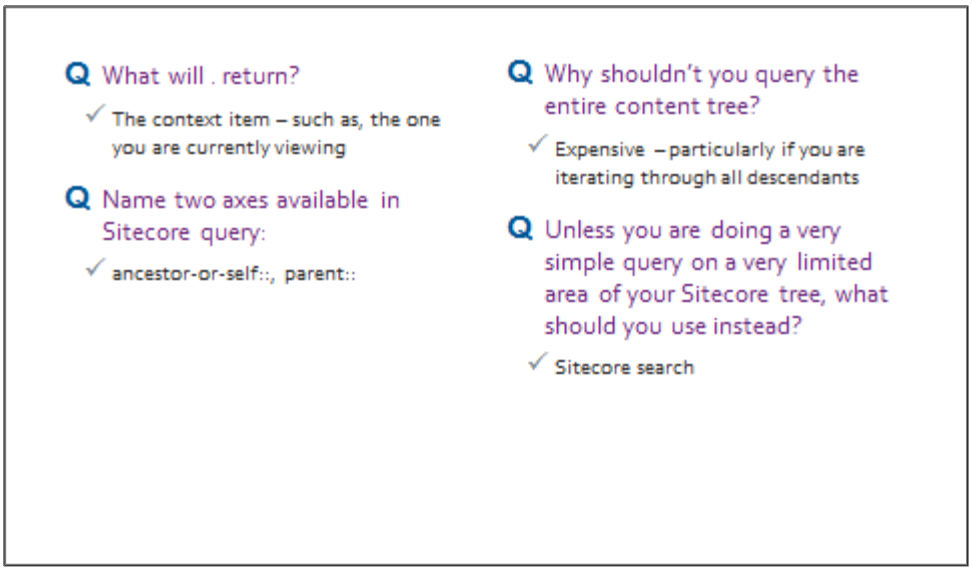
Topic 6.1 Familiar Concepts



Topic 6.2 Dealing with Larger Sites

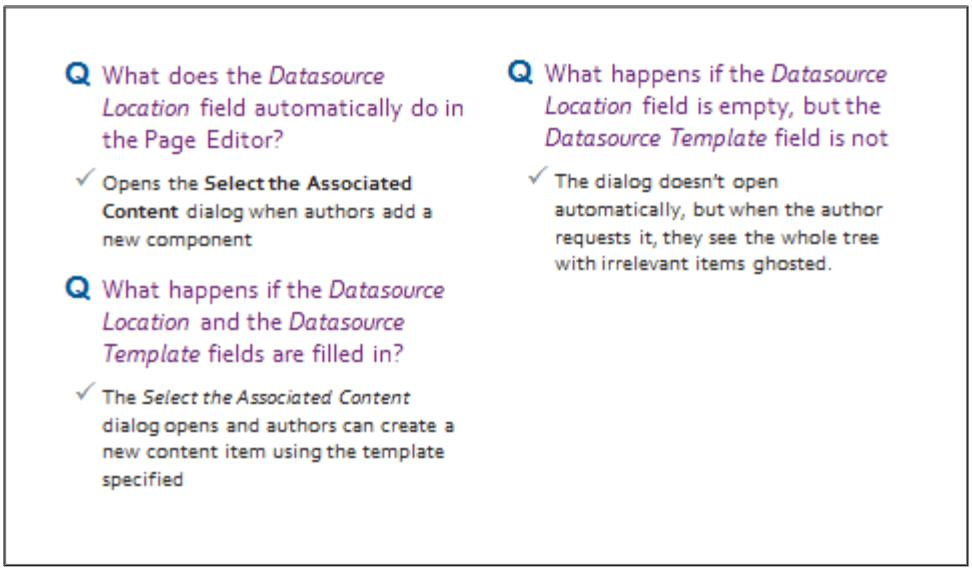


Topic 6.3 Sitecore Query

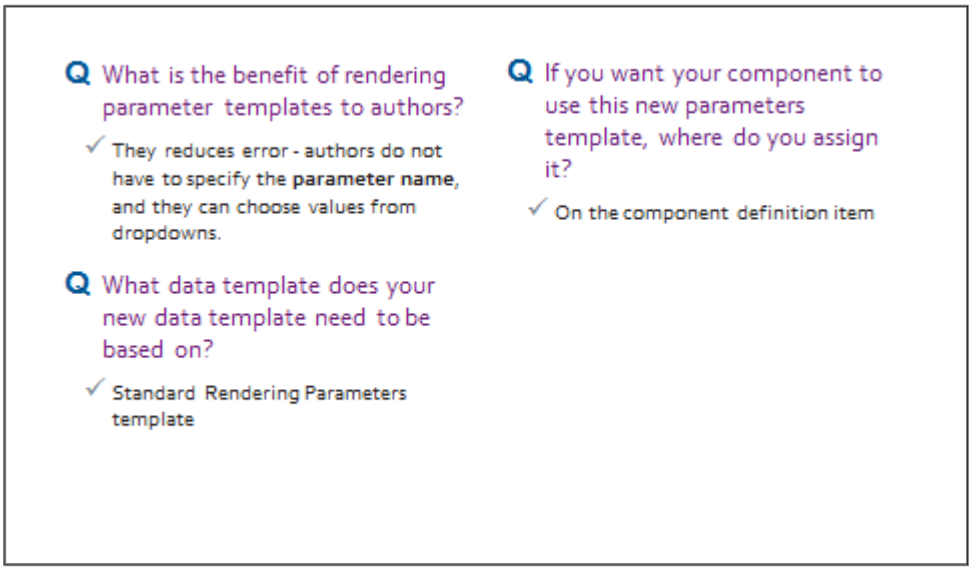


Module 7 Configuring the Page Editor

Topic 7.1 Datasource Restrictions



Topic 7.2 Parameters and Parameter Templates



Topic 7.3 Placeholder Restrictions

Q When you add <sc:placeholder key="myKey" runat="server" /> to your sublayout, what must you do for the placeholder to be selectable in the Page Editor?

✓ Create a placeholder settings definition item in Sitecore.

Q What else do these placeholder settings items allow you to specify for authors?

✓ Which components authors can insert into a placeholder.

Topic 7.3 *(continued)* Compatible Renderings / Allowed Controls

Q When you add <sc:placeholder key="myKey" runat="server" /> to your sublayout, what must you do for the placeholder to be selectable in the Page Editor?

✓ Create a placeholder settings definition item in Sitecore.

Q What else do these placeholder settings items allow you to specify for authors?

✓ Which components authors can insert into a placeholder.

Topic 7.4 Advanced Page Editor Configuration

Q Once Custom Experience Buttons are assigned, where do they show up?

✓ On a component / field and displayed in the Page Editor

Q Name two examples of Custom buttons:

✓ Field Editor and WebEdit button

Q On what type of definition item can you assign Custom Experience buttons?

✓ A component definition item or a field definition item

Q What do Edit Frames do?

✓ Surround a particular area on a page or component and display buttons allowing you to edit fields that would not be normally editable in the Page Editor

Q In which database do you create the Custom Experience buttons and edit frames?

✓ Core

Module 8 Dealing with Your Data

Topic 8.1 Item Buckets

Q Use a bucket when...

✓ You do not care about having a hierarchical item structure and/or you have a large number of items

Q In order to store an item in a bucket, you must...

✓ Make that item or the data template's standard values it is based on **bucketable**

Q After making changes to the bucketability of the items in a bucket, you must...

✓ Sync the bucket

Q What do facets allow you to do?

✓ Progressively apply filters (based on fields) to narrow down your result set

Q How do you create a bucket?

✓ Select the item you want to turn into a bucket and click the 'Bucket' command in the Configure tab

Topic 8.2 Search

Q What defines how Sitecore items should be indexed?

✓ Index configuration files in /App_Config/Include

Q Search is built using a provider model. What does this mean?

✓ You can plug in whatever search provider you want

Q What syntax do you use to query an index?

✓ LINQ

Q What is the name of Sitecore's default search result class?

✓ `SearchResultItem`

Q When building your own search result class, how do you account for fields with spaces?

✓ Decorate with `[IndexField("Page Heading")]`

Q What method should you use to return a rich results object?

✓ `.GetResults();`

Q You can index fields by **type** or...

✓ Name

Q You can include/exclude by individual field names or...

✓ Item templates

Q Which attribute must be set to 'YES' in order for values to be stored in the index?

✓ storageType

Q Which two search providers does Sitecore ship with?

✓ Lucene and Solr

Q If you wanted to store the number of comments a news article has, what type of field might you use to perform the calculation?

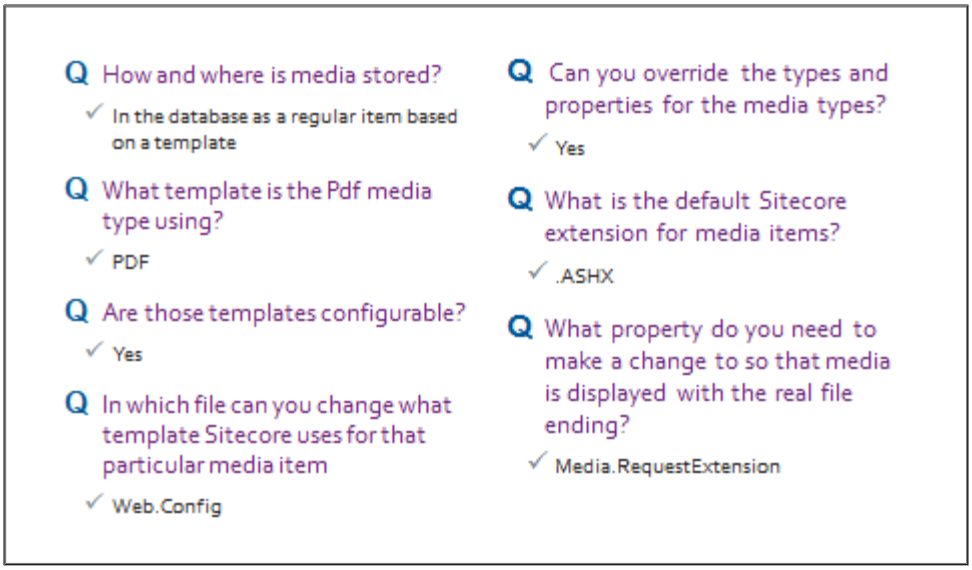
✓ A computed field

Q Why should you tune your index configuration and not index everything by default?

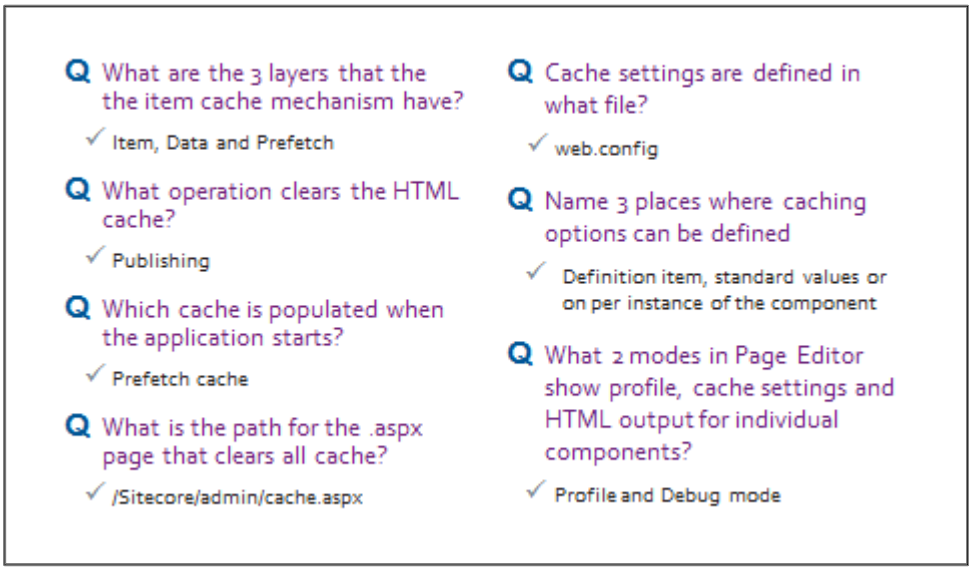
✓ Large, unwieldy index as your solution grows

Module 9 Recommended Practices

Topic 9.1 Working with Media



Topic 9.2 Caching and Performance



Topic 9.3 Publishing

Q Name three publishing modes

✓ Incremental , smart and re-publish

Q Publishing restrictions can be applied to _____ , _____ and _____

✓ Numbered versions of items , the item itself and targets

Topic 9.4 Installing and Scaling Sitecore

Q List some benefits of using Sitecore Installer

✓ Checks prerequisites, correct DLLs, logs, remove an existing installation

Q Can you do a manual installation?

✓ Yes

Q What is the minimum recommendation for production environment?

✓ CM with core, master and web

✓ CD with core and web

✓ Separate DB server

✓ Use scaling guide

Q Name some features of Sitecore Instance Manager

✓ Packages, customize, extend, reinstall &remove instances

Q Name some features of Sitecore Rocks

✓ Packages, pick steps to include and remove instances

Q Where is the Include folder?

✓ App_Config

Q Where can you view the merged web.config?

✓ /sitecore/admin/showconfig.aspx

Topic 9.5 Team Development and Development Environment

Q Name two development models

✓ Inside and outside the web root

Q Which is the best practice?

✓ Development outside the web root

Q What are the options to source control your items?

✓ You can serialize your items and files or use TDS

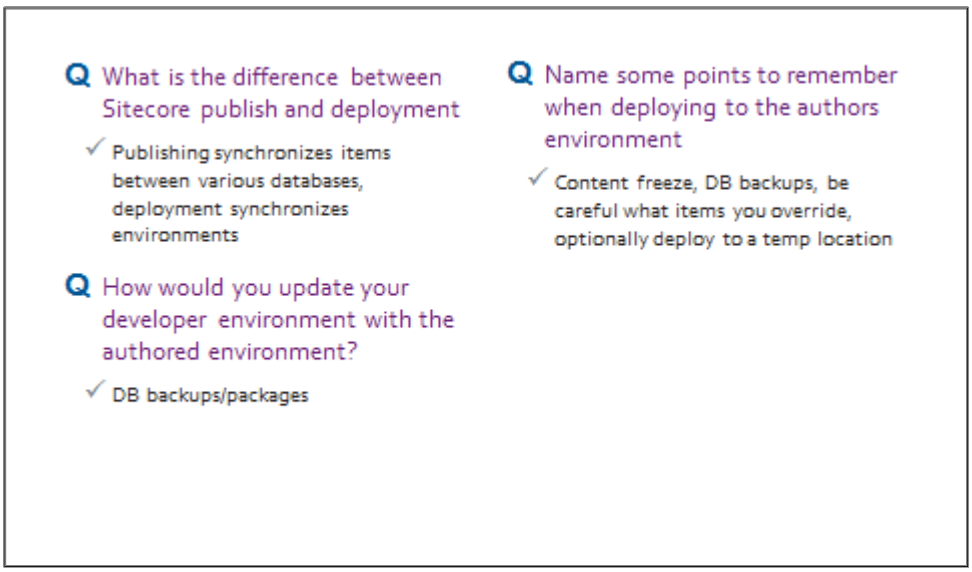
Q What are Sitecore packages and what do they contain?

✓ zip files that contain items and code files

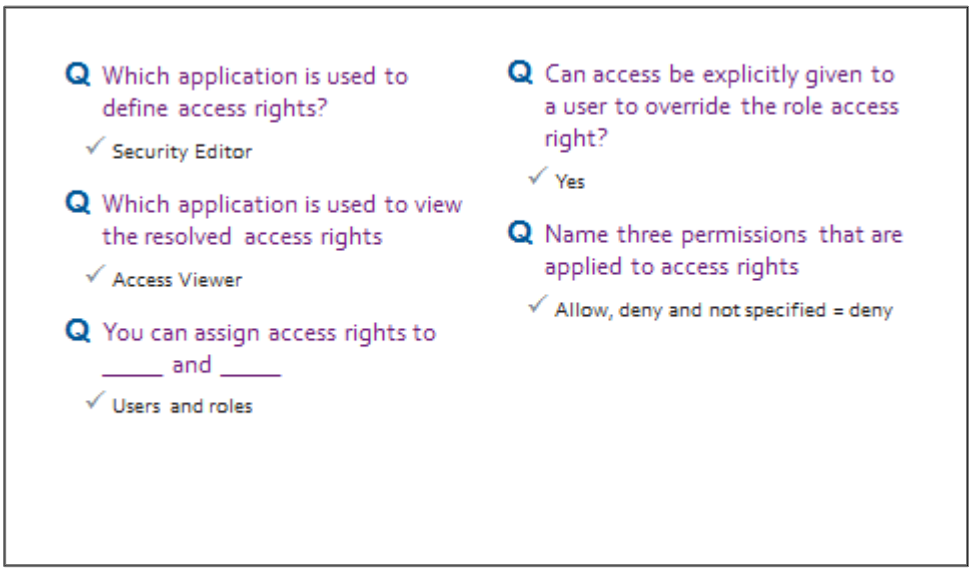
Q What is the name of the .aspx that you go to when upgrading Sitecore?

✓ */sitecore/admin/updateinstallationwizard.aspx*

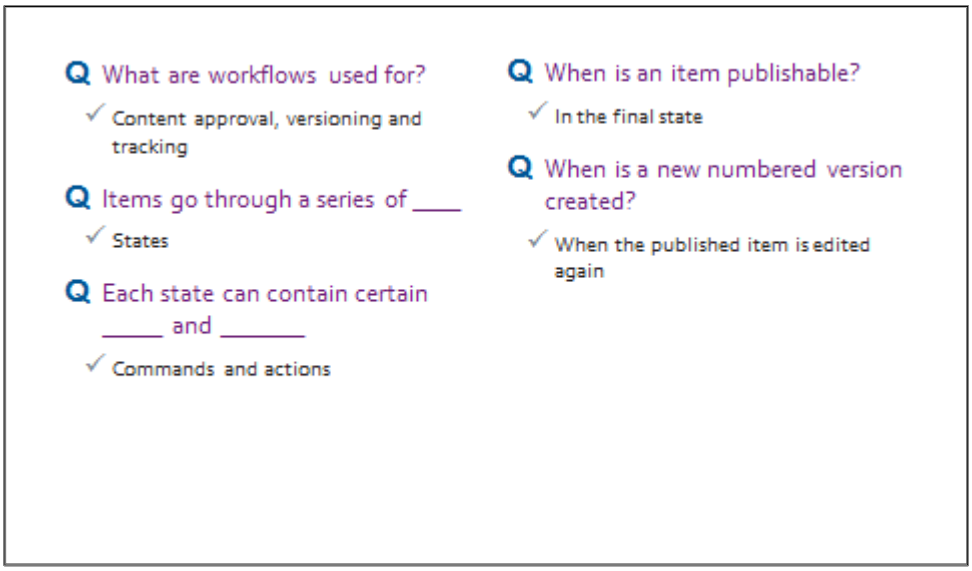
Topic 9.6 How to Deal with Deployment



Topic 9.7 Security

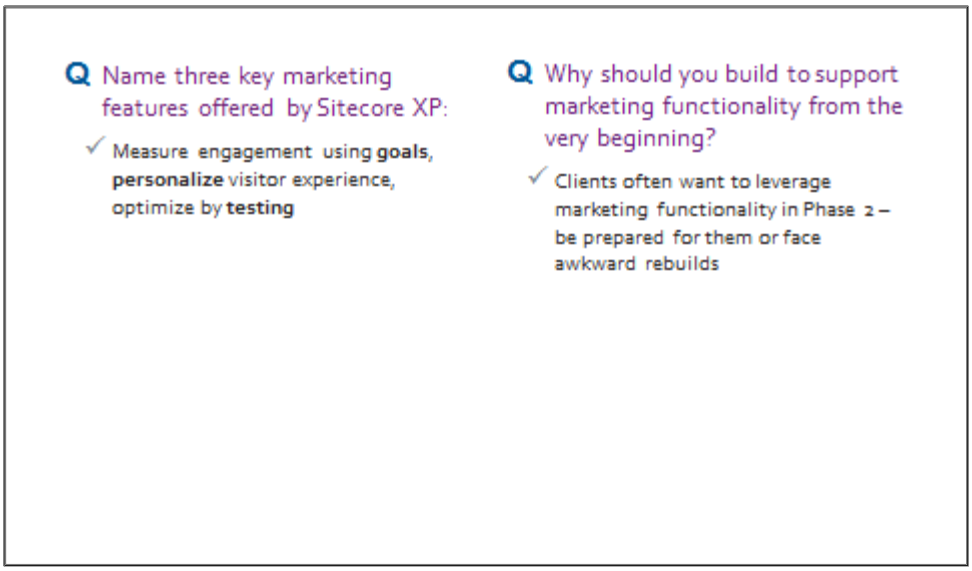


Topic 9.8 Workflows

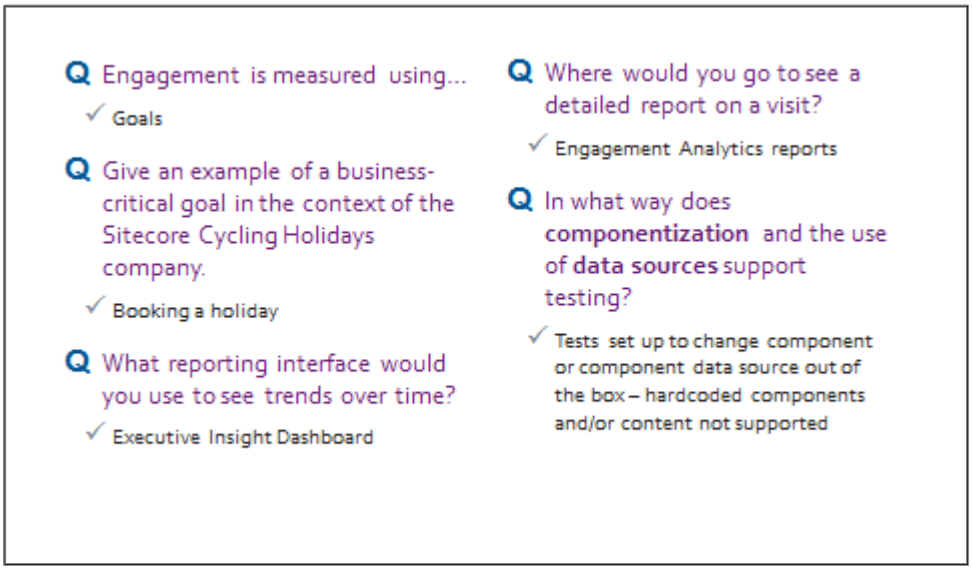


Module 10 Marketing Functionality

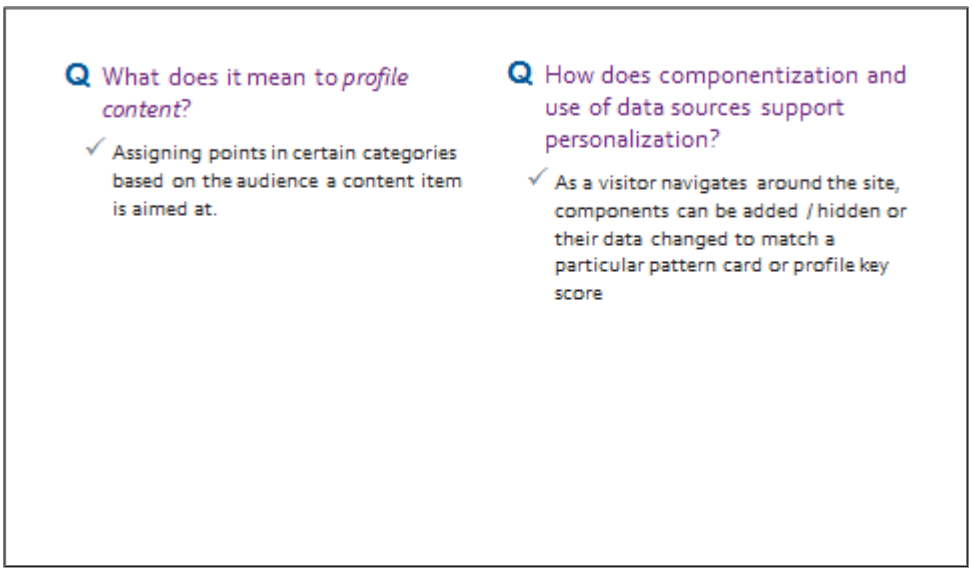
Topic 10.1 Introduction to the XP



Topic 10.2 Engagement Value and Goals



Topic 10.3 Profiling and Personalization



Module 11 Optional Topics

Topic 11.1 Branch Templates

Topic 11.2 Other Item and Template Properties

Q Items exist in a ___ and ___version.
- ✓ Language and numbered version

Q What does unversioned for field versioning mean?
- ✓ Single value per language

Q What *does shared for field versioning* mean?
- ✓ Single value for all numbered versions in all languages

Q What is the default field versioning?
- ✓ Versioned – different values for all languages and numbers

Q What is the field called where you set up field sorting?
- ✓ Sortorder

Q Where is this field found?
- ✓ Standard fields

Q On what type of item do you do field sorting?
- ✓ On the field definition item

Q Name two fields that you can use to help a multi-lingual audience
- ✓ Long Description and Short Description fields - also part of the standard fields

Topic 11.3 Pipelines and Events

Q What are pipelines
- ✓ Break down operations into multiple configurable steps

Q Where are pipelines defined?
- ✓ In the web.config as <processors /> and <pipelines />

Q In what folder on the file system do you store your pipeline configuration files?
- ✓ The include folder

Q What method do all processors implement?
- ✓ Process()

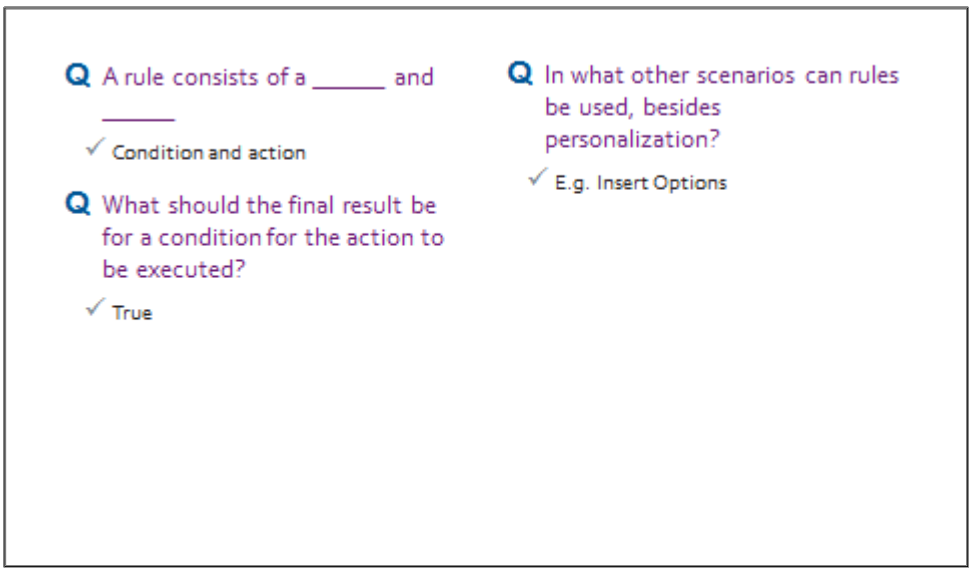
Q What are the 3 options for creating pipelines
- ✓ Add processor to existing pipeline
- ✓ Override and extend existing processors
- ✓ Create new pipelines that are called from code

Q Give an example of an event that you can subscribe to
- ✓ Item saved / created / updated / deleted / moved / renamed / and so on

Q How do you subscribe to events?
- ✓ Using event handlers

Q Where are events defined?
- ✓ In the web.config

Topic 11.4 Rules



Topic 11.5 Placeholder Overrides