

# 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 of 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 Marengo**

Copy Edited Course  
Supported Formatting Efforts  
Provided Final Polish

For the last 15 years, Susan Marengo 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.*

#### Copyright Notice:

Information in this document is subject to change without notice. All example companies, products, domain names, email addresses, logos and people are fictitious. No part of this document may be copied or transmitted electronically with the prior approval of Sitecore. Sitecore has no responsibility for any links to external web sites provided in this document.

# Contents

<b>Module 1</b>	<b>Sitecore Overview</b> .....	<b>8</b>
Topic 1.1	Course Overview .....	9
Topic 1.2	What Is Sitecore? .....	11
Topic 1.3	Sitecore from an Author's Perspective .....	15
Topic 1.4	Sitecore from a Developer's Perspective .....	20
<b>Module 2</b>	<b>Defining Data</b> .....	<b>23</b>
Topic 2.1	Creating Data Templates .....	24
Topic 2.2	Data Template Inheritance .....	30
Topic 2.3	Standard Values .....	39
Topic 2.4	Insert Options .....	44
Topic 2.5	Summary and Optional Lab .....	46
<b>Module 3</b>	<b>Presentation</b> .....	<b>48</b>
Topic 3.1	Presentation Is Dynamic and Modular .....	49
Topic 3.2	Preparing to Build .....	51
Topic 3.3	Creating a Layout .....	57
Topic 3.4	Creating Components .....	63
Topic 3.5	Dynamic Binding .....	67
Topic 3.6	Outputting Content .....	74
Topic 3.7	Summary and Optional Lab .....	76
<b>Module 4</b>	<b>Sitecore API</b> .....	<b>78</b>
Topic 4.1	Basic API Concepts and Retrieving Items .....	79
Topic 4.2	Item Links .....	85
Topic 4.3	Creating, Modifying, and Deleting Items .....	89
Topic 4.4	Working with Complex Fields .....	97
<b>Module 5</b>	<b>Advanced Presentation Concepts</b> .....	<b>105</b>
Topic 5.1	Reusable Content .....	106
Topic 5.2	Layout Deltas .....	115
<b>Module 6</b>	<b>Real World Solutions</b> .....	<b>117</b>
Topic 6.1	Familiar Concepts .....	118
Topic 6.2	Dealing with Larger Sites .....	130
Topic 6.3	Sitecore Query .....	133
<b>Module 7</b>	<b>Configuring the Page Editor</b> .....	<b>137</b>
Topic 7.1	Datasource Restrictions .....	138
Topic 7.2	Parameters and Parameter Templates .....	141
Topic 7.3	Placeholder Restrictions .....	144
Topic 7.4	Advanced Page Editor Configuration .....	147

<b>Module 8</b>	<b>Dealing with Your Data</b> .....	<b>153</b>
Topic 8.1	Item Buckets .....	154
Topic 8.2	Search.....	161
<b>Module 9</b>	<b>Recommended Practices</b> .....	<b>173</b>
Topic 9.1	Working with Media.....	174
Topic 9.2	Caching and Performance .....	176
Topic 9.3	Publishing .....	181
Topic 9.4	Installing and Scaling Sitecore.....	185
Topic 9.5	Team Development and the Development Environment.....	189
Topic 9.6	How to Deal with Deployment .....	195
Topic 9.7	Basic Security.....	197
Topic 9.8	Workflow .....	199
<b>Module 10</b>	<b>Marketing Functionality</b> .....	<b>202</b>
Topic 10.1	Introduction to the Customer Experience Platform (XP).....	203
Topic 10.2	Engagement Value and Goals .....	205
Topic 10.3	Profiling and Personalization .....	211
<b>Module 11</b>	<b>Optional Topics</b> .....	<b>214</b>
Topic 11.1	Branch Templates .....	215
Topic 11.2	Other Item and Template Properties.....	217
Topic 11.3	Pipelines and Events .....	227
Topic 11.4	Rules.....	231
Topic 11.5	Placeholder Overrides.....	233
	<b>Review</b> .....	<b>239</b>

# 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

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

- ✓ Define data and assemble presentation components
- ✓ Use common API features
- ✓ Configure and build for the Page Editor
- ✓ Build for and use basic marketing features such as real-time personalization
- ✓ Deal with complex sites and large volumes of data
- ✓ Use recommended practices around installation, environments, scaling and optimization
- ✓ Extend Sitecore's core functionality
- ✓ Find help and modules



#### Course Format

11 modules



Each module has several topics



Demos and walkthroughs



Apply – Labs



Review – Questions



Extend



## Module Overviews

<p><b>Module 1 Sitecore Overview</b></p> <p>Course overview</p> <p>What is Sitecore?</p> <p>Sitecore from an author's perspective</p> <p>Sitecore from a developer's perspective</p>	<p><b>Module 2 Defining Data</b></p> <p>Creating data templates</p> <p>Data template inheritance</p> <p>Standard values</p> <p>Insert options</p> <p>Summary and optional lab</p>
<p><b>Module 3 Presentation</b></p> <p>Presentation is dynamic and modular</p> <p>Setting up a Visual Studio Project</p> <p>Creating a layout      Creating components</p> <p>Dynamic binding      Rendering content</p>	<p><b>Module 4 Sitecore API</b></p> <p>Basic API concepts &amp; retrieving items</p> <p>Item links</p> <p>Creating, deleting, and modifying items</p> <p>Working with complex fields</p>
<p><b>Module 5 Advanced Presentation Concepts</b></p> <p>Reusable content</p> <p>Layout deltas</p>	<p><b>Module 6 Real World Solutions</b></p> <p>Familiar concepts</p> <p>Dealing with larger sites</p> <p>Sitecore query</p>
<p><b>Module 7 Configuring the Page Editor</b></p> <p>Datasource restrictions</p> <p>Parameters and parameter templates</p> <p>Placeholder settings</p> <p>Advanced Page Editor configuration</p>	<p><b>Module 8 Dealing With Your Data</b></p> <p>Item buckets</p> <p>Search</p>
<p><b>Module 9 Recommended Practices</b></p> <p>Working with media      Caching &amp; performance</p> <p>Publishing      Installing &amp; scaling</p> <p>Team development      Dealing with deployment</p> <p>Basic security      Workflow</p>	<p><b>Module 10 Marketing Functionality</b></p> <p>Introduction</p> <p>Engagement values &amp; goals</p> <p>Profiling &amp; personalization</p>
<p><b>Module 11 Optional Topics</b></p> <p>Branch templates</p> <p>Other item and template properties</p> <p>Pipelines and events</p> <p>Rules</p> <p>Placeholder overrides</p>	<p><b>Exam and Certification</b></p> <p>Theory exam with 40 questions</p>

## 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

**Sitecore combines content management with a marketing platform**

**Design and populate a website**

**Tailor the experience to individual visitors**

**What does Sitecore offer marketers?**

**Design and populate pages inline**

**Personalization and multivariate testing**

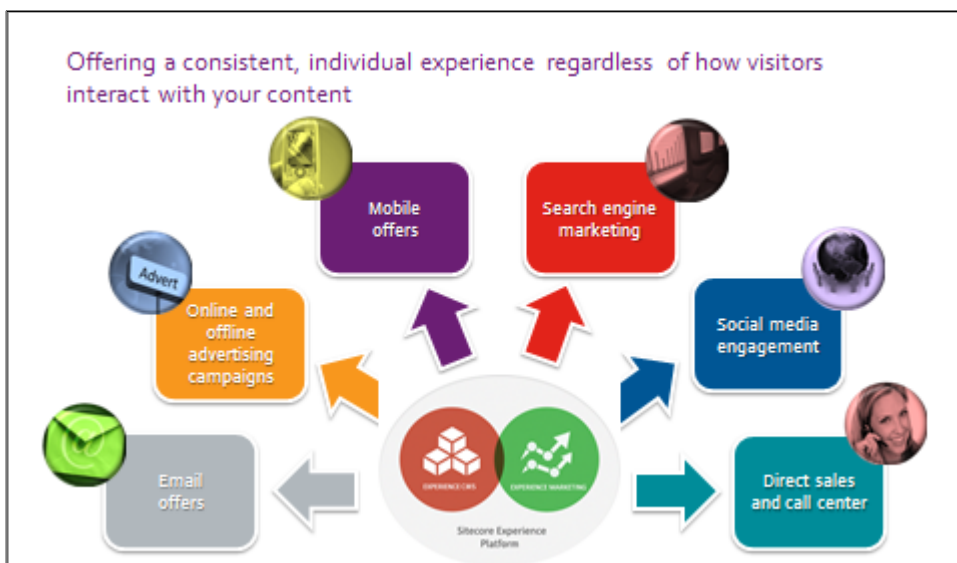
**Cross-channel communication**

**Integrate with external tools and databases**

**Rich reporting on individual visitors**



#### Cross-Channel Communication



### A Typical Scenario

3 Land on site, callback form personalized based on IP and how you found the site

2 Follow campaign link

1 Google "Cycling Holidays"

4 Sign up for callback

5 Receive e-mail – "Visit Facebook while you wait!"

6 Like Facebook page, click link to site – personalized 10% off offer

7 Sales rep can see that you triggered that offer.

### The Developer's Role

**Who are you?**  
Professional ASP.NET developers  
Passionate about supporting business goals

**What are your responsibilities?**  
Build and scale according to recommended practices  
Support Sitecore's marketing and analytics features  
Support the Page Editor  
"Don't fight the framework!"

**This course will get you there!**

### 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.


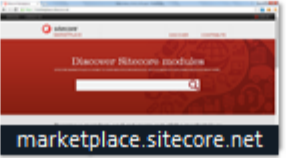
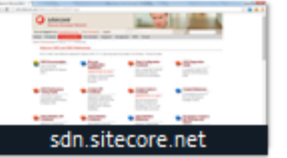
**MVP program**  
 Technical previews, MVP summit, closed product feedback circuit

**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?

<p><b>Knowledgebase</b> <i>(support articles)</i></p>  <p>kb.sitecore.net</p>	<p><b>Marketplace</b> <i>(modules)</i></p>  <p>marketplace.sitecore.net</p>	<p><b>Developer Network</b> <i>(docs, forum, downloads)</i></p>  <p>sdn.sitecore.net</p>
<p><b>Useful Guides</b></p> <ul style="list-style-type: none"> <li>• Scaling Guide</li> <li>• CMS Diagnostics Guide</li> <li>• CMS Performance Tuning Guide</li> <li>• Page Editor Recommended Practices for Developers</li> </ul>	<p><b>Documentation</b></p> <ul style="list-style-type: none"> <li>• <a href="https://doc.sitecore.net/">https://doc.sitecore.net/</a></li> <li>• <a href="https://sitecore-community.github.io/docs/">https://sitecore-community.github.io/docs/</a></li> </ul> <p><b>Modules</b></p> <ul style="list-style-type: none"> <li>• Email Experience Manager</li> <li>• Web Forms for Marketers</li> <li>• Social Connected</li> </ul>	
<p>All modules: <a href="http://sdn.sitecore.net/Resources/Downloads%20Overview.aspx">http://sdn.sitecore.net/Resources/Downloads%20Overview.aspx</a></p>		

## How Do I Contact Support?

<a href="http://support.sitecore.net/">http://support.sitecore.net/</a> <i>(with SDN login details)</i>	Try reproducing the issue on a clean installation before contacting support
One problem per issue	Look at log files using log analyzer
Timestamp your responses	Search marketplace before asking for code examples
Attach relevant configuration files	Compare ShowConfig.aspx to stock config
Attach recent log files, screenshots of UI errors, and files relating to issue	
Attach exceptions as plain text documents	
Public FTP (10MB+ files): <a href="http://sdn.sitecore.net/Support/Sitecore%20Public%20FTP.aspx">http://sdn.sitecore.net/Support/Sitecore%20Public%20FTP.aspx</a>	
HelpDesk Best Practices: <a href="http://sdn.sitecore.net/Support/Helpdesk%20Best%20Practices.aspx">http://sdn.sitecore.net/Support/Helpdesk%20Best%20Practices.aspx</a>	

## 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

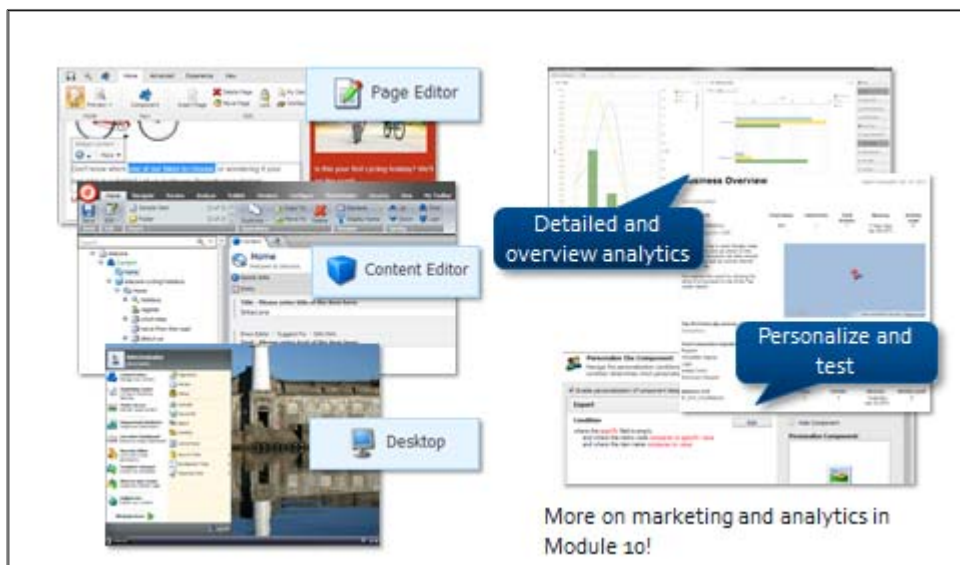
#### Objectives

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*.

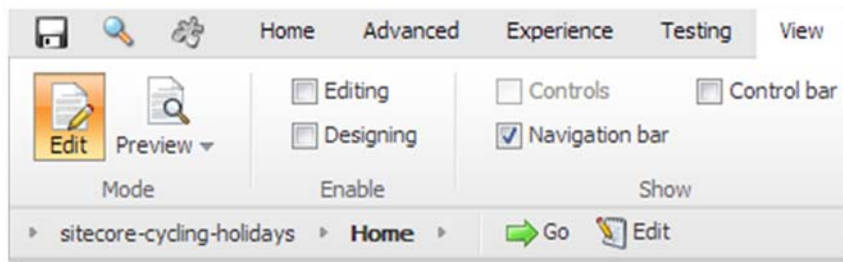
3. Double-click the **Page Editor** icon to log into that interface.
4. **Expand** the ribbon using the arrow in the top right-hand corner.

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.

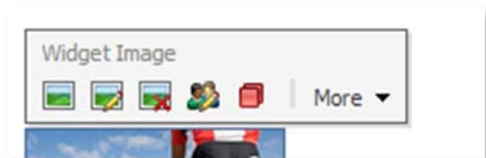


### Demo – Editing in Page Editor

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.



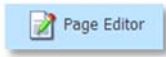
### Note

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

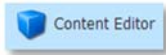
7. Change the **Heading** field to *Choices of bike* and then, on the ribbon, click **Publish > Preview** to see the changes.

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

Everything in Sitecore is an item

An item is a unit of content not a page

Items are **NOT** files

Items are represented in a hierarchical structure called the **Content tree**

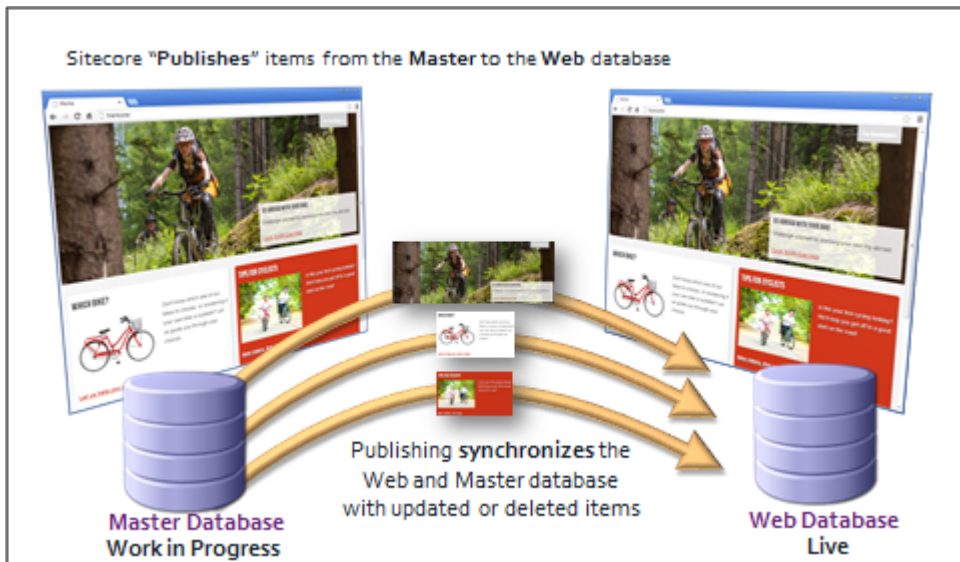
Some items are addressable via a **URL**

URL maps to **position in tree**

Sitecore interfaces display all or part of the Content tree

```
graph TD; Content[Content] --> Home1[Home]; Content --> sitecore[sitecore-cycling-holidays]; Home1 --> Home2[Home]; Home2 --> holidays[holidays]; holidays --> battle[battle-of-the-hills]; holidays --> cotswolds[cotswolds-adventure]; holidays --> cycle-cal[cycle-california]; holidays --> cycle-south[cycle-south-east-asia]; holidays --> discover-cop[discover-copenhagen]; holidays --> discover-thai[discover-thailand]; holidays --> mountain[mountain-biking-for-beginners]; holidays --> register[register]; holidays --> which-bike[which-bike];
```

## Publishing and the Sitecore Databases



## Topic 1.4 Sitecore from a Developer's Perspective

### Introduction

#### Objectives

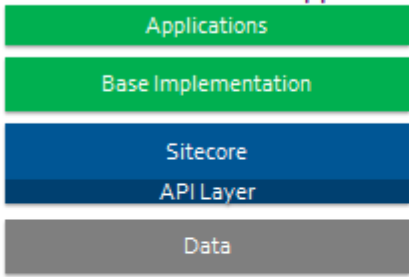
By the end of this topic you will be able to:

- 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


**Sitecore is an ASP.NET application**



**Sitecore is a framework:**  
Content versioning, media library, design/editing interfaces, language versioning, search API, adaptive design via devices, rich API, hierarchical data storage

**Current Requirements**

- ASP.NET 4.5
- SQL 2008 R2 SP1
- IIS Integrated Mode
- MVC 5.1 (for SPEAK UI)



**Sitecore CMS Compatibility Table**  
<https://kb.sitecore.net/articles/087164>


**MVC Compatibility Table**  
<https://kb.sitecore.net/articles/522918>

**Modules Compatibility Table**  
<https://kb.sitecore.net/articles/541788>

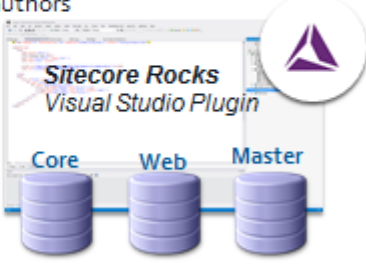
#### Separation of Concerns

**Authors and developers have separate responsibilities**

**Authors...**  
Manage **content** and **presentation** using your custom implementation of Sitecore

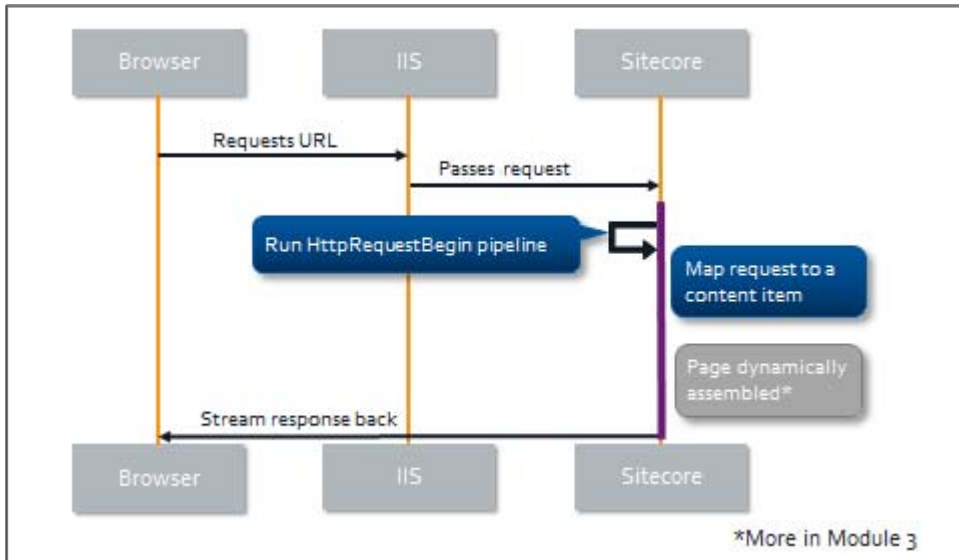


**Developers...**  
Define **data structure** and **presentation components** for authors



**Pages are dynamically assembled on demand**

### How Sitecore Handles a Request



### What Gets Installed?

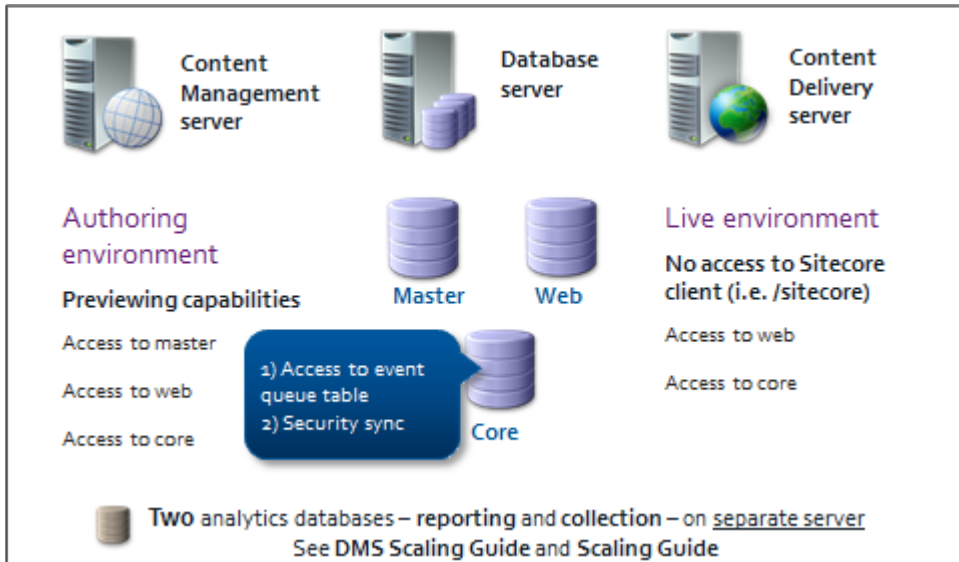
**Three main databases + analytics**

- Master** – authoring database, work in progress
- Web** – published, live content
- Core** – settings and membership
- Analytics** - installed as separate package!

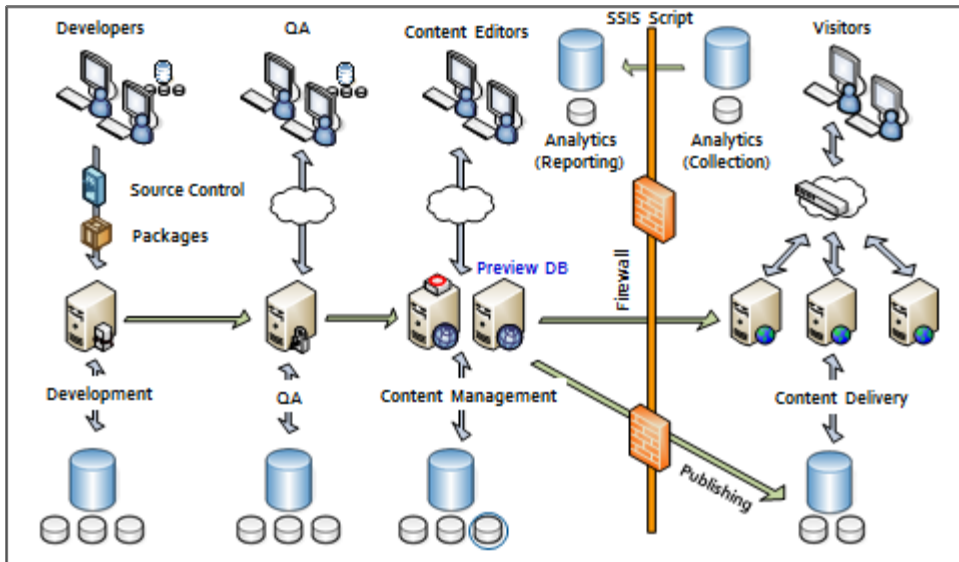
**File system**

- Website** – Base Sitecore application and your work
- Database** – database files / logs
- Data** – license, logs, packages, indexes

### 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

**An item**

- Is a unit of content made up of fields
- Has fields organized into field sections

**Data templates**

- Define a type of item
- Determine what field sections, field types, and field names an item will have
- Field types determine the editor control e.g. Image field, Rich-Text field



**Field Section**

**Various fields**

**Data Template defining field sections and fields**

Basic Information	
Heading	Single-Line Text
Main Content	Rich Text
Main Image	Image

#### Field Type Determines...

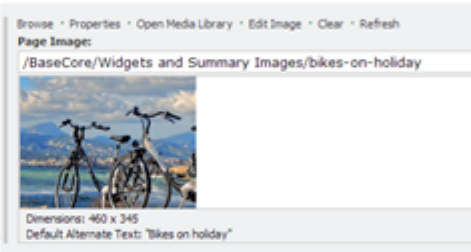
**Editing control** in the Content Editor

*The value stored in the database (raw value)\**

*Sitecore API Field class\**

*Sitecore web control\**

What can go into the source field



Page Content	
Page Heading	Single-Line Text
Page Summary	Rich Text
Add a new field	Single-Line Text

(\*more about field types in Module 4!)



### Field Source

Ability to restrict an authors browsing location for certain field types

Selection list root item GUID

Selection item GUID (rich text fields)

Query or source Parameters

**Note:** An inherited field source cannot be overridden (more about field types in Module 4!)

### Who Decides What the Data Structure Will Be?

Inferred from designs, wireframes, and static HTML

Start project by designing your data structure (like SQL schema)

What data templates and fields can you infer from the designs?

Date	20th April 2014
Price per person	£2000 per person

## Are Fields Shared?

Consider a site with the follow three data templates

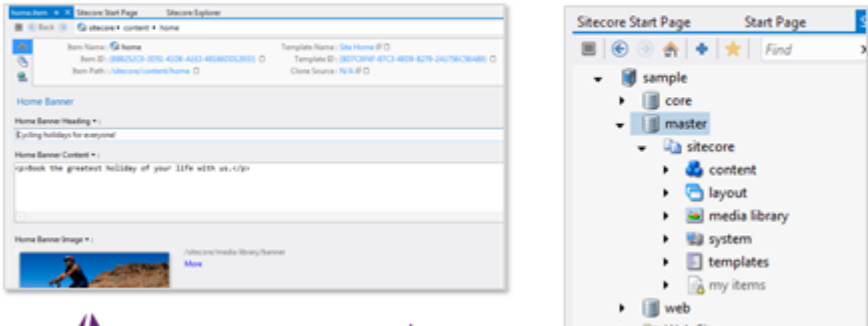
Three of these fields are shared. Create a base template!

<p><b>Holidays Section</b></p> <ul style="list-style-type: none"> <li>Heading</li> <li>Main Content</li> <li>Main Image</li> </ul>	<p><b>Trip Details</b></p> <ul style="list-style-type: none"> <li>Heading</li> <li>Main Content</li> <li>Main Image</li> <li>Price per person</li> <li>Start date</li> </ul>	<p><b>Bicycle Details</b></p> <ul style="list-style-type: none"> <li>Heading</li> <li>Main Content</li> <li>Main Image</li> <li>Type</li> <li>Suitability</li> </ul>
--	--	--


## We Use Sitecore Rocks

A free plug-in adding the Content Tree to the Visual Studio IDE

This course uses Sitecore Rocks extensively.



The image shows two screenshots. The left one is a screenshot of Sitecore Explorer displaying a content tree with items like 'Home Banner' and 'Home Banner Content'. The right one is a screenshot of the Sitecore Start Page in Visual Studio, showing the Sitecore Rocks content tree on the left side of the IDE.




### 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*.

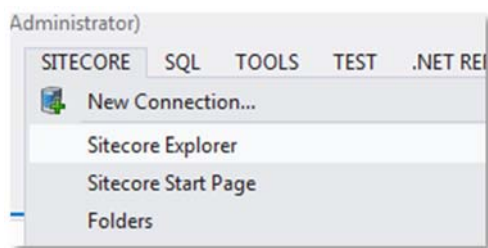
Create **field sections** and **fields** as described in the following table:

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

### Detailed Steps

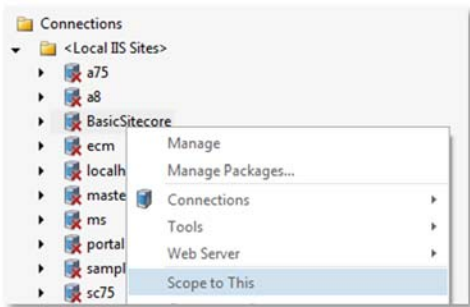
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

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

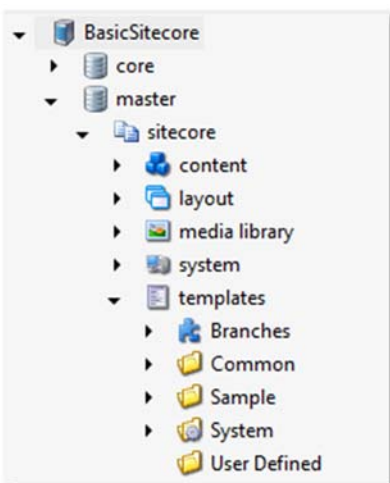


3. Locate your instance in the Sitecore Explorer window under **Connections > <Local IIS Sites>**. The instance name is *BasicSitecore* unless your instructor tells you otherwise.

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



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



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



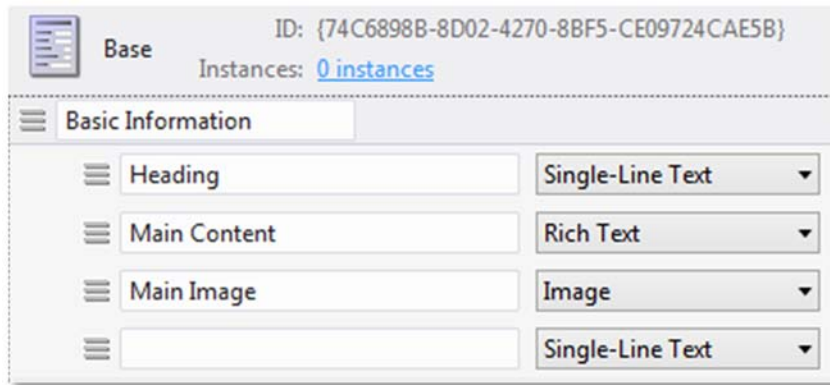
### Keyboard Shortcut **CTRL + Shift + Space = Commandy**

*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.*

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

Field section	
<b>Basic Information</b>	
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

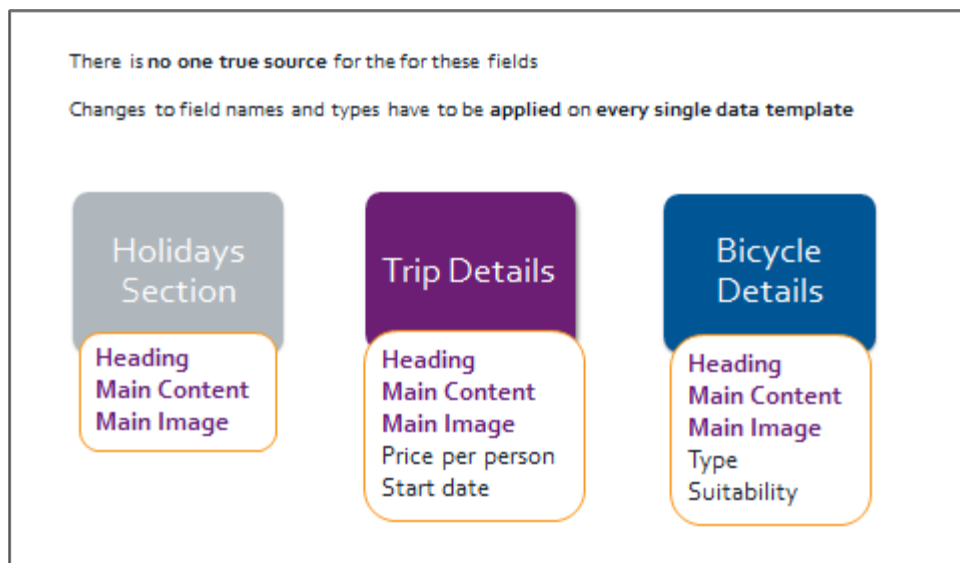
#### Objectives

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

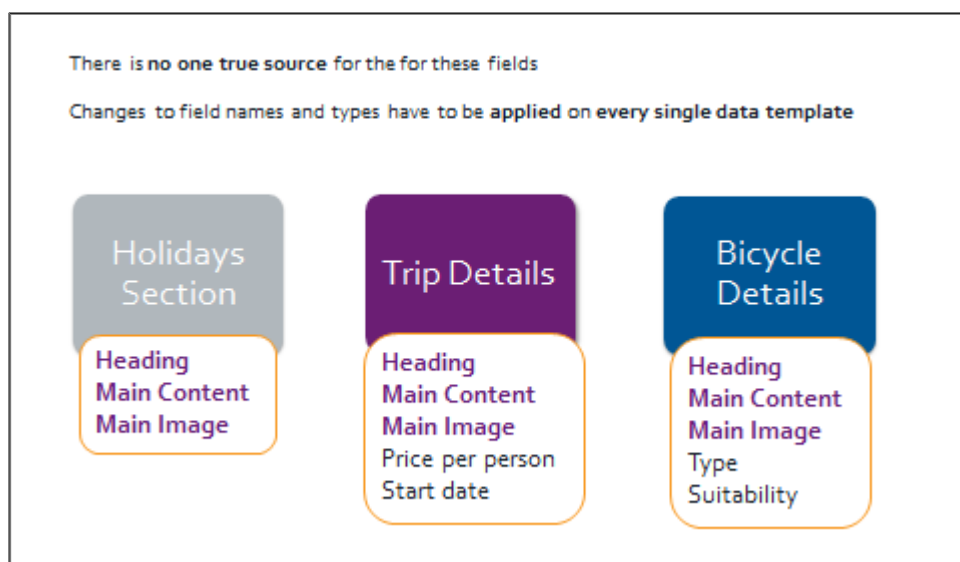
- Specify the base templates a data template inherits from.
- Describe the benefits of data template inheritance.
- Name the data template that all data templates inherit from.

### 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 ( $A \rightarrow B \rightarrow A$ ).

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**

Every **instance of an item** using these data templates gets all these fields

Get data template inheritance right! Refactoring later will cause you to lose field values.

<div style="background-color: #f9a825; border-radius: 10px; padding: 10px; margin-bottom: 10px;"> <p>Holidays Section</p> <p><b>Heading</b></p> <p><b>Main Content</b></p> <p><b>Main Image</b></p> </div>	<div style="background-color: #6a3d9a; border-radius: 10px; padding: 10px; margin-bottom: 10px;"> <p>Trip Details</p> <p><b>Heading</b></p> <p><b>Main Content</b></p> <p><b>Main Image</b></p> <p>Price per person</p> <p>Start date</p> </div>	<div style="background-color: #e31a1c; border-radius: 10px; padding: 10px; margin-bottom: 10px;"> <p>Bicycle Details</p> <p><b>Heading</b></p> <p><b>Main Content</b></p> <p><b>Main Image</b></p> <p>Type</p> <p>Suitability</p> </div>
--	--	--

**Watch out** for circular data template inheritance – a data template should **never** inherit directly or indirectly from itself

**Note:** Field names must be **unique**, but field sections with the same name will **merge**

**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

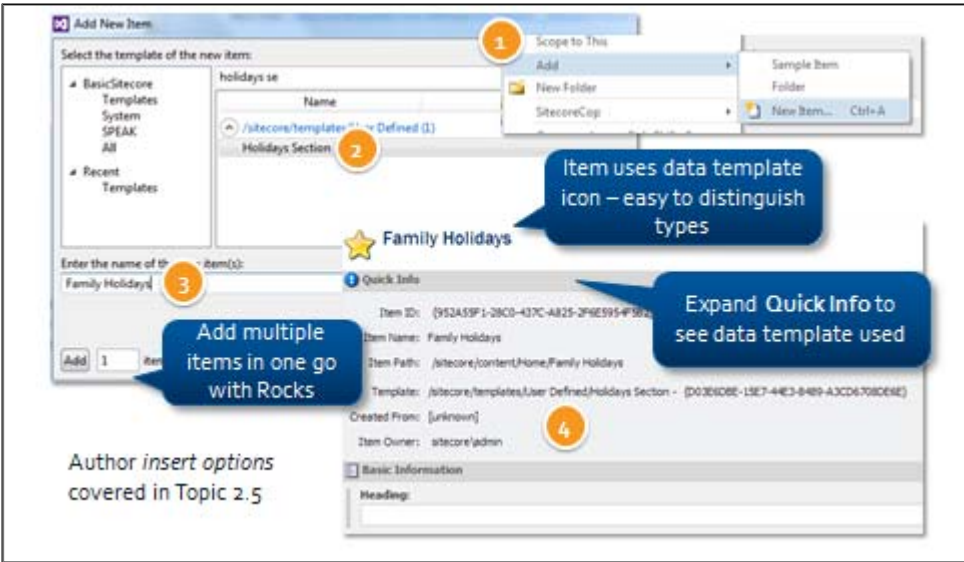
Encapsulate all fields required for a specific feature in a data template, which other templates can then inherit from. For example:

- 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



The screenshot shows the 'Add New Item' dialog in Sitecore Rocks. It features a tree view on the left with 'Basic\Sitecore Templates\System\SPEAK\All' expanded. The main area shows a table of templates, with 'Holidays Section' selected. A context menu is open over the 'Holidays Section' row, showing options like 'Add', 'New Folder', and 'New Item...'. Annotations include:
 

- 1: Scope to This (pointing to the 'Add' menu item)
- 2: Item uses data template icon – easy to distinguish types (pointing to a star icon next to 'Family Holidays')
- 3: Add multiple items in one go with Rocks (pointing to the 'Add' button and the 'Family Holidays' text input)
- 4: Expand Quick Info to see data template used (pointing to the 'Quick Info' section which displays item details like ID, Name, Path, and Template).

 At the bottom left, text reads: 'Author insert options covered in Topic 2.5'.



## 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.

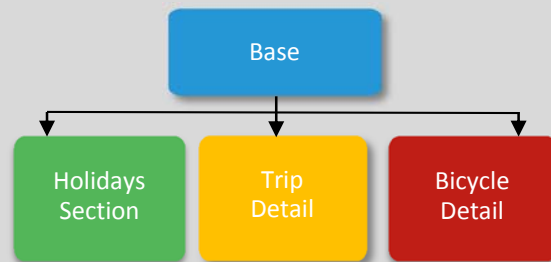


Figure 2-1 Data template inheritance structure

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

##### Overview Steps

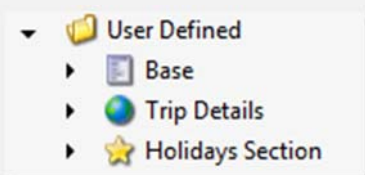
*Optional detailed steps are available below to help you.*

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:

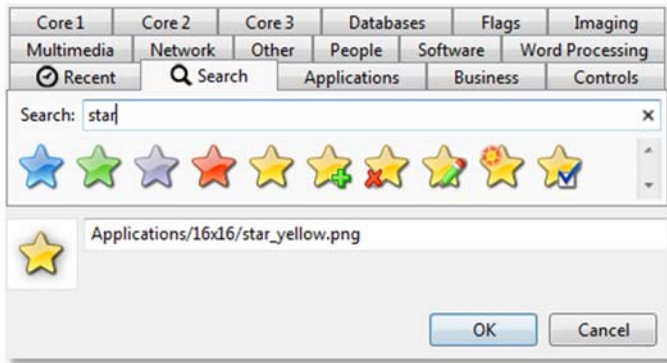


##### Detailed Steps

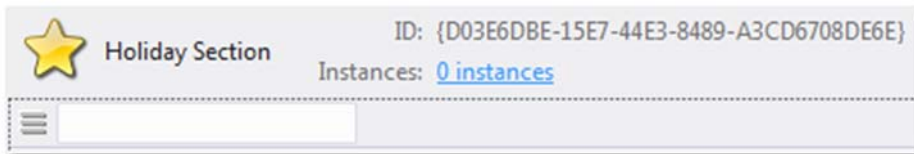
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

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**.



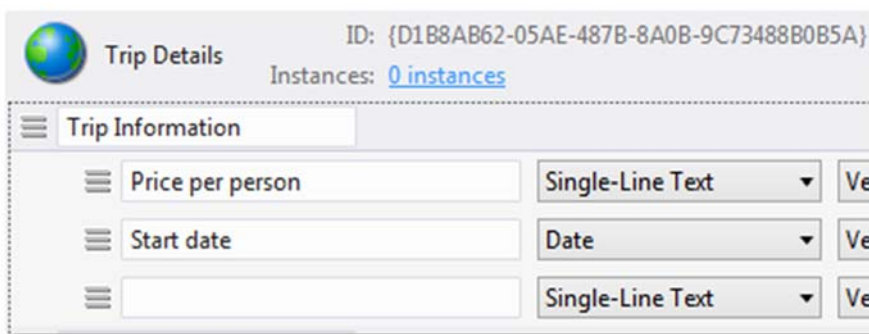
8. **Save**. Your data template will look like this:



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

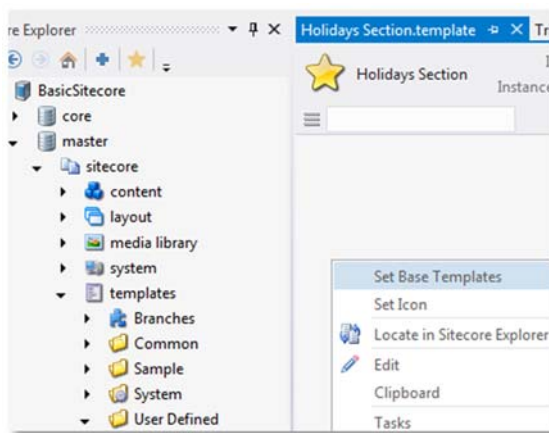
*Optional detailed steps are available below to help you.*

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**.

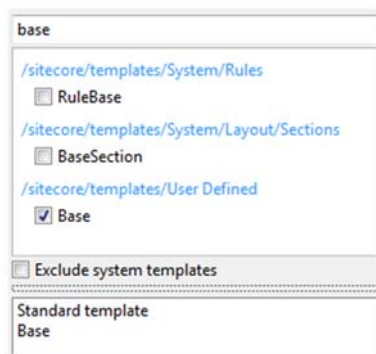
### Detailed Steps

*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.*

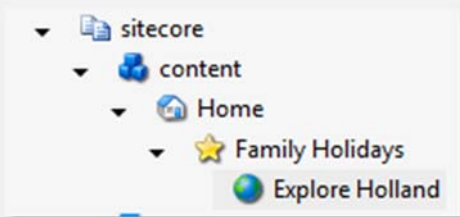
Using **Sitecore Rocks**, right-click on the **Home** item under: `/BasicSitecore/master/sitecore/Content` and create a new item based on the **Holidays Section** item. Name it *Family Holidays*. Populate it with the following sample content, or write your own:

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

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

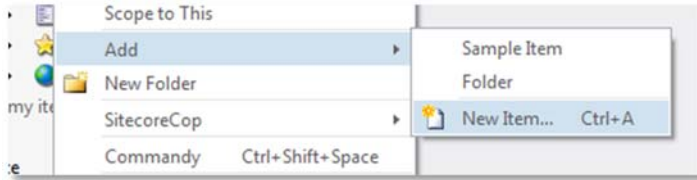
*You will end up with a content tree that looks like this:*



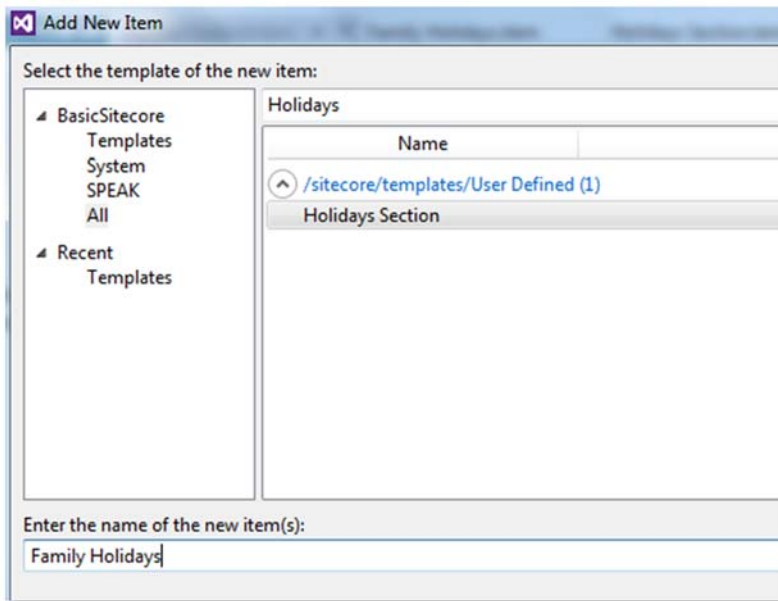
## Detailed Steps

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...**



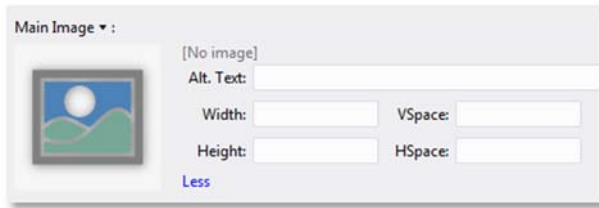
4. Search for **Holidays Section** in the **Add New Item** dialog. Make sure that **All** is selected in the list to the left:



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

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

- 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*.

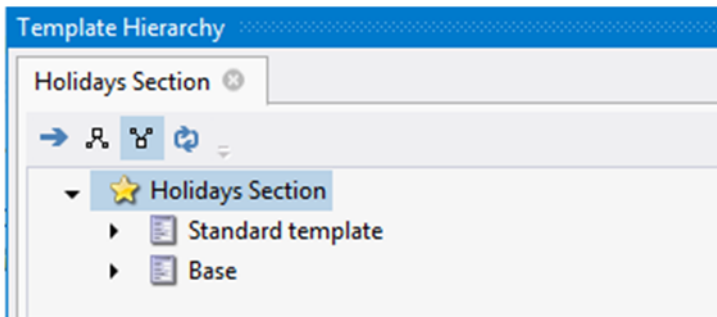


- Right-click the **Family Holidays** item and insert a new **child item** based on the **Trip Details** data template. Name this item *Explore Holland*.
- Populate the item with the following content, or write your own:

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

**i** 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 follow inheritance rules!

**Properties of standard values items**

- Manually created item
- Child item of the data template
- Made up of all the fields from all the inherited templates, including its own
- Always named `__StandardValues`

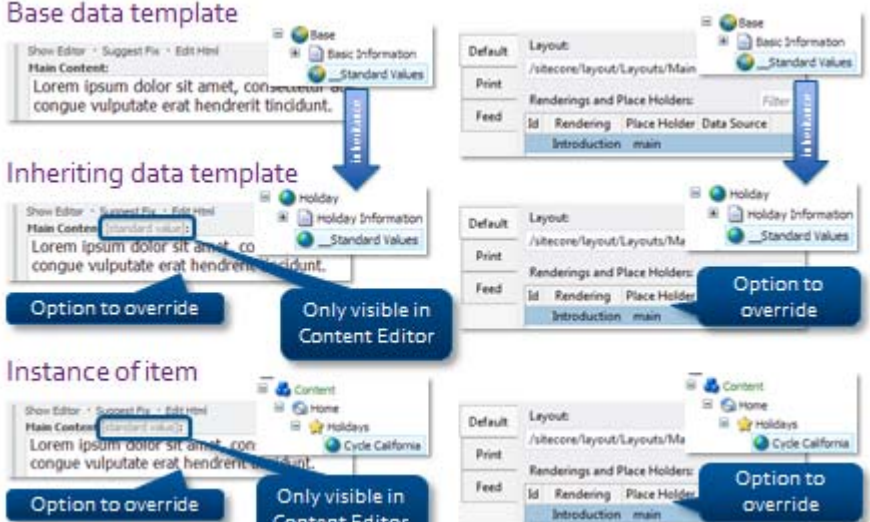
**Standard values allow you to...**

- Specify default field values
- Set initial field values using tokens

**We will also cover...**

- Assigning insert options
- Assigning default look and feel
- Assigning default workflow state

#### Standard Values and Inheritance



**Base data template**

Standard values are inherited from the base data template.

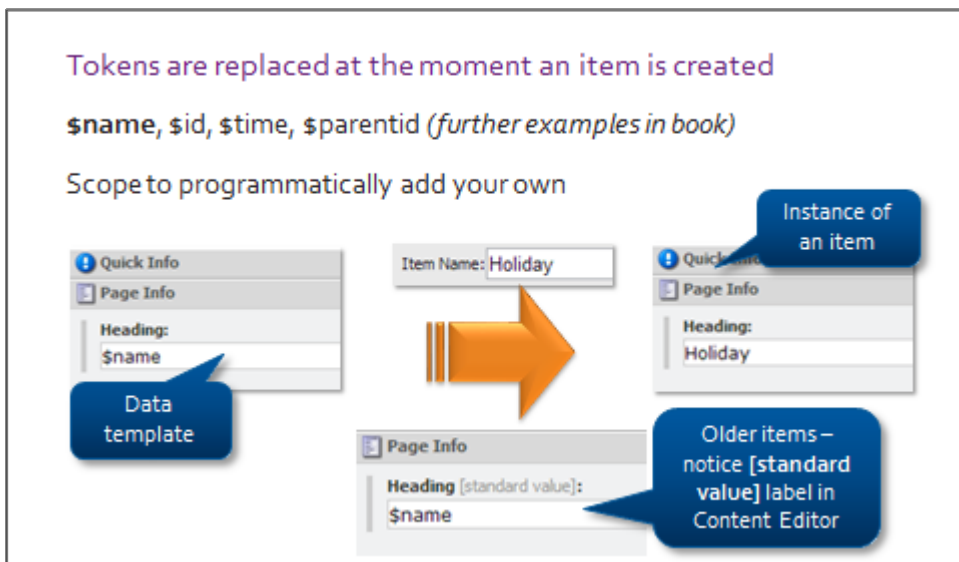
**Inheriting data template**

Standard values are inherited from the base data template. Option to override is available.

**Instance of item**

Standard values are inherited from the base data template. Option to override is available.

## Tokens



Token Name	Description
<b>\$name</b>	Name of the item
<b>\$time</b>	ID of the item
<b>\$now</b>	ID of the parent of the item
<b>\$date</b>	Name of the parent of the item
<b>\$time</b>	Current date (ISO format)
<b>\$now</b>	Current time (ISO format)
<b>\$date</b>	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

**Business names preferred to allow tokens to work**

Hyphenated item names result in hyphenated \$name token replacement

But spaces are encoded as %20 in URLs

**Options:**

- Create item, then rename
- Add entry to `<encodeNameReplacements>` in web.config
- Write a custom pipeline



### 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:

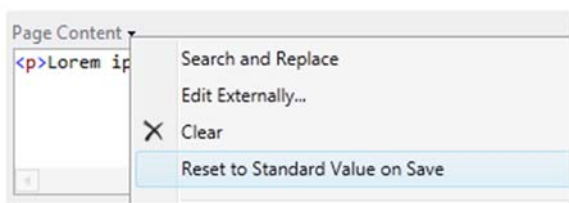
- Create standard values for the **Base** data template.
- Add a **\$name** token to the **Heading** field.
- Add **Lorem ipsum** content to the **Main Content** field.
- Create a new item based on the **Trip Details** data template.
- Change **Main Content** and reset to standard values.

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.*

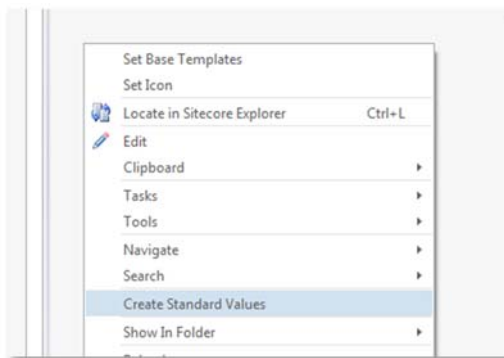
Using **Sitecore Rocks**, create the **Standard Values** for the **Base** data template.

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**.

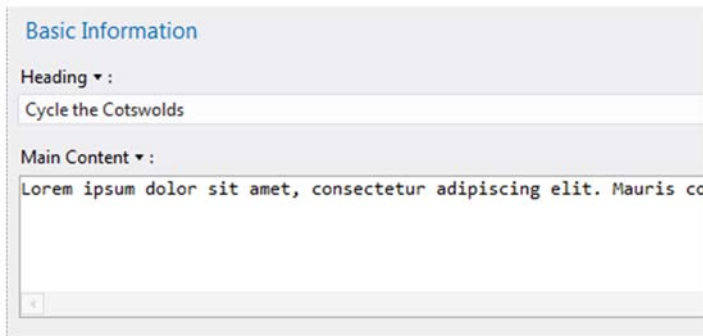
**Note** that the **Heading** field is replaced by the item's name, and the **Main Content** field contains sample content, "Lorem ipsum...". Populate the remaining fields with sample content for the following table, or write your own.

Discover the Cotswolds	
<b>Basic Information</b>	
Main Image	<i>Any from the media library</i>
<b>Trip Information</b>	
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	
<b>Basic Information</b>	
Main Image	<i>Any from the media library</i>
<b>Trip Information</b>	
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

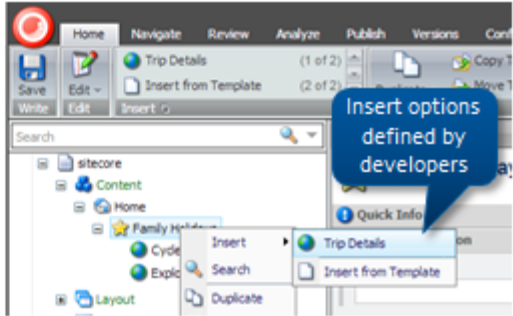
#### What Are Insert Options?

Only **developers and administrators** can create items of any type, anywhere in the content tree (*Rocks does not restrict at all!*)

**Authors** require a list of **allowed data templates**

**Insert options...**

- Define allowed **child items**
- Allow authors to build content tree to the **nth level**
- Reduce risk of **human error**
- Can be **overridden** at item level
- Should be defined on **standard values**



The screenshot shows the Sitecore Content Editor interface. The 'Insert' menu is open, showing options like 'Trip Details', 'Search', and 'Insert from Template'. A blue callout bubble with the text 'Insert options defined by developers' points to the 'Insert' menu. The content tree on the left shows a hierarchy: 'sitecore' > 'Content' > 'Home' > 'Family Holidays' > 'Cycle' > 'Explo' > 'Layout'.

#### 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

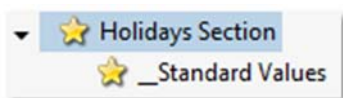
*Optional detailed steps are available below to help you.*

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

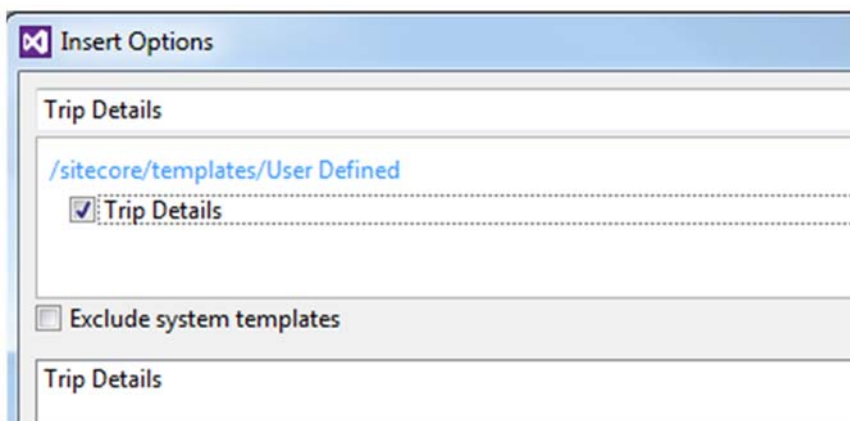
#### Detailed Steps

*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.

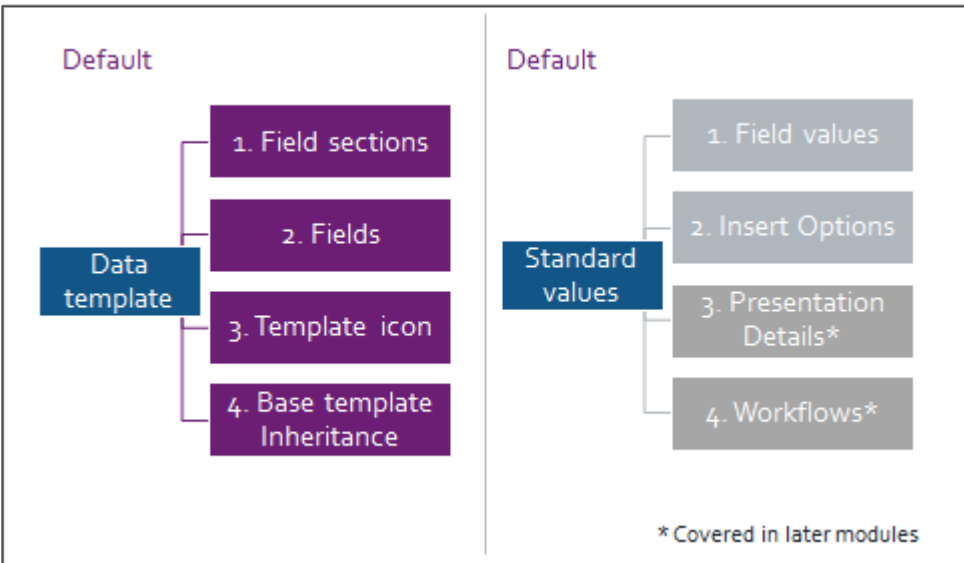
**Field**  
 Holds content. Can be different types – single-line text, rich text, image, link. Type determines the interface that the author sees.

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

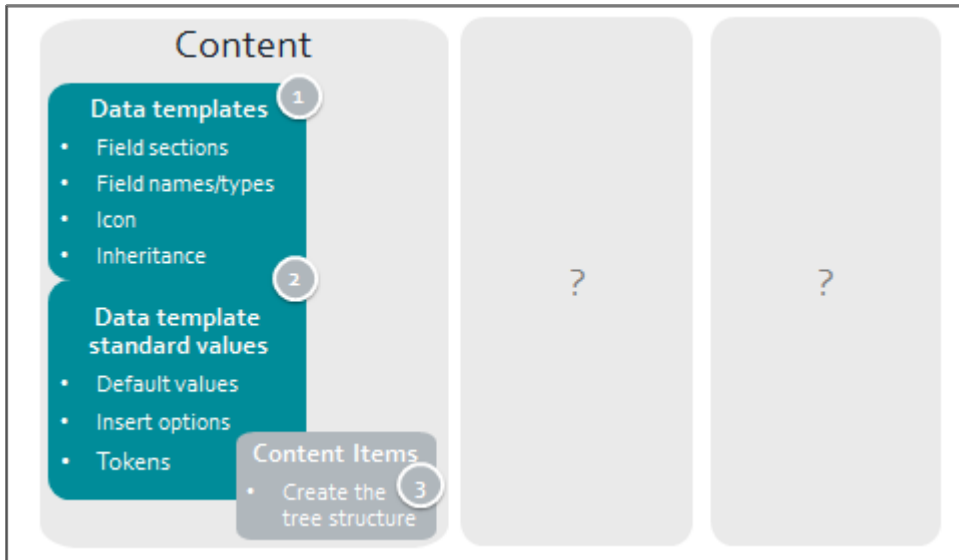
**Standard Values**  
 \_\_Standard Values child item of a data template. Vehicle for defaults.

**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

- Presentation Is Dynamic and Modular
- Preparing to Build
- Creating a Layout
- Creating Components
- Dynamic Binding
- Rendering Content
- Summary and Optional Lab



## Topic 3.1 Presentation Is Dynamic and Modular

### Introduction

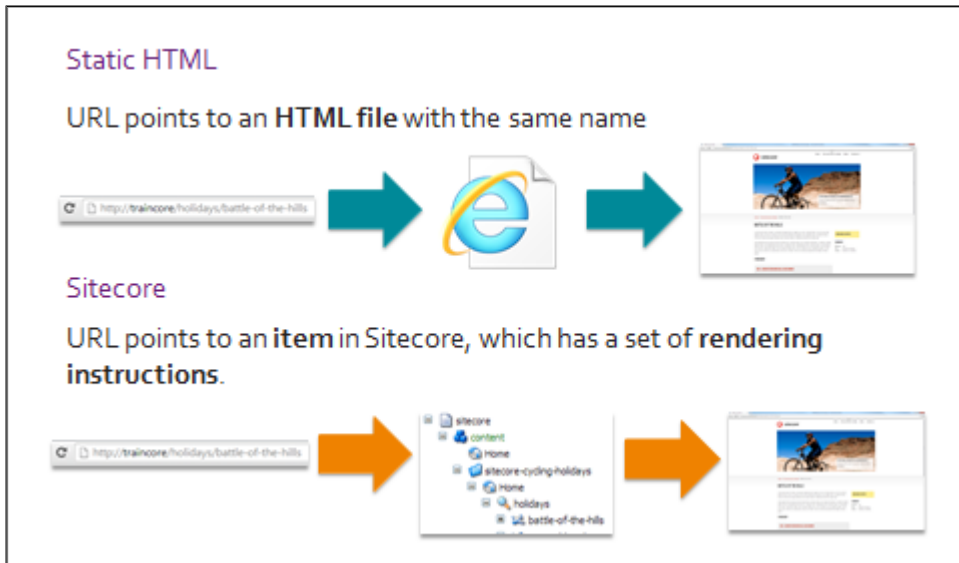
#### Objectives

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

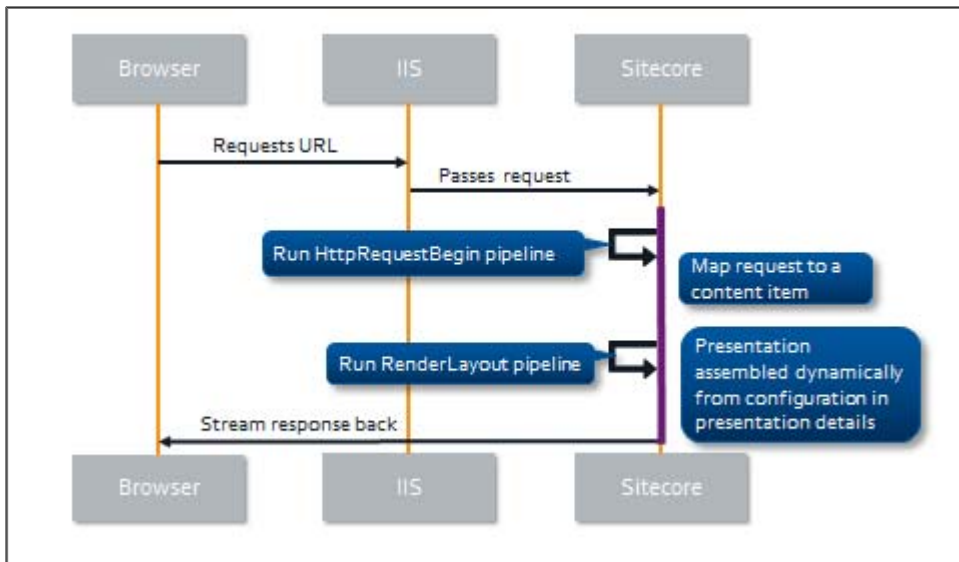
- Explain how Sitecore resolves a page compared to a static site.
- Define presentation details in general terms.

### Content

#### How Sitecore Resolves a Page



#### A Detailed Look at a Request



## Presentation Details

Presentation details are configurations that determine the appearance of an item when it is requested by the browser.

Modified behind the scenes by the Page Editor

Modular components on a canvas

Presentation details map to physical files on the file system - like .aspx and .ascx



### 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:

Id	Rendering	Place Holder	Data Source
Sample Sublayout		main	
Sample Inner Sublayout		/main/centercolumn	
Sample Rendering		content	

## 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

**Break designs into units of reusable functionality**

Consult your front-end developers, designers, and user experience designers!



#### 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?

Choose a site that a member of the group has worked on, or ask the instructor to suggest a known Sitecore site.

Look at several different page types. How would you break these pages up into **components**?

Is there any **nesting** of components – such as content within two or three-column grids?

This guide will help you with this process

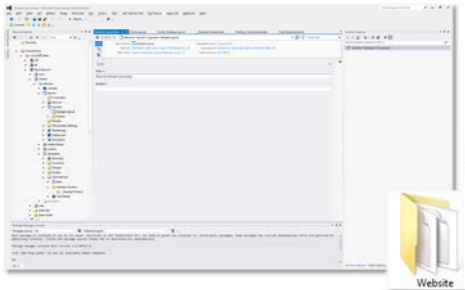
## Create a Visual Studio Project

### Presentation details map to physical files on the file system

Sitecore is an **ASP.NET application** – it has a web root and web.config

**Create and manage** presentation files in a normal Visual Studio project

Connect **Sitecore Rocks** to project – you never have to leave Visual Studio!



### 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**:

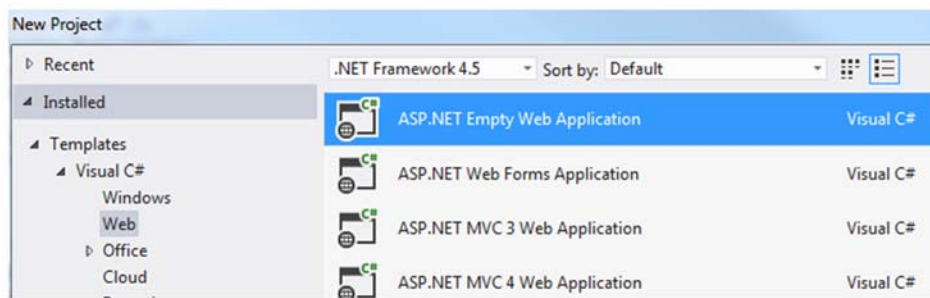
- .sln
- .csproj
- Properties folder
- obj folder

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

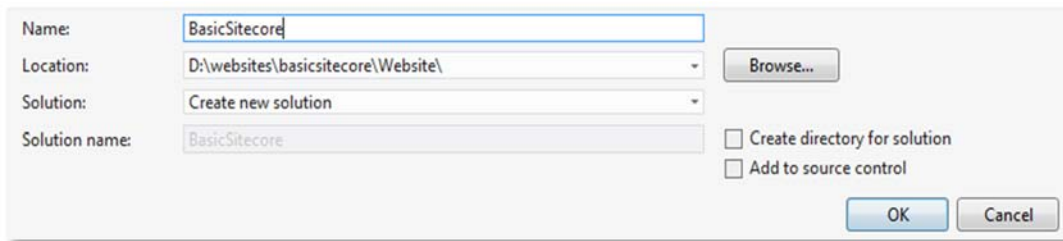
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

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:

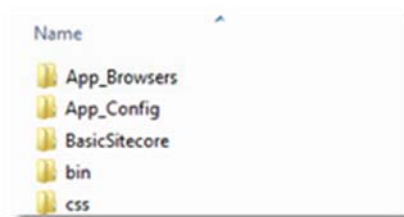


4. Name the project *BasicSitecore*, and change the **Location** to the **web root of your Sitecore instance**. This differs depending on the computer you are using, but is likely to be:  
`C:\inetpub\wwwroot\BasicSitecore\Website`.  
The web root specifically refers to the **Website** folder, where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

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

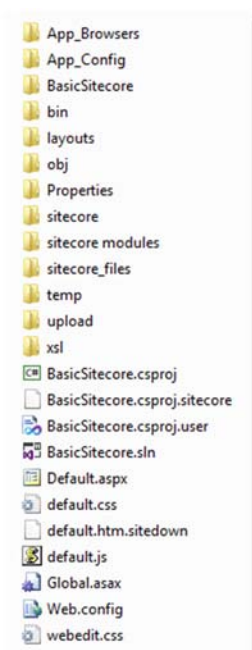


- Click **OK**.
- The solution will appear in a **subfolder** in the web root. Some of the contents of this subfolder need to be moved into the web root. **Close** the solution navigating to **File>Close Solution** on the **Visual Studio** toolbar.
- In the **file system**, navigate to your Sitecore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

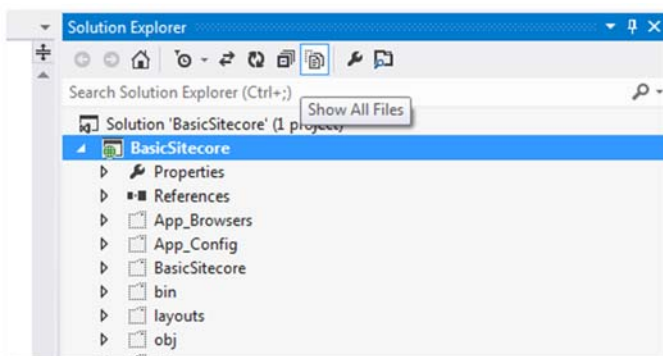


- Open this folder. Copy the following files and folders into the **web root** (one folder above). Your files should end up on the same level as Sitecore's own web.config:
  - BasicSitecore.sln
  - BasicSitecore.csproj
  - Properties
  - obj

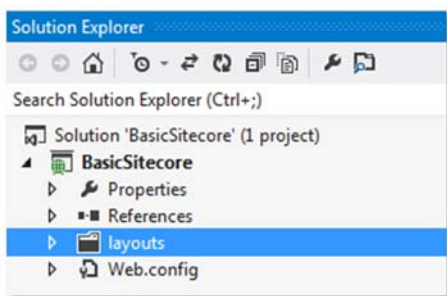
**Do not** overwrite Sitecore's own web.config. Your project folders and files are now mixed in with the default Sitecore folders and files in the web root:



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.*

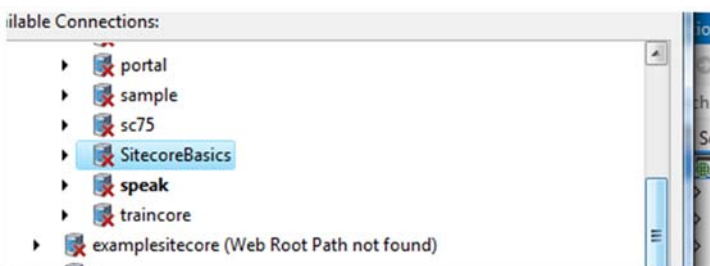
Right-click the **BasicSitecore** project and select **Sitecore>Connect to Sitecore...**

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**

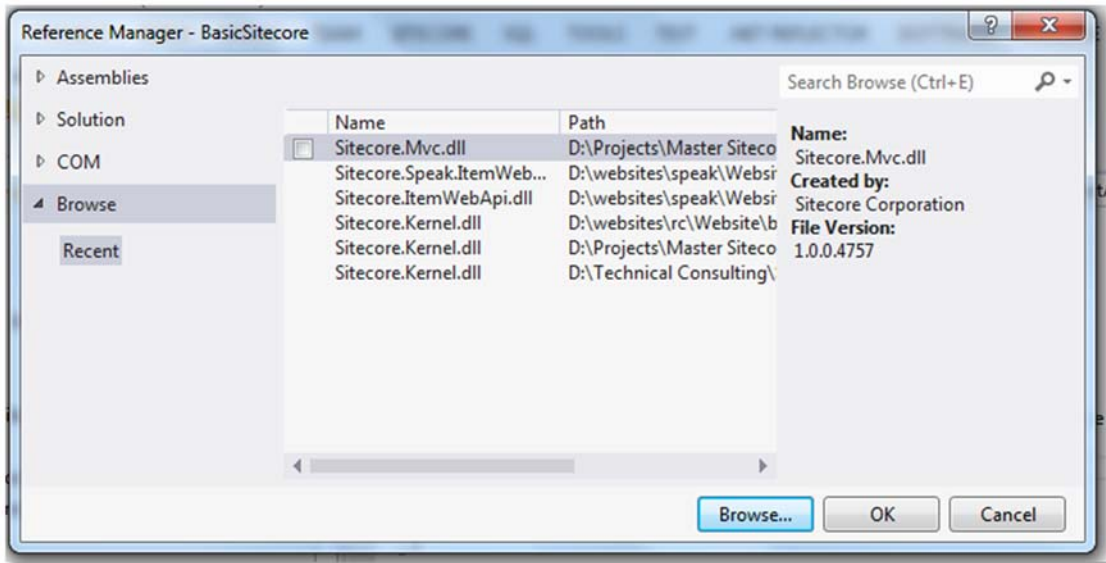
*Optional detailed steps are available below to help you.*

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

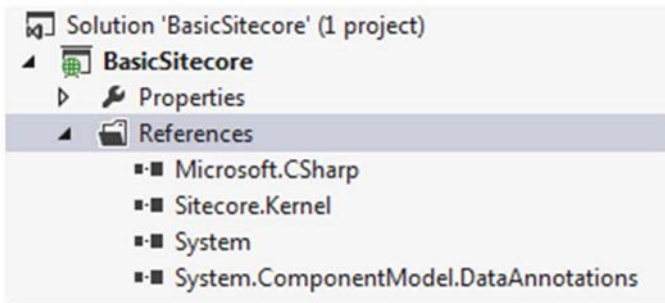
**Detailed steps**

*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

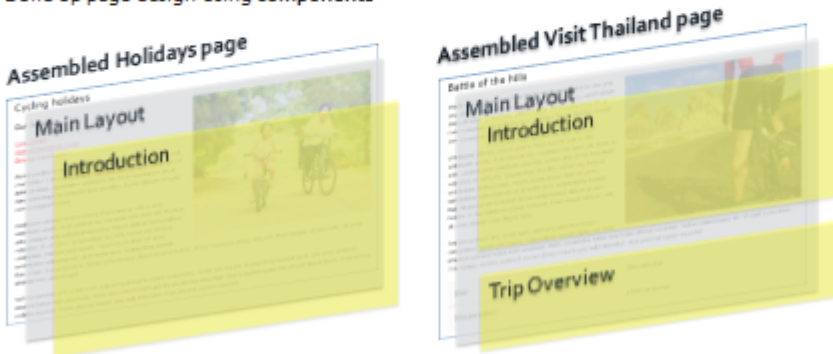
#### One Layout for All Items Within a Site

A layout is a canvas or the 'scaffolding' for a site

Define a single shared layout for all items within a site (*per device!\**)

Set layout on standard values (consider setting layout on Base)

Build up page design using components



#### What Is a Layout?

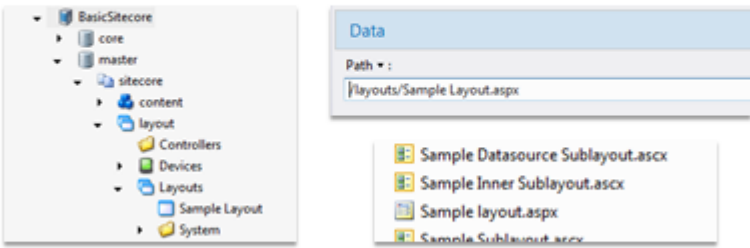
A layout is...

An .aspx file on the file system

A corresponding layout definition item in the tree

Linked by the Path field

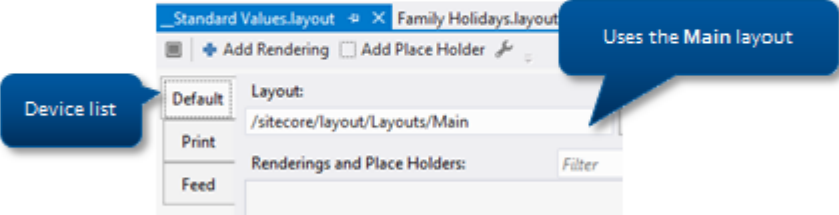
See lab instructions for details on how to create a layout in Sitecore Rocks!



**Note:** The item and the file can be referred to as 'a layout'

## Assigning a Layout

**Items must have a layout**  
 Part of an item's presentation details  
 Recommended practice – assign to data template's standard values  
 An item can have one layout per device\*(mobile, print, RSS)



\* More on devices in Module 6



### 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.*

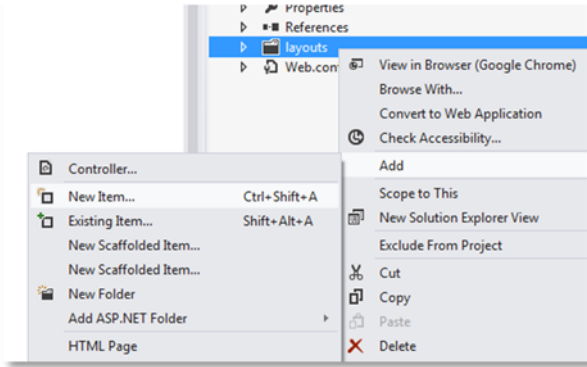
Using **Visual Studio's Solution Explorer**, create a new **Sitecore layout** in the **layouts** folder. When you are prompted by Sitecore Rocks to choose a location for the corresponding item, navigate to: `/BasicSitecore/master/sitecore/Layout/Layouts`.

##### Detailed Steps

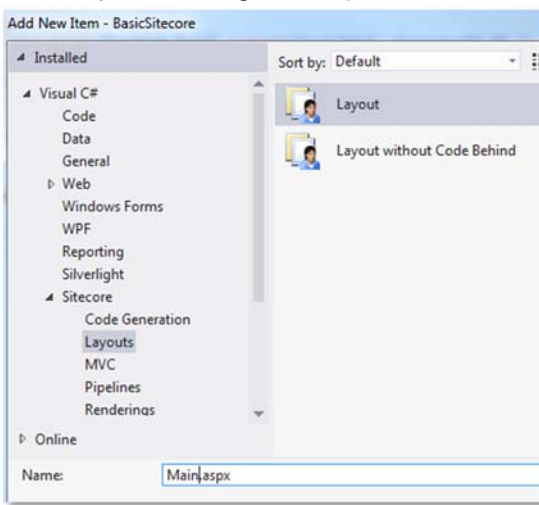
*If you completed the lab using the Overview Steps above, you **do not** need to follow these steps.*

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

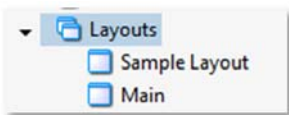
- Right-click the **Layouts** folder and select **Add>New item...**



- 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.
- Select **Layout** in the right-hand panel, and call it **Main.aspx**:



- Click **Add**, and wait for Sitecore Rocks to prompt you to create an item.
- When prompted by Sitecore Rocks to **create a layout item**, navigate to: */BasicSitecore/master/sitecore/Layouts* and select it.
- Click **OK**.
- Using Sitecore Explorer, right-click and **refresh** the */BasicSitecore/master/sitecore/Layouts* item.
- 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 -->

<!-- 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" %>
```

- 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

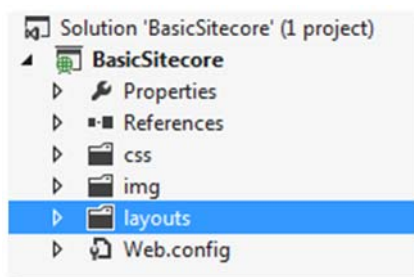
```
<!DOCTYPE html>
<!--[if (gte IE 9)]!(IE)]><!--><html> <!--<![endif]-->
<head>
<meta charset="utf-8" />
<title>Campaign Page</title>
<meta name="description" content="Test" />
<meta name="author" content="" />

<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 -->

<!-- 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>
```

- Save the file.
- On the **file system**, browse to **Student Resources Folder > WND Labs > Module 3 > Topic 3.3**.
- 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:



- Build the solution.

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

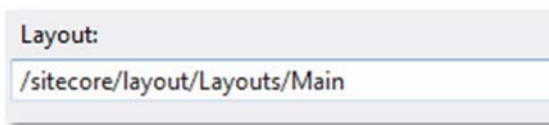
*Optional detailed steps are available below to help you.*

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.

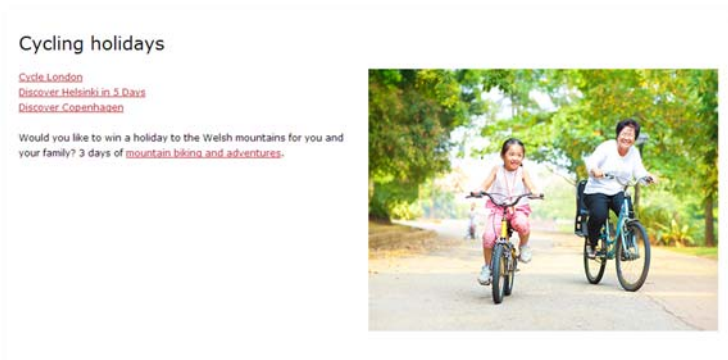
**Detailed Steps**

*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

Pages are assembled from smaller units of functionality

Various types – collectively known as **components**

Components can be **re-used and moved**, and they can be **nested**

Arranged to create a unique **page design**

#### What Is a Component?

**A component can be a...**

- Sublayout (Sitecore's wrapper for a user control)
- XSLT Rendering
- Web control

**We will always use the term *component*.**

There are Sitecore MVC alternatives – see videos!

**All components consist of...**

<b>Definition item in Sitecore:</b>	<b>File on the file system:</b>
/sitecore/Layout/Sublayouts	.ascx (sublayout)
/sitecore/Layout/Renderings	.xslt (XSLT rendering)

Linked by a Path field on definition item\*



## Tip – Sitecore MVC Videos

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

Part 2: [https://www.youtube.com/watch?v=dW\\_rQp9bMmE](https://www.youtube.com/watch?v=dW_rQp9bMmE)

## What Is a Sitecore Sublayout?

**A Sitecore sublayout is...**

Most commonly used component in a web forms implementation

An .aspx file on the file system (with normal .aspx.cs code behind)

A corresponding sublayout definition item in the Sitecore tree

Linked by the Path field

See lab instructions for details on how to create a sublayout in Sitecore Rocks!

Wrapper for normal user control



## 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.aspx.

### Lab A. Create a Sublayout Called *Introduction*

#### Overview Steps

*Optional detailed steps are available below to help you.*

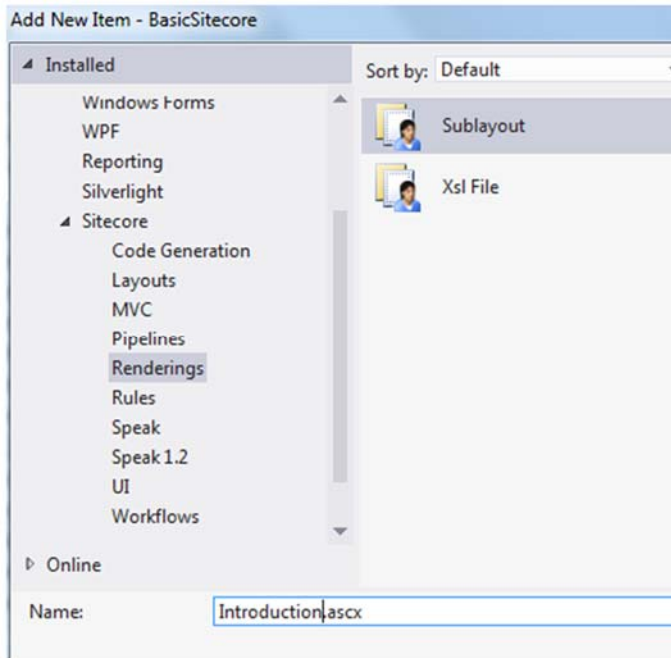
Using **Sitecore Rocks**, create a new sublayout called *Introduction*. Ensure that the corresponding definition item is created under: `/BasicSitecore/master/sitecore/Layout/Sublayouts`.



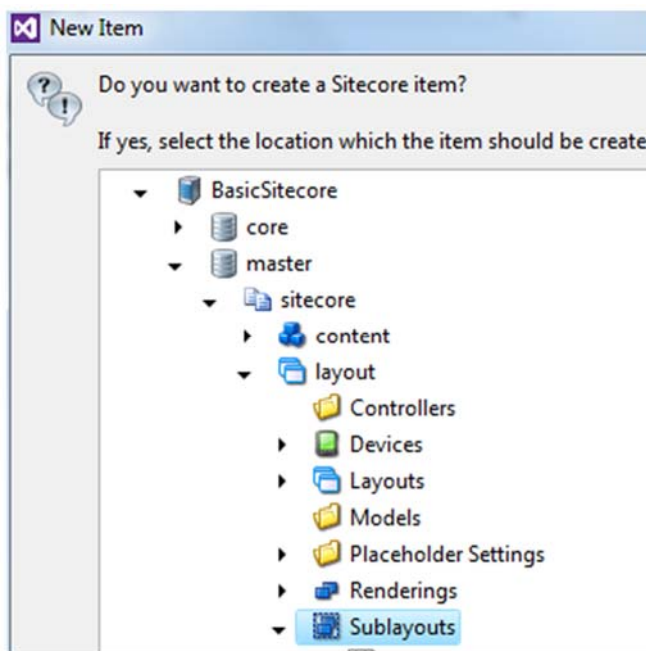
## Detailed Steps

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 -->

<!-- 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 -->

<!-- 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?

Declare **placeholders** within markup

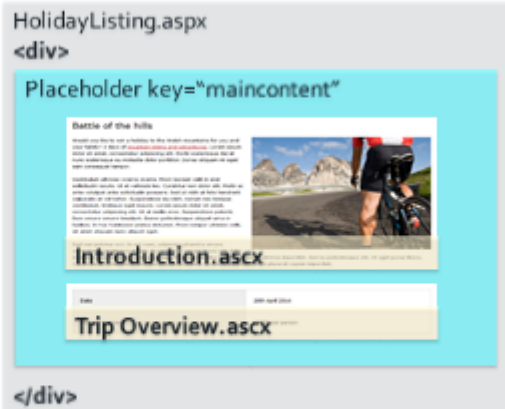
Give them a **unique key**

Bind **any number** of components to a placeholder in an item's presentation details.

Page Editor does this behind the scenes!

HolidayListing.aspx

```
<div>
Placeholder key="maincontent"
<div>
  Battle of the Hills
  </div>
  Introduction.aspx
  </div>
  Trip Overview.aspx
</div>
```



#### What Are Placeholders?

Placeholders are Sitecore controls

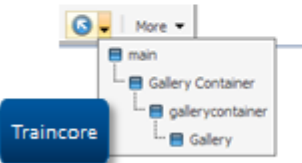
They are identified by an attribute called **Key** (ID not required)

Can be added anywhere in an .aspx, .asp, or .xslt

Define where components can be dynamically bound to an area of the page

Placeholders are added **directly** into code:

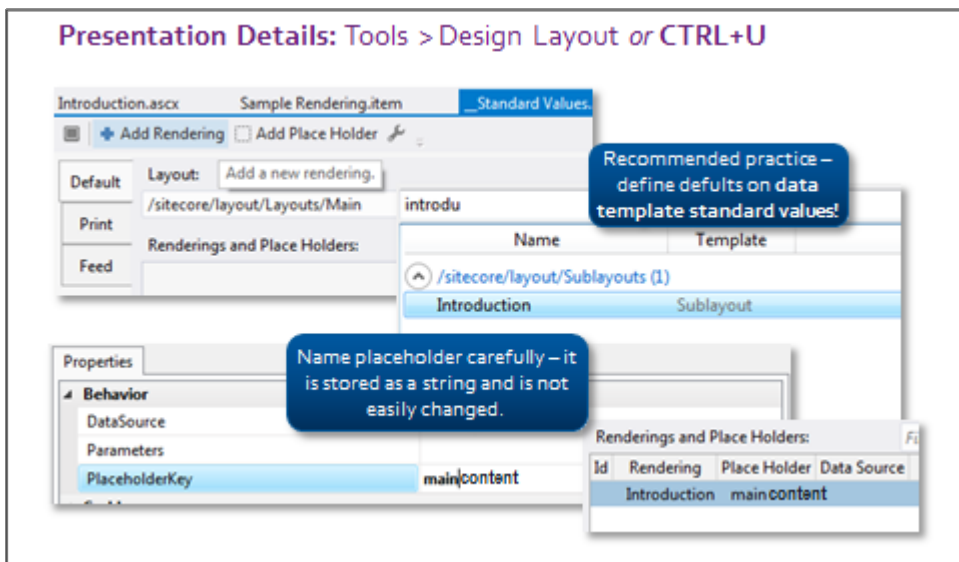
```
<sc:Placeholder Key="maincontent" runat="server" />
```



Lowercase not mandatory, but makes it easier to distinguish between component and placeholder.

(There are **placeholder settings** items – required by Page Editor. More in **Module 7!**)

## 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 **Holidays 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 be assembled from existing components

Changes to page structure do not require a developer

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

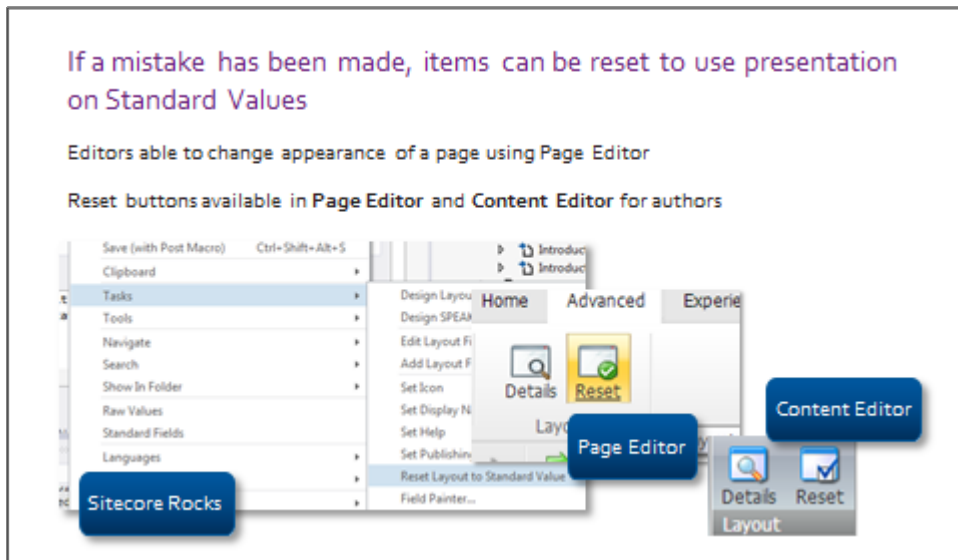
Supports Sitecore's personalization and testing features (more in Module 10!)

### Static binding

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

```
<sc:Sublayout Path="/layouts/Header.aspx" 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 <div> 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 <div> 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**

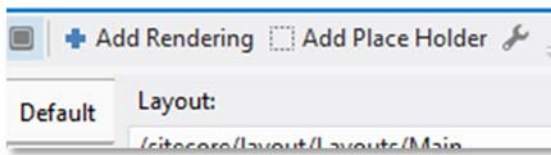
*Optional detailed steps are available below to help you.*

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

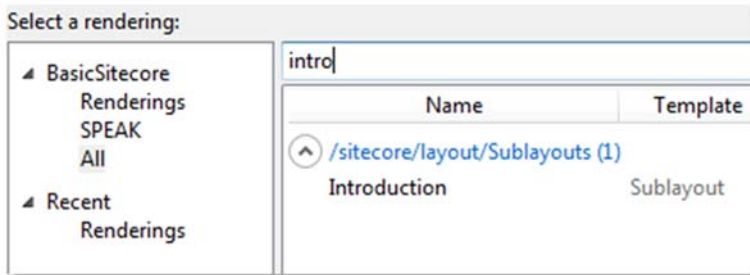
**Detailed Steps**

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

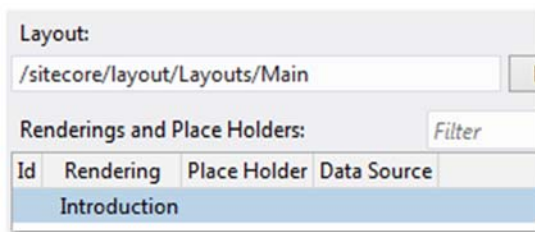
- Using the Sitecore Explorer, locate the **Base** data template's standard values: */BasicSitecore/master/sitecore/Templates/User Defined/Base/\_\_\_Standard Values*.
- Right-click the **\_\_\_Standard Values** item and select **Tasks>Design Layout**.
- In the **Presentation Details** window, click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



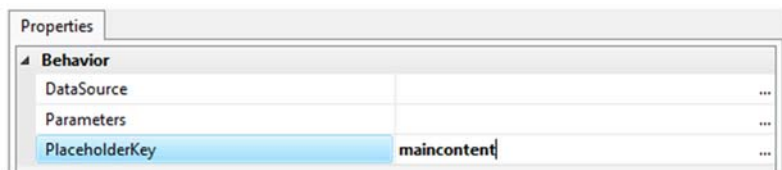
- Enter **Introduction** in the Search field. Ensure that **All** is selected on the left-hand menu:



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



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



- 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.

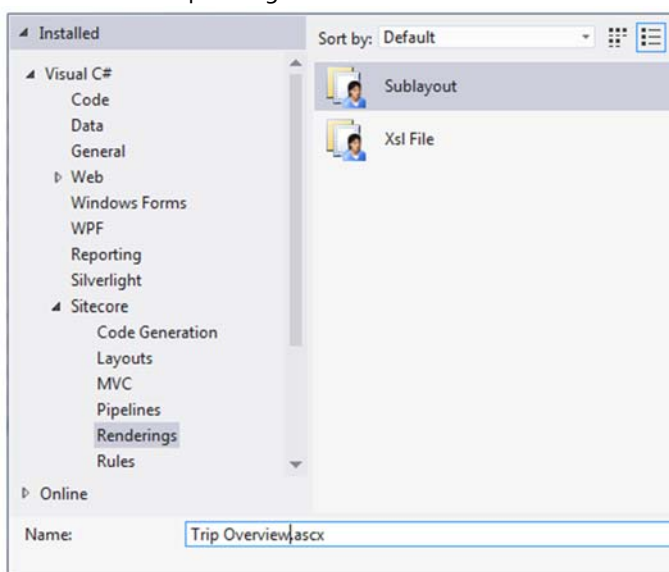
**Hint:** This data template inherits from base. When prompted, copy the layout from the standard values item.

**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:

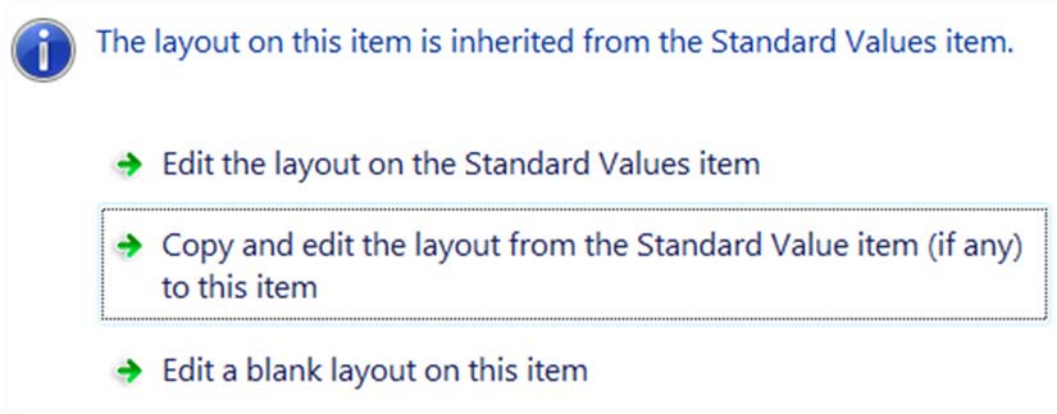
### Code Sample

```

<!-- 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 -->

```

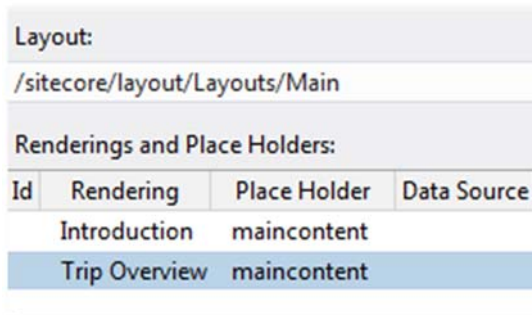
6. **Build** the solution and click **Save**.
7. Using the **Sitecore Explorer**, locate the **Trip Details** item under: */BasicSitecore/master/sitecore/Templates/User Defined/Trip Details*.
8. Double-click the item to open it.
9. Right-click inside the grey area in the right-hand content panel and 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.

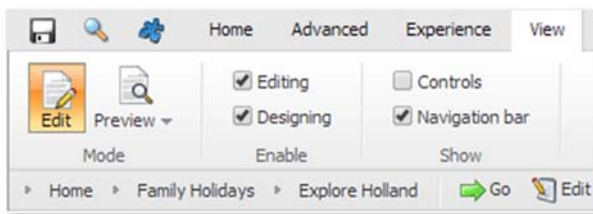


15. In the **PlaceholderKey** field, type *maincontent*.



Layout:			
/sitecore/layout/Layouts/Main			
Renderings and Place Holders:			
Id	Rendering	Place Holder	Data Source
	Introduction	maincontent	
	Trip Overview	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

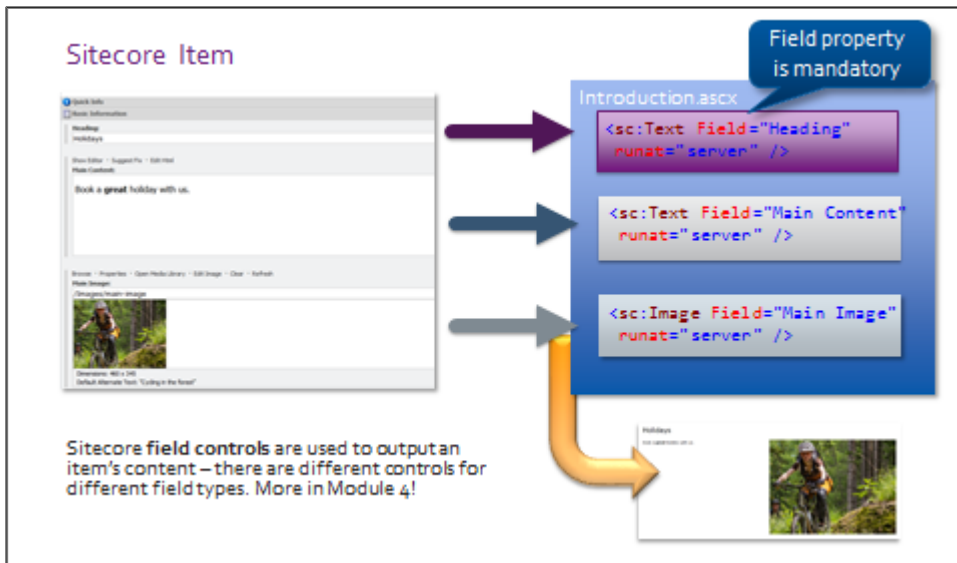
#### Objectives

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

Depending on the control, Sitecore offers a different set of options

On Sitecore's Image control, you can set:

CssClass for a particular instance of an image

Dynamically resize with MaxWidth or MaxHeight (caches an image of that size on the server!)

```
<sc:Image Field="Main Image" MaxWidth="200" runat="server" />
```

Your settings in code trump author settings

If you set an image Alt text in code, an author's changes in the Content Editor will not take effect. Assess whether you want authors to have control of this property.

## 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

##### Presentation details

An item's presentation details are instructions that tell it which layout and components to use, and which placeholders to bind the components to.

##### Layout

A layout is your page canvas. It is linked to an `.aspx` file on the file system, which may also be referred to as a 'layout'. You can have one layout per item *per device*.

##### Component

A unit of functionality. All components have a definition item in the Sitecore tree, and a file on the file system. Components can be: Sitecore sublayouts (`.ascx`), XSLT renderings (`.xslt`), or web controls (`.cs`).

##### Placeholder

A placeholder is a Sitecore control that determines where on a page a component can be inserted. Inserted amongst regular HTML markup, and must have a key.

### The Renderings Field

#### All presentation is stored as custom XML in the Renderings field

When presentation details are updated, the raw value of the renderings field is updated. This XML contains all information about layout, devices, components, and placeholders.

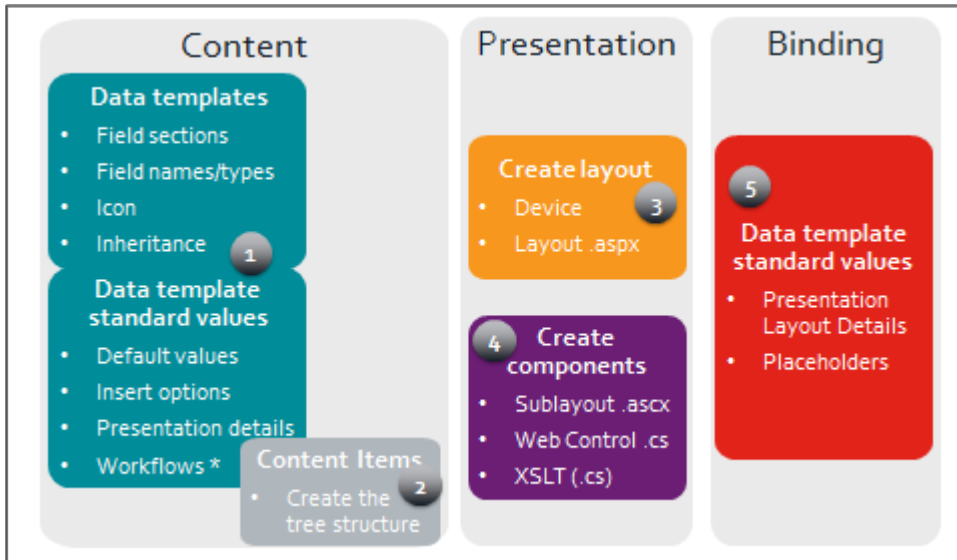
##### Layout

\_Renderings ▾:

```
<?xml version="1.0" encoding="utf-8" ?>
<d id="(FE5D7FDF-89C0-4D99-9AA3-85FBD009C9F3)" l="(7030A208-2D89-441A-9843-6932E217D80F)">
  <r id="(DB023D16-EBAS-43AA-AD44-E353CD388E8D)" ph="main" uid="(FC8E67B0-84FB-4A08-8F05-73617F4D783F)" />
  <r id="(91592835-7CB6-4726-9DC5-75CF383540E3)" ph="main" uid="(27D55CF5-8591-42E6-BFES-BA085D5D7387)" />
</d>
</?xml>
```

On an item, this field contains a **layout delta**. More on that in Module 5!

## 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

##### 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`



#### 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     {
  
```

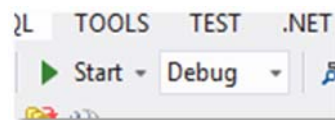


### Demo – Basic API

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

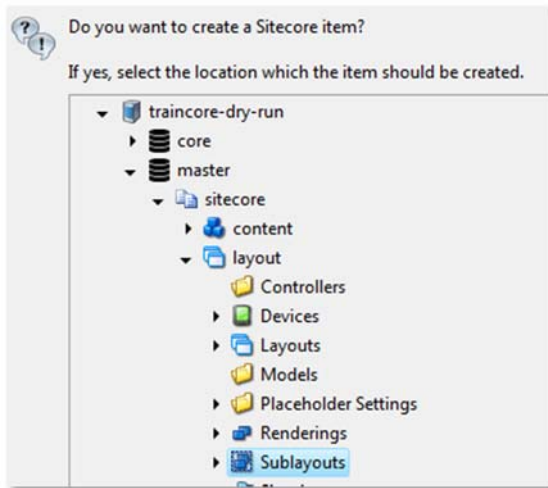
In the following lab, you will:

- 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.

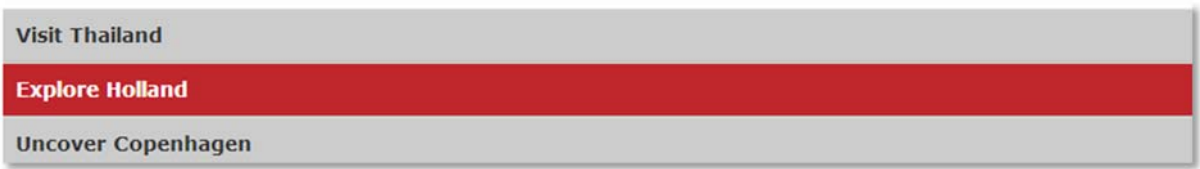
#### Lab A. Build a Navigation Sublayout

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.

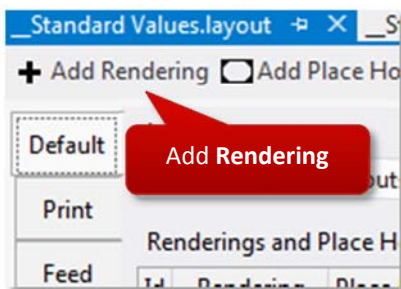
3. You will be prompted to add an accompanying sublayout definition item. Select your **location** in the **Sitecore tree**.



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](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

The Sitecore ItemResolver resolves a URL to an item and can interpret various different URL formats:

<http://mysite/...>

```
Family Holidays
Family Holidays/
Family Holidays.aspx
Our%20Holidays.aspx (alias)
en-us/Family Holidays.aspx
Family Holidays.aspx?sc_lang=se-SE
?sc_itemid={F02D3ACC-45B1-45A1-9BD2-C19B9D81099E}
~/link.aspx?_id=F02D3ACC-45B1-45A1-9BD2-C19B9D81099E&_z=z
```

Display names allow alternative item names for different languages so if the holidays site was translated into Danish, the content editor would see 'ferier' rather than 'holidays'

```
item.Name = Holidays
item.DisplayName = Our Holidays
item.ID = {F02D3ACC-45B1-45A1-9BD2-C19B9D81099E}
```



#### Knowledge Check

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

## LinkManager and Item URLs

Always use the **LinkManager** to output an item's URL  
 Use `LinkManager.GetItemUrl()` – recommended practice; SEO-friendly URL  
 Do NOT use `LinkManager.GetDynamicUrl()` – URL with a GUID!

May require IIS configuration changes

Customize URL with `UrlOptions`...

Language embedding, use .aspx, show language-sensitive display name

...or globally in web.config

Override **LinkManager** completely

Change individual settings

```
<linkManager defaultProvider="sitecore">
  <providers>
    <clear/>
    <add name="sitecore"
      type="Sitecore.Links.LinkProvider, Sitecore.Kernel"
      addAspxExtension="true"
      alwaysIncludeServerUrl="false"
      encodeNames="true"
      languageEmbedding="asNeeded"
      languageLocation="filePath"
      shortenUrls="true"
      useDisplayName="false" />
  </providers>
</linkManager>
```

### 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);
```

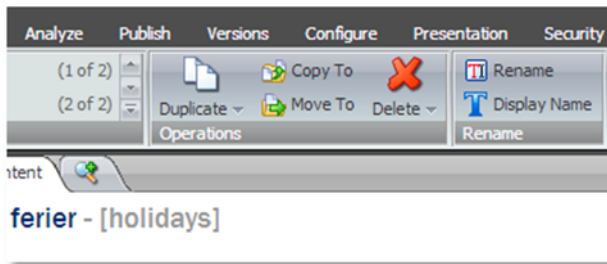
4. In the same file, *before* retrieving the context item's URL, create a `UrlOptions` object and set the `UseDisplayName` property to `true`.
5. Pass the object into the `Sitecore.Links.LinkManager.GetItemUrl()` method along with the context item.

### 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

 An infographic titled 'Controlling Your URLs' with a light blue background. It contains three main sections:
 

- Decide how you want to render links:** Individually or globally. Accompanied by an icon of a blue chain link and a globe.
- URL resolution handled automatically:** Regardless of how a URL comes in, Sitecore knows which item to map it to. Accompanied by a blue arrow pointing right with the word 'URL' inside, and a red circular icon with a white 'S' and the word 'ITEM' next to it.
- Multiple URLs for a single page can be confusing for:** Google Analytics, JavaScript third party services, and Web Log Analyzers. Accompanied by the Google Analytics logo, the 'Js' logo, and a line graph titled 'Daily Page Access' showing data over time.

### **i** 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-LineText	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

There are 7 steps:

1. Get master database – *do not* edit directly in web!
2. Change security context
3. Retrieve existing item or create new item  
(To create a new item, you need a parent item and data template)
4. Put item into an editing state: `item.Editing.BeginEdit();`
5. Set the item's field values to the new values
6. End editing to save changes: `item.Editing.EndEdit();`
7. You must **publish your changes** if you want them to appear in the web database



## Security and Editing Contexts

Security context	Editing context
Requests are made in the context of <b>anonymous</b> or <b>currently logged-in user</b> . You can <b>force</b> code to run regardless of <b>context user's permissions</b> :	Editing and renaming must happen in <b>editing context</b> .
<b>Switch to another user</b> <pre>using (new UserSwitcher(user)) {}</pre> <p>(This method is <b>safer</b> in production, however you can also use Security Disabler)</p> <pre>using (new SecurityDisabler()) {}</pre>	<pre>item.Editing.BeginEdit(); // Editing here item.Editing.EndEdit();</pre> <p>Calling <b>.Editing.EndEdit()</b> <b>saves your changes!</b></p>

### 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

It is unlikely that you will create items like this in production

No access to master database from CD environment

What are my options for adding content?

Item Web API (bundled with 7.2+)

Custom web API or service

Third party service (high volume user-generated content, such as comments)

Should user-generated content live in Sitecore?

Do authors need access?

Is this content better suited to an external system?



### 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**

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

Field section	
<b>Comment</b>	
Field name	Field type
Comment Author	Single-line text
Comment Text	Rich Text

- Assign** a template icon (for example, user1\_message.png).
- 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**

- Create a new sublayout called **Comments Form** (use **Add > New Item...** and select **Renderings > Sublayout** from the **Sitecore** menu).
- 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*).
- Assign the sublayout to the **maincontent** placeholder on the **Trip Details** data template **standard values**.
- Publish** Sitecore.
- 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.

- 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.IsNow` 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](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

Field type determines

Sitecore API field class      Type of source accepted

Raw value      Control in Content Editor

The screenshot shows the Sitecore Content Editor interface. On the left, the 'Basic Information' tab is active, showing a 'Single-Line Text' field with a 'Versioned' dropdown and '0 validations'. The 'Data Source' is set to a GUID. Below this, a 'Parameters' list includes 'Data Source (datasource)', 'Query (query)', 'Database (database)', 'Database (databasename)', 'Bind Mode (SPEAK)', and 'Property Type (SPEAK)'. A callout bubble labeled 'Final result' points to the 'Data Source' field. On the right, the 'Control in Content Editor' view shows a rendered image field with a callout bubble labeled 'View of control in Content Editor'. A 'Press Build' callout bubble is positioned over the 'Build' button in the top right corner.

#### Simple Fields

Simple fields contain text only

Field	Raw value	Source	Control	Class
Singe-line text	Text	N/A	<sc:Text />	Field
Multi-line text				
RichText Field	Text	Path, Guid	<sc:Text />	Field
Integer	Text	N/A	N/A	Field
Droplist				
Grouped droplist	Text	Path, Guid	N/A	GroupedDroplistField

## 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		Query, Path, GUID		
Multilist	Pipe-delimited IDs	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&amp;_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="{540898F1-790F-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

```
{540898F1-790F-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:

```

```

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");
```

### 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

Sitecore controls use this method too!

## 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 `CommentsForm.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 `CommentForm` 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
<code>.URL</code>	The contents of the link input (e.g. <code>AuthorWebsite.Text</code> )
<code>.Text</code>	The <i>same</i> contents of the link input ( <a href="http://www.sitecore.net/">http://www.sitecore.net/</a> )
<code>.Target</code>	"_blank"
<code>.LinkType</code>	"external"

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

- 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.

- 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.
- Add the new sublayout to the **standard values** of the **Trip Details data template**, below the **Comments Form** sublayout:

Id	Rendering	Place Holder	Data Source
	Introduction	maincontent	
	Trip Overview	maincontent	
	Comments Form	maincontent	
	Comments List	maincontent	

- Save** and **publish** Sitecore.
- Browse** to a trip details page to confirm that the sample HTML appears.
- 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*).
- Open the *CommentsList.ascx* **code behind**.
- 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.
- 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:

- Comment Author (**Text**)
- Comment Text (**Text**)
- Comment Author Website (**Link**)
- 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:

**Martina Welander by 10/05/2013 08:31:12**

I really loved this holiday.

<http://www.google.co.uk/>



### 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 Title<sub>1</sub>, Quote Title<sub>2</sub>, Quote Author<sub>1</sub>, Quote Author<sub>2</sub> 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?

Allow your component to display content from an item that is not the context item

**Home Page.aspx**

**WHICH BIKE?**

Don't know which one of our bikes to choose, or want your own bike is suitable? Let us guide you through your choice.

**LET US HELP YOU PICK A BIKE**

**Benefit?**

Reusable content and components and prepares for Experience Marketing implementation

### Datasources Allow...

A component to take data from an item that is not the context item

Content and components are reused and Experience Marketing ready

You can use GUID's and physical paths to items in the datasource field

**Component**

**Component**



### 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

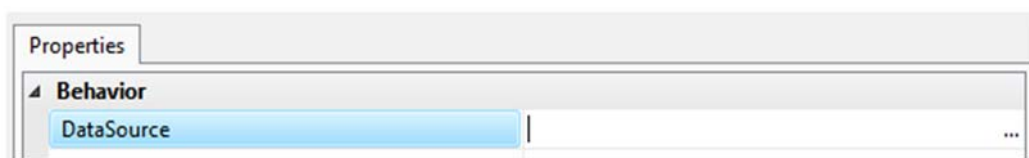
What if your component always had to accept a datasource? You can hide your controls by checking what mode the page is using in Sitecore. `Context.PageMode` then performing your logic.

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.

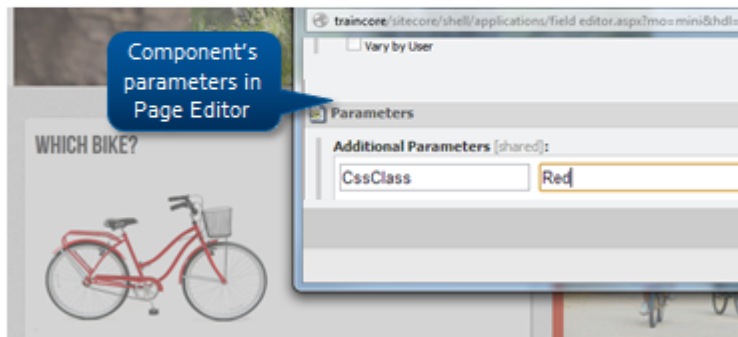


## Component Rendering Parameters

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 browser 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?

### Both directly on the item

- Component appears on that item only and must be added by author
- Datasource appears on that item only and must be set by author

### Both on standard values

- Component appears on all items sharing that particular data template
- Datasource appears on all items sharing that particular data template

### Component on standard values, datasource on item

- Component appears on all items sharing that particular data template
- Datasource varies per item, and must be set by author

## 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 -->
     <!-- 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.

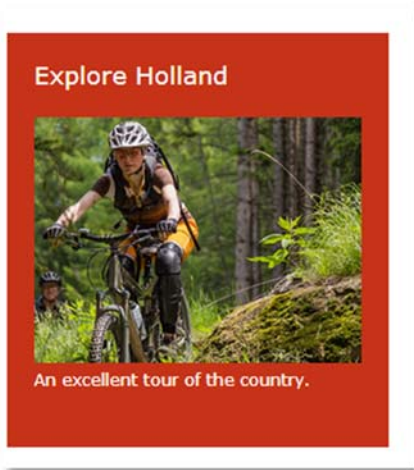


- In the **Page\_Load** method of the featured **Featured Trip.ascx.cs** file, get the component's datasource using the following code:

### Code Sample – Datasource

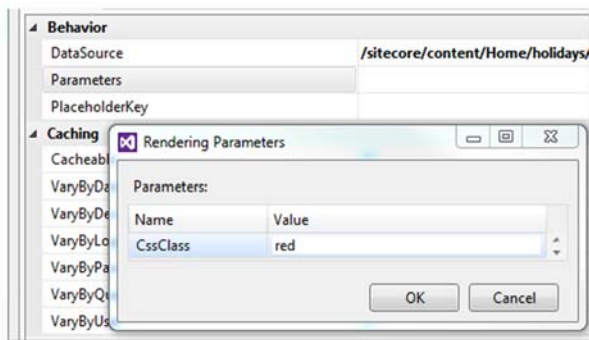
```
var sublayout = Parent as Sublayout;
var datasource = sublayout.DataSource;
```

- In the **Featured Trip.ascx.cs** file, set the **.DataSource** property of each control to the **datasource** item.
- Click the **Save** button to publish the solution.
- 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

- Using Sitecore Rocks, navigate to the Family Holidays page where you added the **Featured Trip** component.
- In the **Design Layout** pane, double-click the **Featured Trip** component to open its **Properties**.
- Click the **Ellipses ( . . . )** button next to the **Parameters** property.
- 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.



- In **Featured Trip.ascx.cs**, get the component **parameters** as a **string** (use `((Sublayout)Parent).Parameters`).
- 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?

When presentation details are overridden on an item (differ from presentation details defined on the standard values) – layout deltas get created

When developers update standard values, the item gets updated via the delta

View which items have presentation overridden by ..

Right-click on Quick Action Bar

- Missing Versions
- Publishing Warnings
- Validation Rules
- Presentation Overridden
- Refresh

Presentation is overridden for this item.

Cycle California

Discover the Cotswolds

#### A Layout Delta Is Like a Transform

Presentation details are stored as XML in the Renderings field.

```

<r>
  <d id="{FESD7FDF-89C8-4D99-9AA3-85F8D009C9F3}" l="{7030A208-2D89-441A-9843-6932E217D88F}">
    <r id="{D8023D16-EBA5-43AA-AD44-E353CD388E8D}" ph="main" uid="{FC8E67B0-84FB-4A08-8F05-73617F4D783F}" />
    <r id="{91592835-7C86-4726-90C5-75CF383540E3}" ph="main" uid="{27055CF5-8591-42E6-7F5-BA08505D7387}" />
  </d>
</r>
    
```

Un-edited standard values presentation

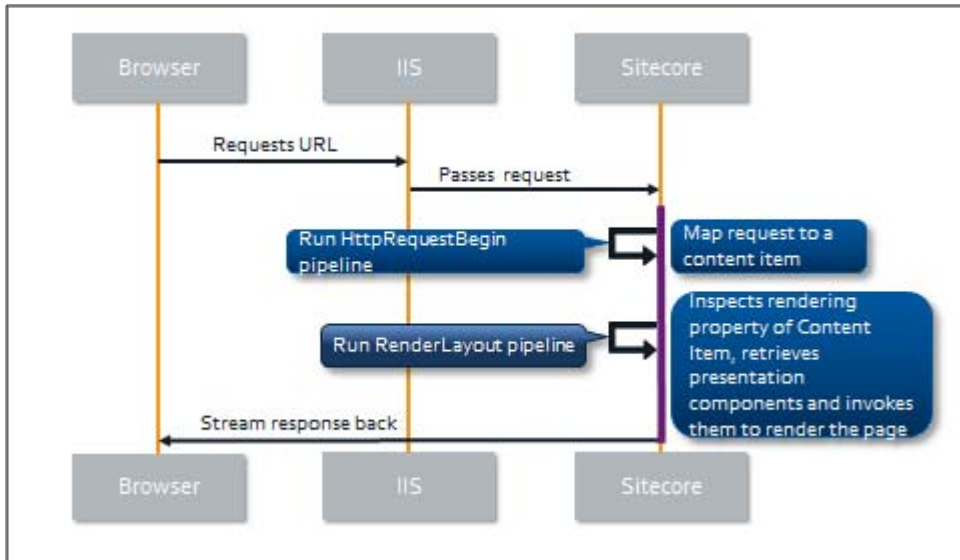
```

<c xmlns:p="p" xmlns:s="s" p:p="1" s:sxd="http://www.w3.org/2001/XMLSchema">
  <d id="{FESD7FDF-89C8-4D99-9AA3-85F8D009C9F3}">
    <r uid="{99269CAA-F5AF-4259-A9D4-A97D9E2EA1DA}" s:id="{CDD3F213-81BB-4770-8FEC-4E10D65EAA66}" s:ph="main" />
  </d>
</c>
    
```

Layout delta containing only changes to standard values

When an item is requested, the layout delta is applied on top of the standard values XML. It can add, remove, or edit

### 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

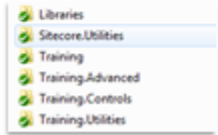
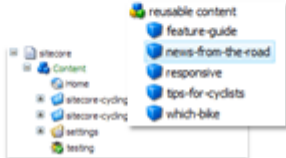
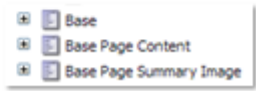
Traincore defines main and footer navigation in a separate folder under: */Global* rather than generating it from the site structure. This is a much more inexpensive query; it allows you to include external link items and makes it easy to see at a glance what the site’s navigation is going to look like.



### 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

Project Structure	Topology	Templates
<p>Abstract functionality in <b>separate projects</b> – for example, <i>Sitecore.Utilities</i></p> <p>Patch in config changes <i>/App_Config/Include</i></p> <p>Reference .dlls in a <b>Libraries</b> folder</p>	<p><b>Subfolder</b> items for sites</p> <p>Separate repository of <b>reusable content</b> – no presentation!</p> <p>Shared <b>settings</b> – uses shared <b>fields</b></p>	<p>Organized into <b>subfolders</b></p> <p>Author-friendly names</p> <p>Use of <b>icons</b> (fields and field sections)</p> <p>Prevent <b>field redundancy</b> – use <b>multiple templates</b> (Base Page Content)</p>
		



## 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

**Levels of Nesting**

- Layout – e.g. BaseCore
- Containers – e.g. Page Container
- Columns – e.g. Three Column
- Functional blocks – e.g. Widget

**Components can be moved & re-used**

- Containers can be re-ordered
- General Widgets can appear in columns, the footer, and widgets container
- Appearance and content changed using parameters and datasources

### Supporting the Page Editor

Componentize, componentize, componentize

Make HTML developers aware of componentization requirements early

Separation of data and presentation

Use the FieldRenderer and Sitecore controls

Dynamic binding for movable components, static binding for components with fixed locations

**Why Build for the Page Editor?**

- Must form part of requirements – your clients were sold this functionality
- Much easier to change appearance of a page in future if presentation is modular
- Building for Page Editor means building for Sitecore's marketing features

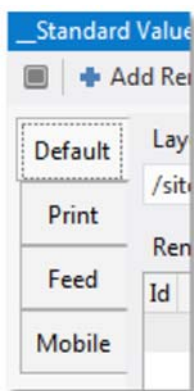


## Demo – Create a Mobile Site

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

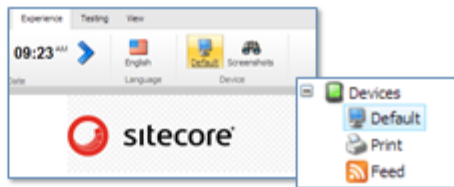
Sitecore is set up to support adaptive design; different presentation, same content, depending on what is being used to access the content

Achieves this using devices – represented as definition items in Sitecore

Configure presentation details per item (or item type), per device

Change device using `?sc_device={ID}` or a custom query string defined on definition item

Use custom code to determine the type of device that is accessing Sitecore, and set context device accordingly

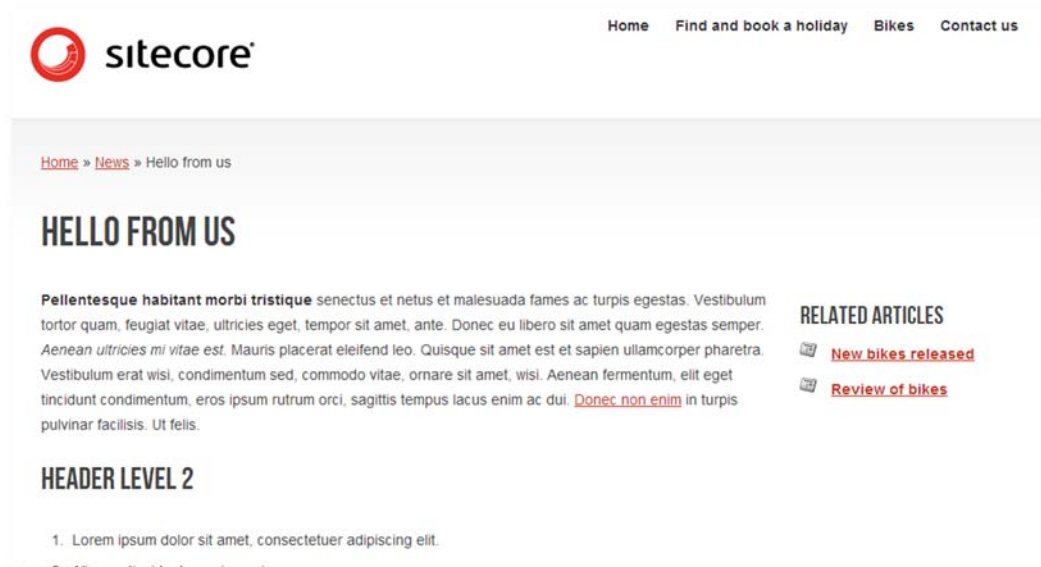


## Apply – Topic 6.1 – 60 min

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?

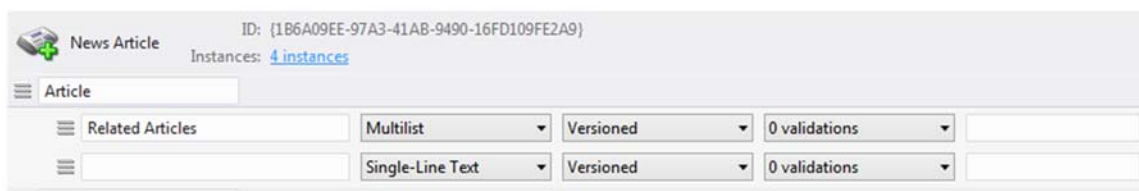
2. Select the following base templates: **Base Page**, **Base Page Content**, and **Base Page Summary Image**.

**i** 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
<b>Article</b>	
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**

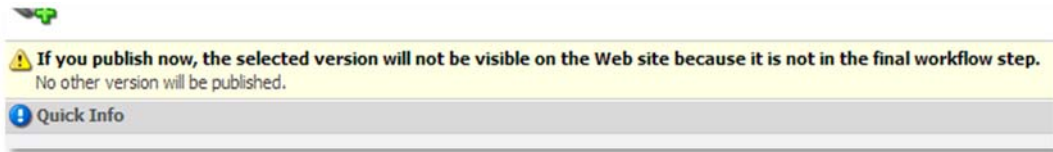
1. Create the News Article template's standard values.
2. On the template's standard values, check that the *\$name* token appears in **Page Title**, **Page Heading**, and **Navigation Title**.
3. Navigate to the **News Listing** template's standard values and set *News Article* as an insert option.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

**i** 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:

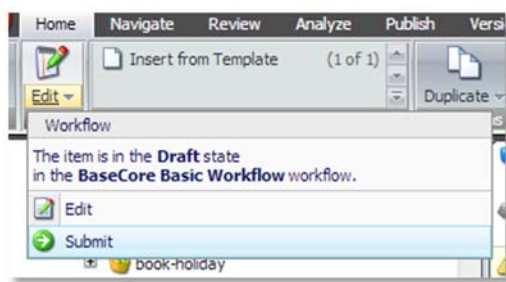


- Populate each item with sample content. For each article item, select two or more **related articles** in the **Related Articles Multi-list**.
- The following will appear at the top of each news article item:



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).

- 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.



- 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

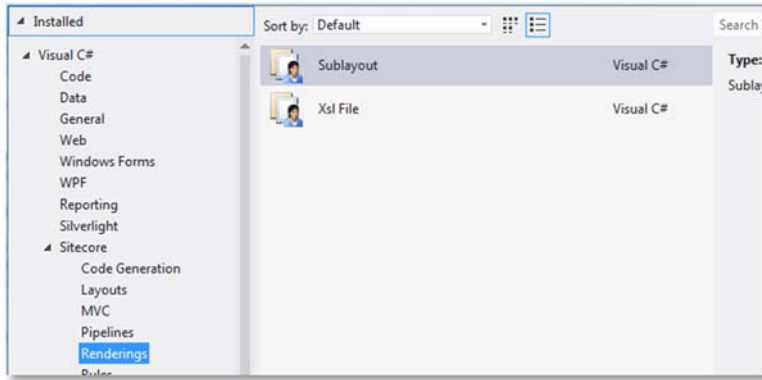
In the following labs you will:

- 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

- 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*.

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



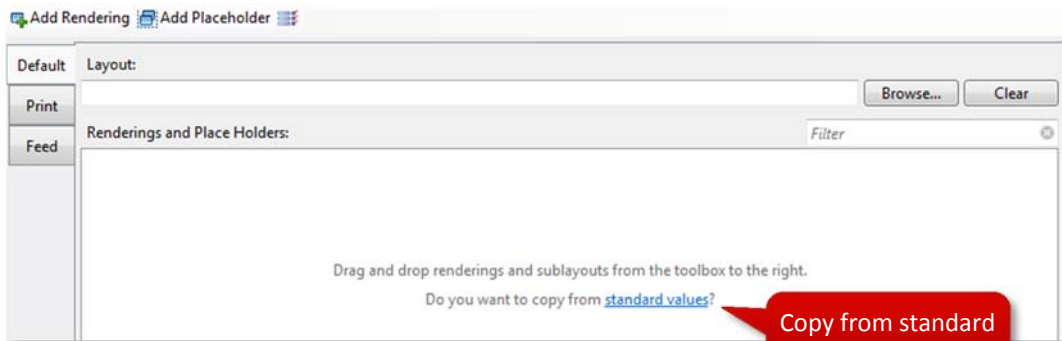
- When prompted by Sitecore Rocks, add the corresponding **sublayout definition** item to `/Traincore/master/sitecore/layout/Sublayouts/BaseCore/Content/`.



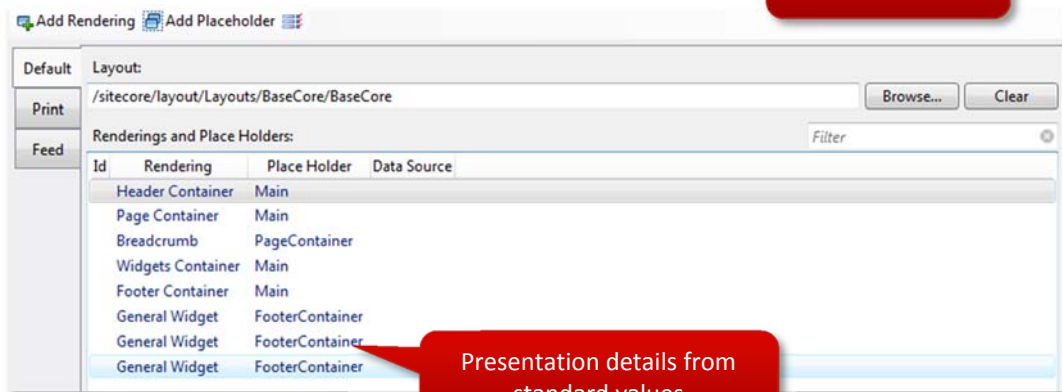
**Tip**

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.

- 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.
- Navigate to the **News Article** template's **standard values** and open the **Design Layout** pane.
- Copy** presentation details from standard values when prompted.



Copy from standard values

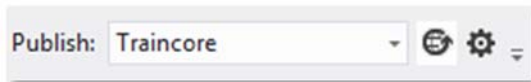


Presentation details from standard values

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 %>"
```

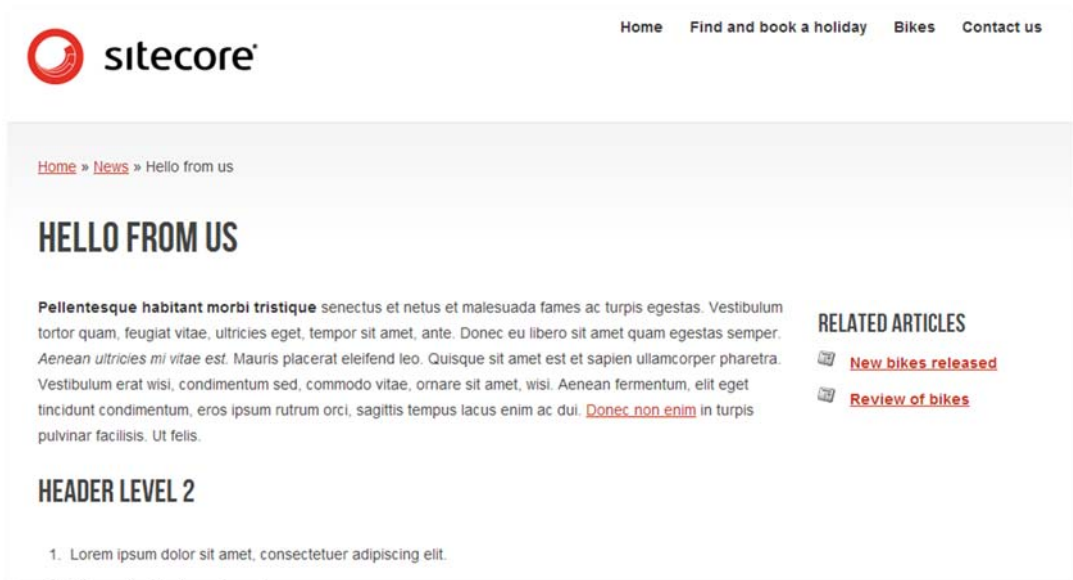
7. The sample repeater also prompts you to add a URL. Replace **[URL HERE]** with a **method** that retrieves the **SEO-friendly URL**. (**Hint**: refer to Module 4 if you are unsure of which method to use).

### 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


**Things to consider when building a multi-language site:**

- Components must be able to support different content lengths
- All content must be translated—including
- Some field values need to be shared across languages
- Build feature that allows users to change the site language (query string, stored in cookie)
- Restrict author access to their languages only



**Consider language settings:**

```
<setting name="DefaultLanguage" value="en-US" /> (default)
<site _ language="en-US" /> (per site)
<setting name="ClientLanguage" value="en" /> (language pack)
```



#### Remember!

**Handle items that do not have language versions**

When looping through a list of items, check if there is a version available in your language (items without versions will still exist in the item.Children collection – avoid \$name!)

```
foreach (Item child in item.Children)
{
    if (child.Versions.Count <= 0)
    {
    }
}
```

**Custom code must take languages into account**

For example, Spider creating a Google sitemap

**Language fallback**

Consider how any fallback solution will affect performance

**Translation vs. Localization**

Are you writing content for a location or translating global content?

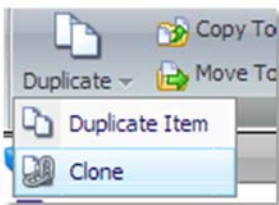


## Demo – Multi-site

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**:

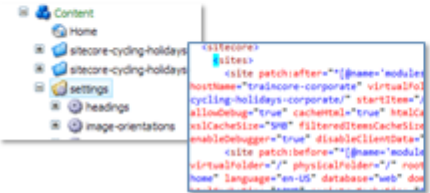


9. Choose **/sitecore-cycling-holidays-corporate/Home** as the item's new parent. **Note** that the cloned item's name is greyed out.
10. Make a change to the clone's **Page Heading** field.
11. Go back to the original item and make a change to the Page Heading of this item.
12. Go back to the clone.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

**⚠ A field in the original item has been changed.**  
 The Page Heading field in the original Special Offers item has been changed. Do you want to accept the change?

- Review the original item.
- Accept the change.
- Reject the change.

## Multi-site

<p><b>How?</b></p> <p>Multiple site root items in tree</p> <p>Shared content outside individual site structure</p> <p>Individual &lt;site /&gt; node in config (use include folder!)</p> <p>Different host name, start item, language, caching options</p> 	<p><b>Warning!</b> ⚠️</p> <p>Multi-site can become too complex!</p> <p>URL resolving/generation affected</p> <p>Shared templates/presentation or separate? <b>Duplication vs. complexity</b></p> <p>Complex security considerations</p> <p>Page Editor initializes in context of the site user logged into</p> <p>Sites share the same application pool (<i>licensing implications!</i>)</p> <p><b>Presentation details cannot be versioned</b> – one change affects all</p>
--	--

### ★ 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](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::*
```

- 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.
- Go to an item based on the **Holiday** data template – the selection is restricted by the query.

### Sitecore Query


Allows you to interrogate an XML representation of Sitecore tree

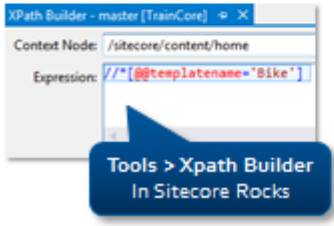
Similar to XPath, prefix with **query:** (*prefix not needed in Xpath Builder tool*)

Primarily used to limit selection in field source

Can contain axes (e.g. ancestor-or-self) and wildcards

Test with **Tools > XPath Builder** in Sitecore Rocks

Sitecore query is **not** Sitecore search – serious performance implications 



Tools > XPath Builder  
In Sitecore Rocks

**Context Node** is the item you want to execute the query from – **it's not part of the query**

**Escape** item names with **hyphens** with # (e.g. #sitecore-cycling-holidays#) in the **Expression** field

Make sure you use **straight quotes**, not **smart quotes** – for example, 'Bike'

### ★ 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

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

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

### Uses for Sitecore Query

**Source field**  
Any field that accepts a source – filter by name, template content, location (for treelists, consider **Treelist Parameters** as an alternative to Sitecore query)  
Prefix with **query:** when using in source fields

**In the API (but only sparingly!)**  
For example: `Sitecore.Context.Database.SelectItems("/sitecore/content/*");`  
For example: `Sitecore.Context.Item.Axes.GetDescendants();`

**For any complex querying, use Sitecore search! (Module 8)**  
Queries an **index**

**LINQ API**  
Sitecore query more likely to lead to **performance issues**



**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.

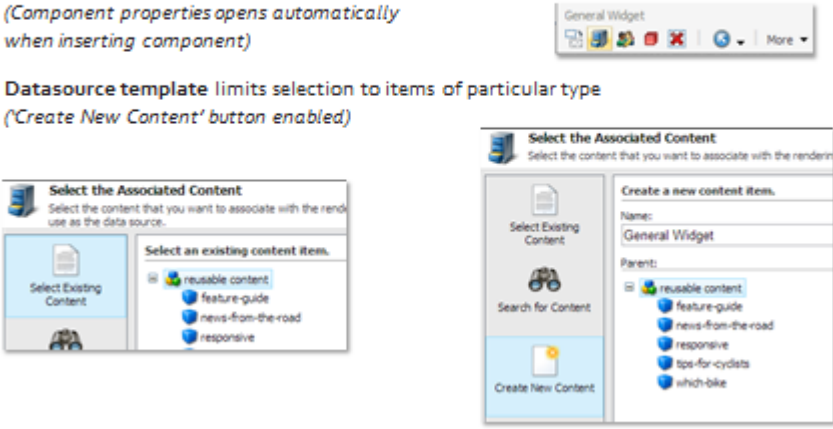
### Content

#### Datasource Location and Template Restrictions

**Configured on component definition item; use them together**

**Datasource location** limits selection to part of Sitecore tree  
(*Component properties opens automatically when inserting component*)

**Datasource template** limits selection to items of particular type  
(*'Create New Content' button enabled*)




#### 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

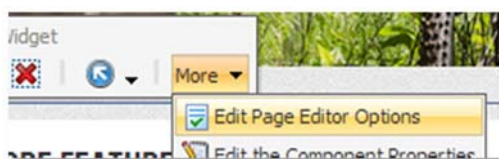
In the following labs you will learn to use the **Page Editor** to restrict the location and type of items authors can select as a datasource.

#### 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.

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



**Lab B. Add a Component**

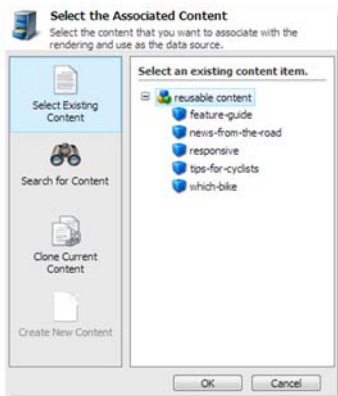
- Select **Add New Component** on the ribbon.



- Add a new **General Widget** to the `widgetscontainer` placeholder.



- 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**.



- Save** the page.

**Lab C. Set Datasource Template**

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

**Lab D. Add Another Component**

- Add another General Widget component to the `widgetscontainer` placeholder.
- 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.
- 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

**Represent parameters as real fields instead of key/value pairs**

**Widget Styling**

Class [unrendered]  
widget:orange

Heading [unrendered]  
strong

Widget Width [unrendered]  
widget-width:half

VS.

**Parameters**

Additional Parameters [shared]

Class

Heading

Widget Width

Base the template on the **Standard Rendering Parameters** template

Build up templates using regular field types (such as dropdown)

On the actual **component definition item**, assign the parameter template

Parameters Template \*

{9D197E70-5C33-489B-94C7-88C4CCB693C2} Browse...

/sitecore/templates/BaseCore/Layouts/Rendering Parameters/General Widget Configuration

Per instance of a component, you can now choose a value for each field

## 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`.



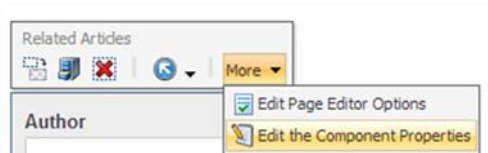
#### Knowledge Check

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.

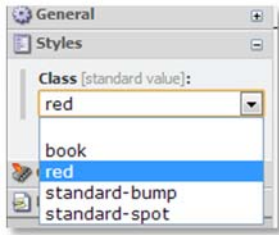


9. **Save** the sublayout.
10. Using the Page Editor, navigate to one of the news article that you created earlier. Select the **Related Article** component and in the floating toolbar and click the **More** button and then the **Edit the Component Properties** option.



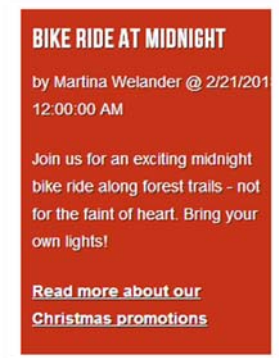
11. Confirm that the **Class** field now appears in the **Component Properties** window.

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

Placeholder settings items are definition items for placeholder keys

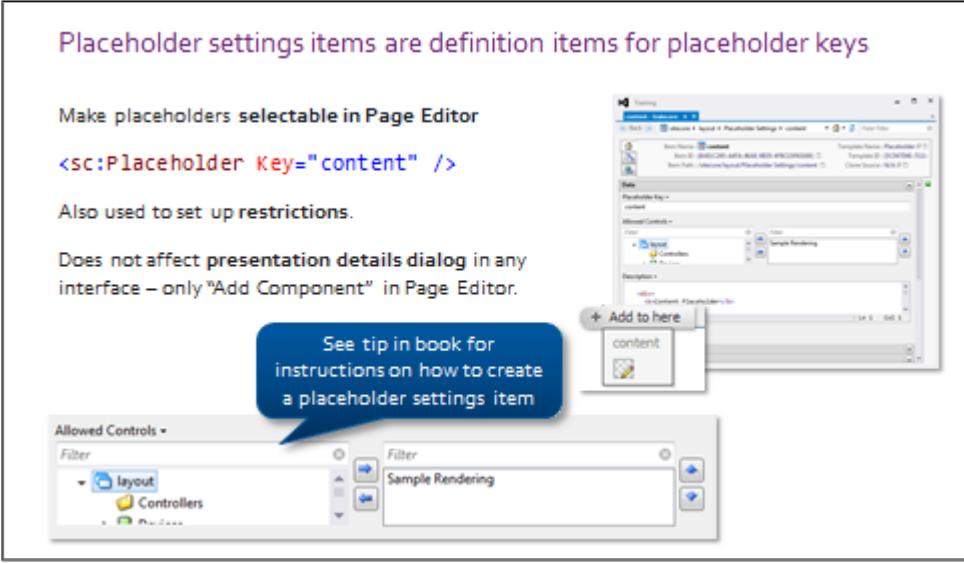
Make placeholders selectable in Page Editor

```
<sc:Placeholder Key="content" />
```

Also used to set up restrictions.

Does not affect presentation details dialog in any interface – only "Add Component" in Page Editor.

See tip in book for instructions on how to create a placeholder settings item



The image shows a screenshot of the Sitecore Page Editor interface. At the top, there's a text box explaining that placeholder settings items are definition items for placeholder keys. Below this, it says 'Make placeholders selectable in Page Editor' and shows the code snippet `<sc:Placeholder Key="content" />`. It then states 'Also used to set up restrictions.' and 'Does not affect presentation details dialog in any interface – only "Add Component" in Page Editor.' A blue callout bubble points to a 'content' placeholder in the 'Allowed Controls' list, with the text 'See tip in book for instructions on how to create a placeholder settings item'. In the background, a 'Placeholder Settings' dialog box is visible, showing fields for 'Placeholder Key' (set to 'content') and 'Compatible Renderings' (set to 'Sample Rendering').



#### Tip

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



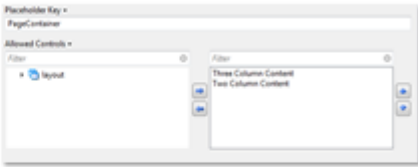
#### Tip

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

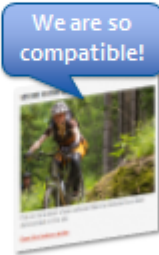



## Compatible Renderings and Allowed Controls

**Allowed controls**  
 Configured on **placeholder settings** item  
 Define **which components** can be added to that particular placeholder  
 If **none** selected, **everything** is allowed



**Compatible renderings**  
 Configured on **component definition** items  
 Must define **each other** as compatible  
 Both must be **allowed controls** of any placeholder you want to add them to  
 Keep in mind that they *may* need to share **parameters** or **datasources**





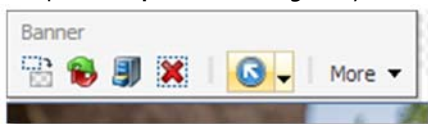
### Demo – Allowed Controls and Compatible Renderings

In the following demo, we will:

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

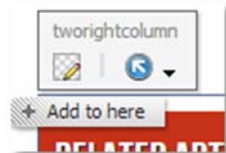
### Compatible Renderings

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**.



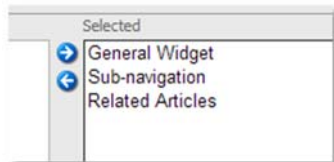


## Knowledge Check

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

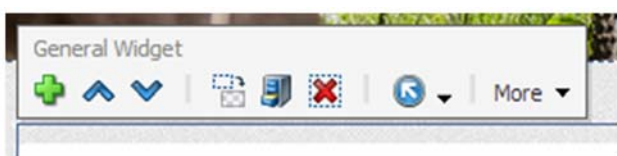
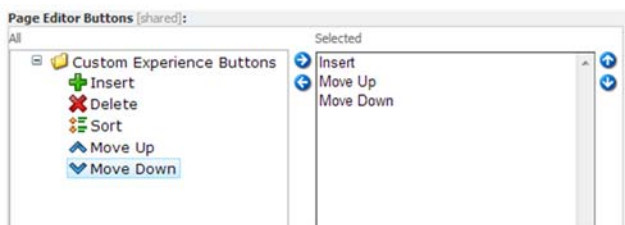
In the following walkthrough, we will:

- 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.

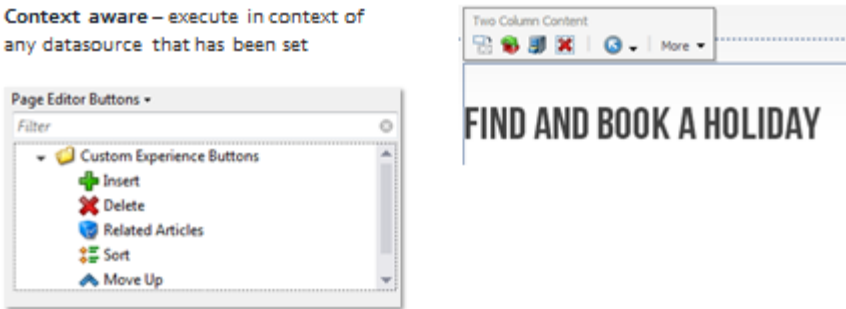
### Custom Experience Buttons

Floating toolbar around fields and components

List can be extended with custom experience buttons

Live in core database (`/core/sitecore/content/Applications/WebEdit/Custom Experience Buttons`)

Context aware – execute in context of any datasource that has been set



The image shows a floating toolbar with icons for undo, redo, and a 'More' dropdown menu. Below the toolbar is a text field containing the text 'FIND AND BOOK A HOLIDAY'. To the left is a 'Page Editor Buttons' panel with a 'Filter' input and a list of buttons: 'Custom Experience Buttons' (expanded), 'Insert', 'Delete', 'Related Articles', 'Sort', and 'Move Up'.

#### **i** 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`.

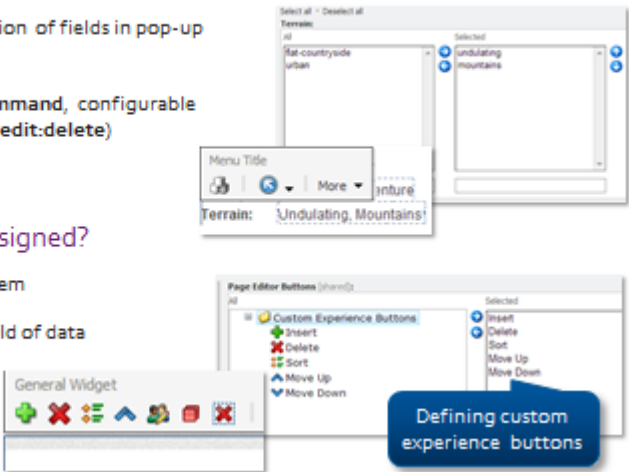
### Types and How to Assign Them

**Types**

- Field Editor** – Edit selection of fields in pop-up window (e.g. multilist)
- WebEdit** – execute a command, configurable (Delete button fires `webedit.delete`)

**Where are they assigned?**

- Component definition item
- Field definition item (child of data template item)



The image contains three screenshots. The top one shows a 'Terrain' multilist field editor with a 'Delete' button highlighted. The middle one shows a 'Menu Title' field editor with a 'Delete' button highlighted. The bottom one shows the 'Page Editor Buttons' panel with a 'Delete' button highlighted. A blue callout box with the text 'Defining custom experience buttons' points to the 'Delete' button in the bottom screenshot.

## 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)

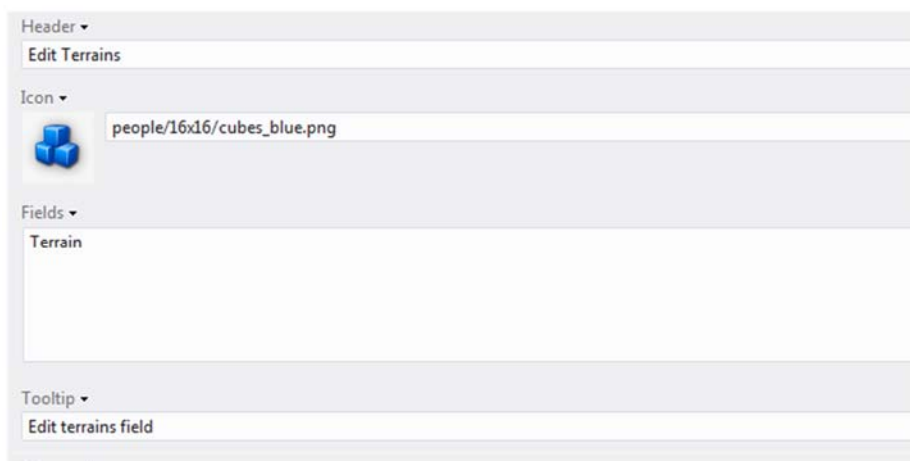


### Demo – Edit Frames

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/bascore-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.

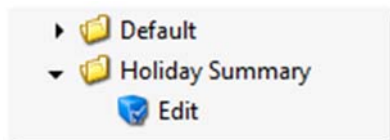
## Apply – Topic 7.4 – 15 min



### Edit Frame

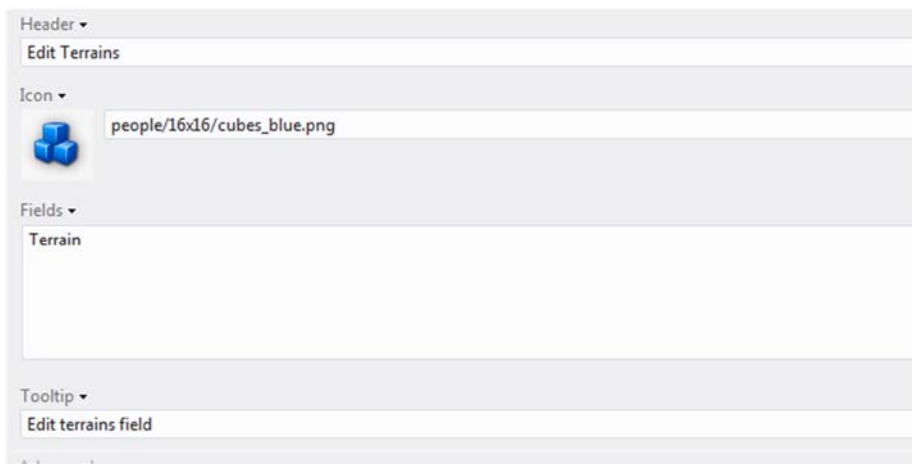
#### Lab A. Add an Edit Frame

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.

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

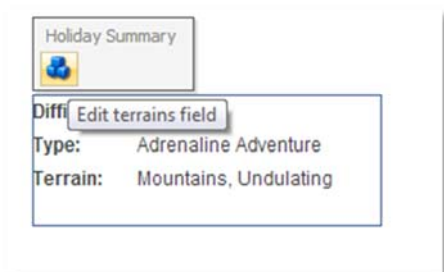


- Switch to **Visual Studio**.
- Open the **bascore-widget-holiday-summary.ascx** sublayout - `/layouts/BaseCore/widgets/bascore-widget-holiday-summary.ascx`.
- Surround the entire definition list HTML (see code sample below) with an `<sc:EditFrame />`.
- Set the **Buttons** property to the **path**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

### Code Sample – Edit Frame

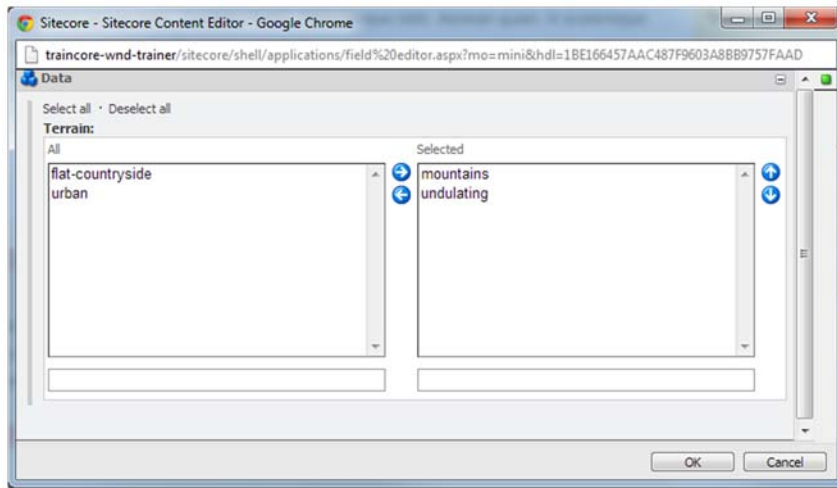
```
<sc:EditFrame ID="EditFrame1" Buttons="/sitecore/content/path/to/buttons" runat="server">
<dl>
<!-- ... -->
</dl>
</sc:EditFrame>
```

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



- 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

##### Reliance on hierarchical data

Not all data needs to be represented as a tree structure

##### Interface not designed for interacting with high volumes of data

Author must manually locate item in tree based on path

*In addition...*

##### Using API to interact with a large Sitecore tree is expensive and slow

Developers forced to interact with hierarchical tree structure – not efficient when dealing with large volumes of data (Topic 9.2)



#### Walkthrough – Using Buckets as an Author

In the following demo, we will:

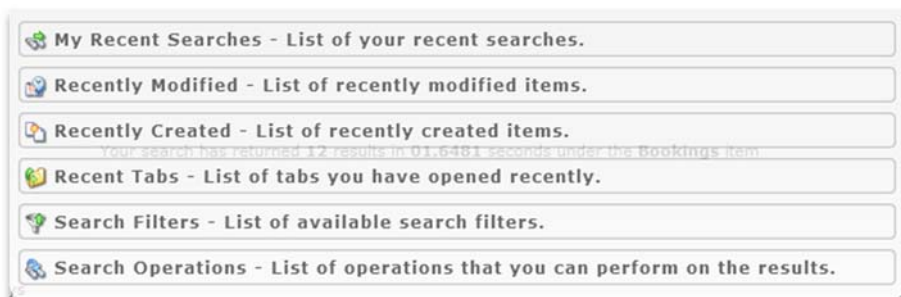
- Use bucket to **search** for bookings.
- Narrow down results using **facets**.
- Edit and create items as normal.
- Use **plain-text search** to search for a widget we know exists, for example, *which-bike*. Use **wildcards, replacements, and exact phrases**.
- Change the results view to **Gallery** 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.

- 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**).

- 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?

A bucket allows you to transform an item into a repository that stores other items **without displaying them in the content tree**

Potential to store millions of items, which a tree structure would not support

The screenshot shows a search interface with a search bar and filters on the right. On the left, a content tree is visible. A callout points to the 'Bookings' item bucket in the tree, stating: "Bookings' item bucket with hidden descendants within traditional tree structure". Another callout points to the search interface, stating: "Buckets search interface".

Your search has returned 2 results in 86,293 seconds under the Bookings item

Friendly search UI for authors to manage a large number of items

## When to Use a Bucket

### Advantages of Buckets

Support for potentially millions of items that would not suit a tree structure and slow down the UI – for example, products, repositories, orders

Automatically organizes content items into a format that improves **internal search engine performance**

Queries run against an **index**, ensuring high performance

Find what you want quickly by using **free text search**, **facets**, and **search filters**

### Should I bucket?

Bucket if:

- You **do not** care about the structure of the data stored in the bucket

- You anticipate a large volume of items

- You would benefit from searching for items rather than locating them in the tree



### 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:



**Unbucketable Item stored in an Item Bucket**  
This item is stored in an item bucket. However, it is not hidden because its template is not bucketable. If you want to hide this item in an item bucket, you must make the template that is based on bucketable.

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

### Any item can become a bucket

Items inside the bucket no longer have a parent / child relationship with *semantic meaning*, and are organized into folders based on creation time (by default, this structure is hidden)

Items in a bucket can still be created, edited and deleted by conventional means

If you want to retain the parent/child relationship (for example, comments should always be children of a news article), you can specify it on the template standard values of the intended parent

Bucket settings configure under: `/sitecore/System/Settings/Buckets`

Facets, number of items in search results, how results should open

**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

### Items that are going to be inside a bucket need to be bucketable

In practice, this means that the items will be **hidden** and restructured according to configuration

(Note that you can store regular content items in a bucket if you want to, but they will behave like regular content items – some bucket functionality will not apply.)

You can either make **individual items** or **templates** bucketable

Sync the bucket when:

- a) You create a new bucket
- b) You make items or templates bucketable
- c) You make items unbucketable



## 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.

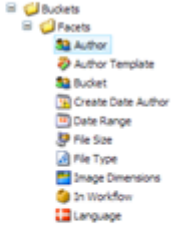
## Search Facets

**Facets allow you to narrow down your search results**

Configurable items in `/Settings/Buckets/Facets`

Specify index field to facet on – e.g., 'sizerange'; may be a special **computed index field** that does not exist in Sitecore

If you facet on a droplink, you would get a list of GUIDs to facet on – not useful, would require **custom index configuration**



## Apply – Topic 8.1 – 20 min



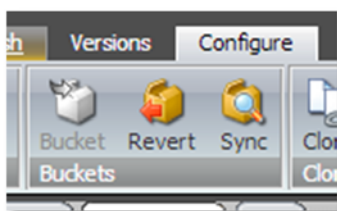
### Create a Bucket and Make Items Bucketable

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

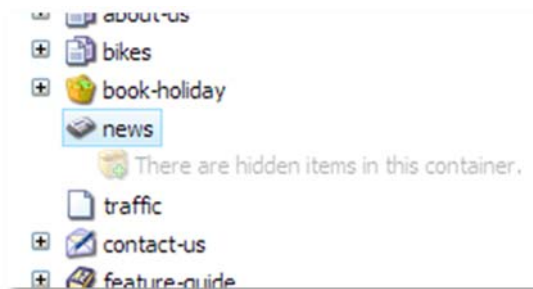
Expand the **News** item. Why are the news article items not hidden, even though the news item has been turned into a bucket?

**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

Item buckets uses Sitecore's search API

You can leverage the same API to build search functionality for your website's front end

```

var index = ContentSearchManager.GetIndex("sitecore_web_index");
using (var context = index.CreateSearchContext())
{
    var queryable = context.GetQueryable<SearchResultItem>()
        .Where(x => x["page heading"].Contains("bikes"));
}

```

Get index by name

Create a search context

Return an instance of IQueryable<T> and use standard LINQ statements

Return type



#### 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"));
```

- 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"));
```

- 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'

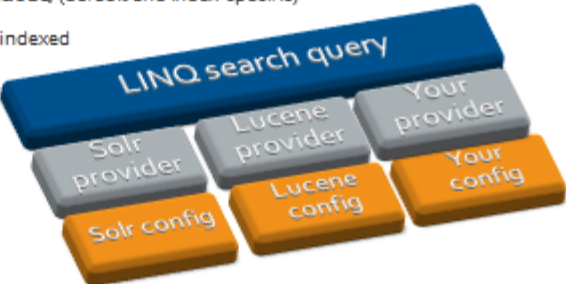
```
.Where(x => x.CreatedDate.Between(new DateTime(2013, 06, 03), DateTime.Now, Inclusion.Both));
```

## Basic Index Configuration

**Search works on an *index that is external to Sitecore***  
 Configuration in `/App_Config/Include/` (default and index-specific)  
 Determine **how** items should be indexed

**Uses provider model**  
 Identical LINQ layer  
 Provider **translates** to work with chosen search technology

**Which one should I use?**  
 Sitecore ships with **Lucene** and **Solr** (Solr wraps Lucene, more advanced)  
 Use **Solr** to scale



**i** 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](http://sdn.sitecore.net/upload/sitecore7/70/sitecore_search_and_indexing_guide_sc70-a4.pdf)

**i** 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.

**i** Tip

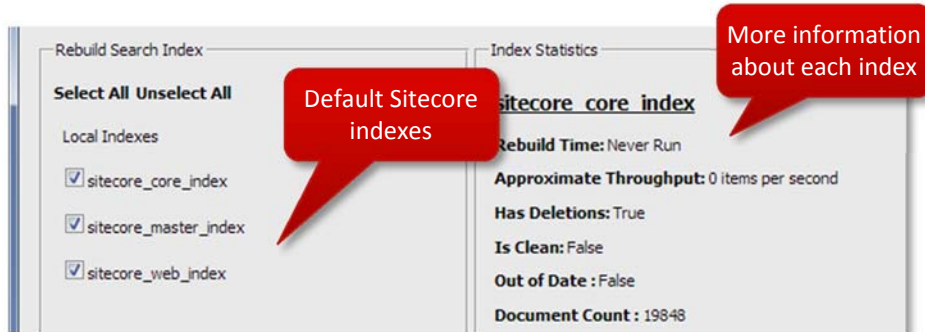
Solr supports distributed indexing, but Lucene does not. This means the search indexes can be offloaded to a separate server and then accessed by all CD servers.

## 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.

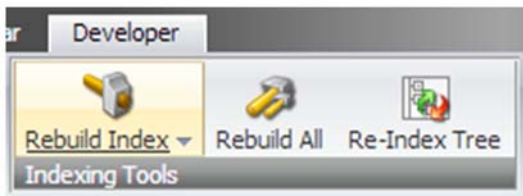


Notice that there is an additional option for Solr: by default, Sitecore defines three indexes – one per database (Master, Core, and Web).

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

 sitecore_core_index	14/04/2013 14:07
 sitecore_master_index	14/04/2013 14:06
 sitecore_web_index	14/04/2013 14:06

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
<code>.Where</code>	<code>.Boost</code> (make this part more important)
<code>.Any</code>	<code>.Page</code> (does skip/take)
<code>.OrderBy</code>	<code>.Between</code> (exclude/include start and end ranges)
<code>.Filter</code>	

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

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



**i** 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

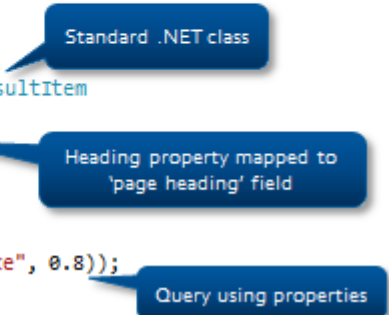
You create your own search result type

- 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`

```

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

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



**i** 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 facets 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

By default, the majority of index configuration is shared between indexes and stored in `.DefaultIndexConfiguration.config` per provider type

For Lucene, that file is:



Sitecore.ContentSearch.Lucene.DefaultIndexConfiguration.config

## Which Fields Should Be Indexed?

The more data you store, the larger the index

Make sure you have configured your index to include only what you need by controlling what is indexed

Selectively *exclude* or selectively *include* fields and templates

```
<IndexAllFields>true</IndexAllFields>
```

Settings can be overridden per index

```
<exclude hint="list:ExcludeTemplate" />
```

```
<include hint="list:IncludeTemplate" />
```

```
<include hint="list:IncludeField" />
```

```
<exclude hint="list:ExcludeField" />
```

Index all except explicitly excluded OR index nothing except included

Include or exclude specific fields or items based on specific templates

By default, the `ExcludeField` list has many standard fields that would not be valuable in an index – e.g., `Allowed_controls` and `UpdatedBy`.

### ★ 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?

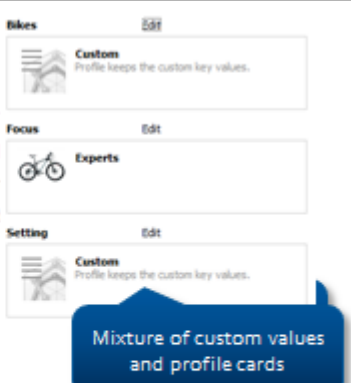
How a field is indexed is determined on a field type (e.g., individual field (e.g., the Page Heading field) level

```
<fieldTypes hint="raw:AddFieldByField"
<fieldNames hint="raw:AddFieldByField"
```

The configuration files allow you to control what goes in the index and how it is queried with increasing granularity

Can target all fields, individual types, or particular fields

You can completely override the analyzer and write a new provider if needed



```
<fieldType fieldName="sizerange" storageType="YES" indexType="TOKENIZED" vectorType="YES"
  <Analyzer type="Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer, Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer, Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer"
</fieldType>
<fieldType fieldName="title" storageType="NO" indexType="TOKENIZED" vectorType="YES"
<fieldType fieldName="text" storageType="NO" indexType="TOKENIZED" vectorType="YES"
```



### 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?

Can store either original value of a field or just a pointer in the index

Useful to set the storageType to yes and store the original value if you want to retrieve this value from the index instead of the database



```
<fieldType fieldName="sizerange" storageType="YES" indexType="TOKENIZED" vectorType="YES"
  <Analyzer type="Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer, Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer, Sitecore.ContentSearch.LuceneProvider.Analyzers.LowerCaseKeywordAnalyzer"
</fieldType>
<fieldType fieldName="title" storageType="NO" indexType="TOKENIZED" vectorType="YES"
<fieldType fieldName="text" storageType="NO" indexType="TOKENIZED" vectorType="YES"
```

MyResultItem (from our example) class is *not* populated unless stored as a value



## Computed Fields

Simple indexing is when a field value put into index without any processing

Computed fields perform lookups and complex logic to determine what gets put into the index and they do not have to match up to a Sitecore field

### Example: Image size

You want to facet a list of articles by associated image size with potentially hundreds of dimensions

Computed fields categorises images into *small*, *medium*, or *large* depending on specific boundaries and these are the values put into the index

Faceting on 'image size' gives a limited number of results

To create your own computed field, add an entry to `<fields hint="raw:AddComputedIndexField" />` in the search configuration files

Your computed field class must implement `IComputedIndexField`



### 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:

- Working with Media
- Caching and Performance
- Publishing
- Installing and Scaling Sitecore
- Team Development and the Development Environment
- How to Deal with Deployment
- Basic Security
- Workflow

## 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

**Media items are stored in the database**

Media items are **ordinary items**

Located in the **content tree**

Based on a **template** which be extended


**Media in database means...**

**No reliance on files** when moving or syncing the environment – e.g. between developers, QA, production etc.

**Note:**

Media items are **cached to the file system** when rendered

Item needs to be **published**



The screenshot shows the Sitecore Content Editor interface for a media item named 'bike-grayhound'. The item is of type 'Media' and is based on a template. A blue callout box points to the 'Media' type, stating 'Template specific to media type, e.g. image, docx, pdf etc'. Below the item details, there is a preview of a bicycle image and a list of properties including 'File Path', 'Image', 'Alt', 'Width', and 'Height'.



#### 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

How does Sitecore know which template to apply when a new file gets uploaded?

Web.Config

```
<mediaType name="JPEG image" extensions="jpg, jpeg">
  <mimeType>image/jpeg</mimeType>
  <forceDownload>false</forceDownload>
  <sharedTemplate>system/media/unversioned/jpeg</sharedTemplate>
  <versionedTemplate>system/media/versioned/jpeg</versionedTemplate>
  <mediaValidator type="Sitecore.Resources.Media.ImageValidator" />
  <thumbnails>
    <generator type="Sitecore.Resources.Media.ImageThumbnailGenerator, Sitecore.Kernel">
      <extension>png</extension>
    </generator>
    <width>150</width>
    <height>150</height>
    <backgroundColor>#FFFFFF</backgroundColor>
  </thumbnails>
  <prototypes>
    <media type="Sitecore.Resources.Media.JpegMedia, Sitecore.Kernel" />
  </prototypes>
</mediaType>
```

Specify which template to use


Types and properties can be changed and overridden

Media files that have not been mapped use the Any mediaType setting

## File Extensions

By default media is presented as .ashx URLs

To render media links with the real file ending, change the following property in the Web.Config :



```
<!-- MEDIA - REQUEST EXTENSION
  The extension to use in media request URLs.
  If the value is not set, the Extension field of the individual media items
  will be used (ie. JPG, GIF, etc.)
  Default value: "ashx"
-->
<setting name="Media.RequestExtension" value="ashx" />
```

**Note:** If you are using IIS7+ Integrated mode and ASP.NET Managed Pipeline, the file extension is irrelevant

## 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



When returned from the database - each caching level gets regenerated with a more complex object

Defined in the `<databases>` section in web.config and Prefetch .config files

There are more specialized caches for XSL, standard values, paths, registry etc.

#### HTML Cache

Page assembled with presentation components

### How to cache a component?

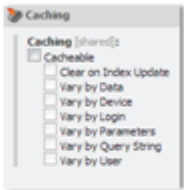

Per instance of the component

Standard values

Definition item

Vary by allows you to store a different version of the component based on e.g. user, datasource etc.

**Note:** Dynamically generated content in a component cannot be cached



## 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

Any form of publishing clears the entire HTML cache

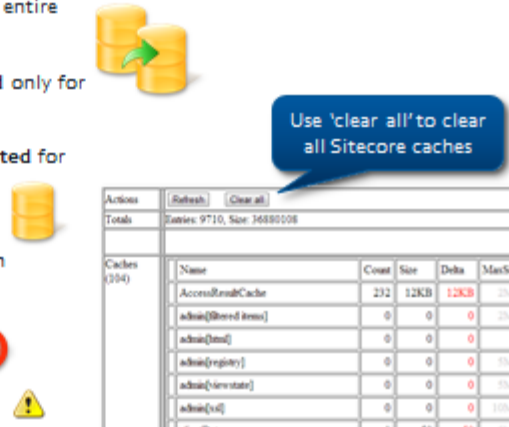
Item and Data caches are flushed only for the items published

When published item gets requested for the 1<sup>st</sup> time, it comes from the database

Prefetch cache is populated when application starts

Cache admin page:  
/sitecore/admin/cache.aspx

HtmlCacheClearer in web.config



Actions		Refresh	Clear all		
Totals					
Entries: 9710, Size: 36880008					
Caches (3/4)					
Name	Count	Size	Delta	MaxSize	
AccessAndCache	232	12KB	12KB	2500	
admin(Shared Items)	0	0	0	2500	
admin(email)	0	0	0	0	
admin(registry)	0	0	0	5000	
admin(serystate)	0	0	0	5000	
admin(ui)	0	0	0	10000	
feedData	1	80	80	1000	

## 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

Cache settings are defined in multiple places in the `web.config`

Default setting per cache eg. Standard Value cache


```
<!-- STANDARD VALUES CACHE SIZE
The default size of the standard value cache.
Default value: 1MB.
-->
<setting name="Caching.StandardValues.DefaultCacheSize" value="1MB" />
```

Cache sizes are per database and per site

You can disable all cache limits by setting the following value to true:

```
<setting name="Caching.DisableCacheSizeLimits" value="false" />
```

Items to be included in the prefetch cache are defined in database-specific `.config` files in `/App_Config/Prefetch`



### 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>



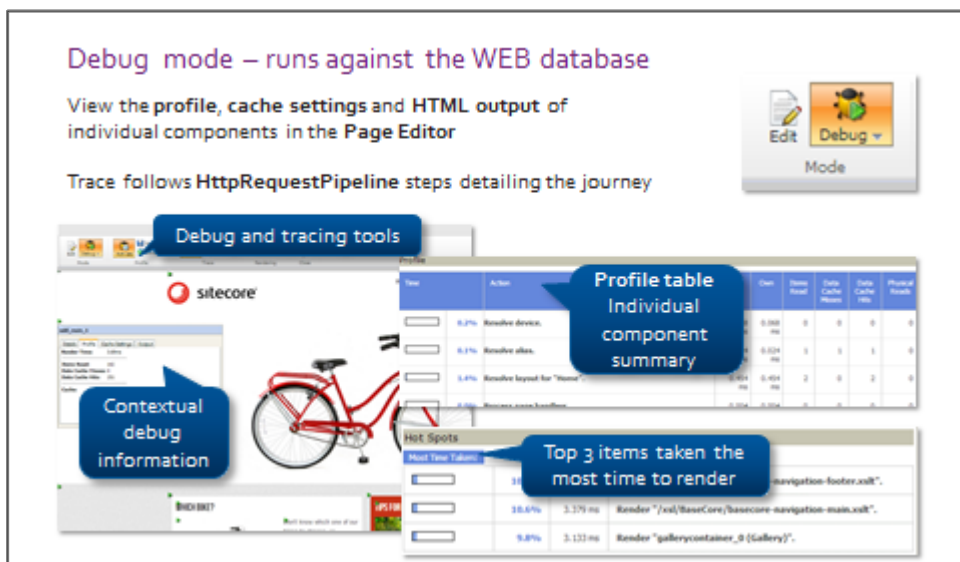
### Demo – Various Cache Settings

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

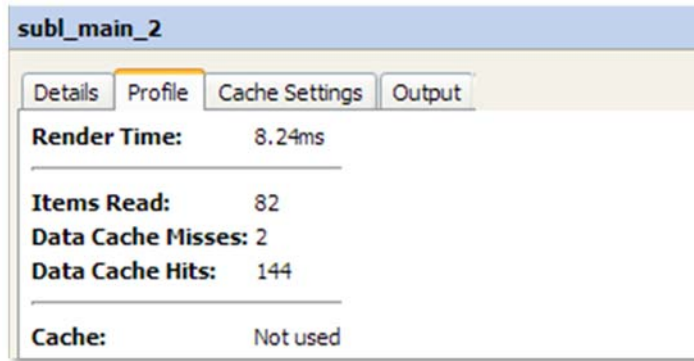
In the following walkthrough, we will:

- 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):



subl_main_2			
Details	Profile	Cache Settings	Output
<b>Render Time:</b>	8.24ms		
<b>Items Read:</b>	82		
<b>Data Cache Misses:</b>	2		
<b>Data Cache Hits:</b>	144		
<b>Cache:</b>	Not used		

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

**3 publishing modes**

**Incremental** – publishes changed items in a publishing queue

**Smart publish** – compares master DB to web DB if the RevisionID is different then the item gets copied across

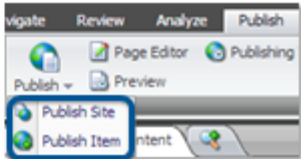
**Republish** – Ignores the publishing queue and copies the items across regardless

Each Sitecore interface has publishing options each invoking the appropriate publishing wizard

**Desktop** – publish site option – incremental, smart and republish wizard

**Page Editor** – publish item option - smart and republish (no incremental....why?)

**Content Editor** – publish site and item option – depending on whether you are publishing one item or the whole site the appropriate wizard will be available



#### Publishing Restrictions

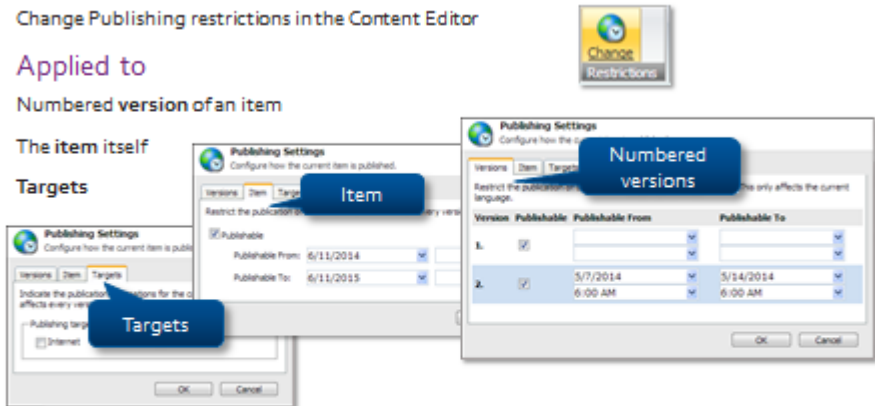
Change Publishing restrictions in the Content Editor

**Applied to**

Numbered version of an item

The item itself

**Targets**



**Note:** This does not automatically publish your items, but simply sets restrictions, they still have to be published manually / scheduled tasks

## Publishing Strategies

**Publishing by editors**

Disable publishing for editors by making everything part of an \*approval process

Full re-publish can slow performance

\* Workflows covered later




**Options for scheduled publishing**

Publishing agent allows you to specify publishing interval and type of publishing

Dependent on the Sitecore scheduling agent

Windows scheduled tasks can be used to run a custom service or page

See blog for more information: <https://www.sitecore.net/Learn/Blogs/Technical-Blogs/Reinnovations/Posts/2014/03/Publishing-Improvements-in-7-2.aspx>

### Demo – Publishing Options and Restrictions

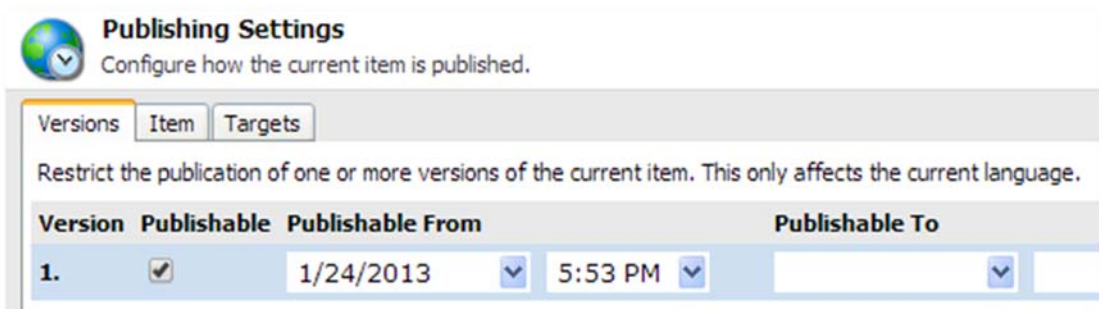
In the following walkthrough, we will:

- 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**:



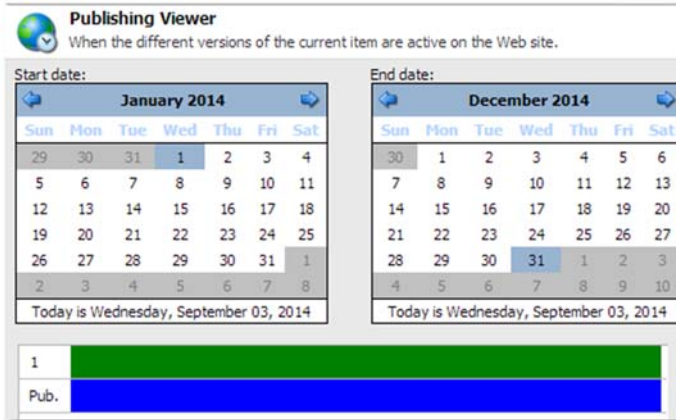
3. Look at the three tabs – Version, Item, and Target:



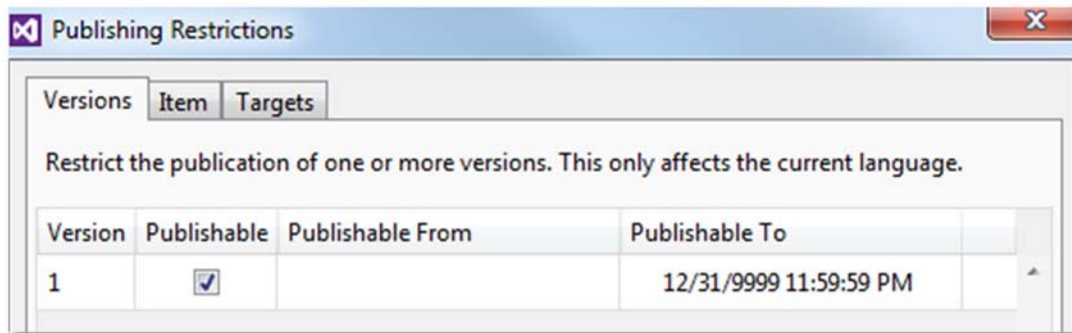
If you clear the **Publishable** checkbox on the **Versions** or **Item** tab, that item or version will be removed when the item is next published.

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.

- 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:



- 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

**In Sitecore rocks:**  
 Perform a whole site or single item publish

View the publishing queue

Perform publishing and view the publishing queue

Publishing queue

Name	Path
which-bike	/sitecore/content/sitecore-cycling-holidays/Home/which-bike
about-us	/sitecore/content/sitecore-cycling-holidays/Home/about-us
bikes-for-racing	/sitecore/content/sitecore-cycling-holidays/Home/which-bike
can-i-bing-my-own-bike	/sitecore/content/sitecore-cycling-holidays/Home/which-bike
Home	/sitecore/content/sitecore-cycling-holidays/Home
main	/sitecore/layout/Placeholder Settings/BaseCore/main

**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](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)

**Quick Sitecore installation from local repository**

**Including packages**

Rocks allows you to pick and choose which steps to execute

**SIM can be customized and extended**

SIM gives you the option to reinstall an instance using the same configuration

**Both allow you to remove Sitecore instances**

The top screenshot shows a list of 12 installation steps for Sitecore Rocks, such as 'Create Project Folder', 'Copy Web Site Files', and 'Copy Database Files'. A blue callout bubble labeled 'Sitecore Rocks' points to the list.

The bottom screenshot shows the Sitecore Instance Manager (SIM) interface. It has a menu with 'Home', 'Browse', 'Edit', 'Maintain', and 'Tools'. Below the menu are buttons for 'Refresh', 'Install Instance', 'Install Modules', 'Download Sitecores', and 'Settings'. A list of instances is shown below, including 'CAU', 'Traincore', and 'TrainCore-Corporate'. A blue callout bubble labeled 'SIM (Sitecore Instance Manager)' points to the interface.

- **SIM download and documentation:**  
[http://marketplace.sitecore.net/en/Modules/Sitecore\\_Instance\\_Manager.aspx](http://marketplace.sitecore.net/en/Modules/Sitecore_Instance_Manager.aspx)
- **Sitecore Rocks:**  
[http://marketplace.sitecore.net/Modules/Sitecore\\_Rocks.aspx](http://marketplace.sitecore.net/Modules/Sitecore_Rocks.aspx)

## Sitecore’s Include Folder

**Smaller configuration files in App\_Config/Include merge into web.config**

Use several .config files for pipelines, events, data folder etc.

Configure where they merge into the web.config

**Version control** the smaller include files so Sitecore upgrades etc. are easier to implement

/sitecore/admin/showconfig.aspx shows you the merged web.config within the sitecore section

Only works for the <sitecore> portion of the web.config

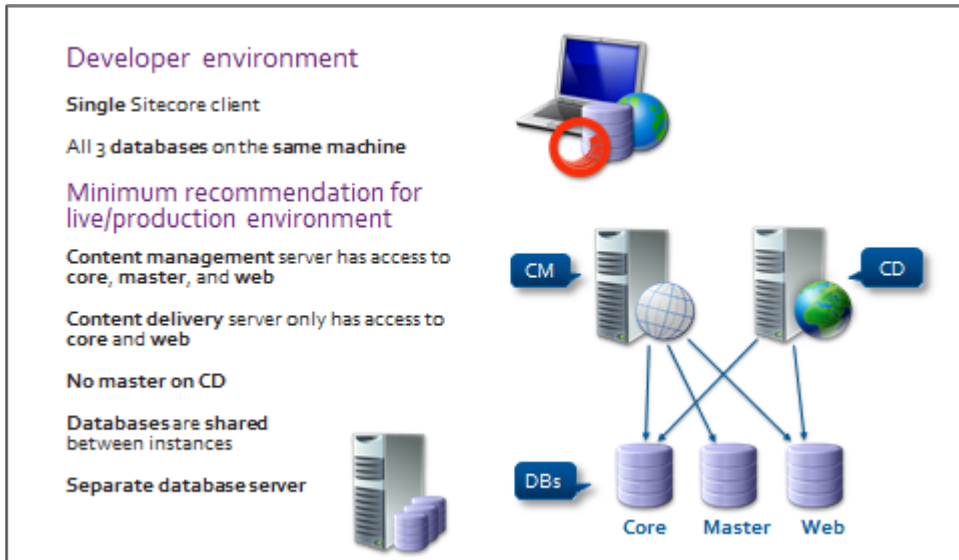
**Benefits**

Changes are isolated and organised

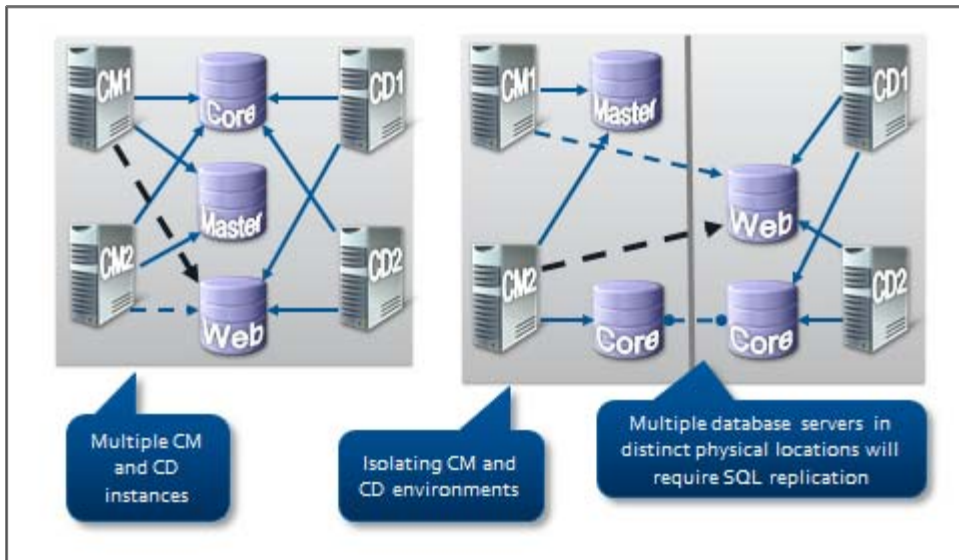
Default web.config stays untouched

The illustration shows a yellow folder labeled 'include folder'. Inside the folder are two white cards labeled 'pipelines' and 'events'.

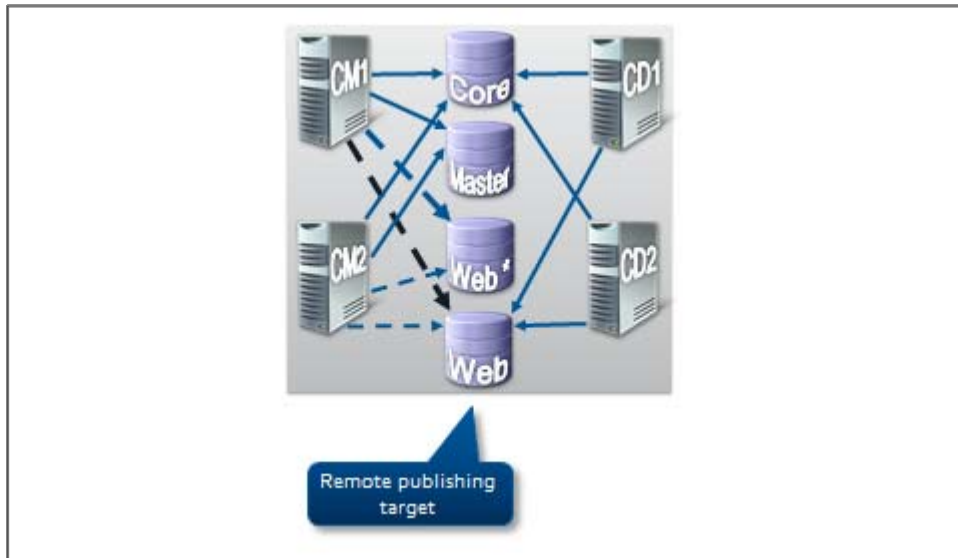
### Scaling Infrastructure



### 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
<p><b>Inside the web root:</b></p> 	<ul style="list-style-type: none"> <li>• Solution setup is quick and straightforward</li> <li>• If your team is using the code-beside development model, you will be able to see your changes almost immediately</li> </ul>	<ul style="list-style-type: none"> <li>• No clear separation of ownership</li> <li>• Difficult to transform configuration files</li> <li>• Development sites may contain code-beside files or other unnecessary artifacts that may alter site behavior inadvertently</li> </ul>
<p><b>Outside the web root:</b></p> 	<ul style="list-style-type: none"> <li>• Clear separation between the solution files and Sitecore-owned files</li> <li>• Ability to transform configuration files within local developer environments without changing the default config files</li> <li>• Better suited for multi-site solutions</li> <li>• Solutions can be bundled easily for shipment</li> <li>• Development sites remain clear of code-behind files</li> </ul>	<ul style="list-style-type: none"> <li>• 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</li> <li>• Initial solution setup can be complex as you create and employ custom post-build actions</li> </ul>

## Source Control and Serialization Options

Working outside the web root makes it easier to source control your custom code

Sitecore items can be serialized into text files then added to source control

Serialization makes it easier to synchronize changes across local development environments

Methods of serializing:

Post-save macro in Sitecore Rocks

Serialization page - /sitecore/admin/serialization.aspx

Serialization settings in the web.config

Third party products like Hedgehog's Team Development for Sitecore



### Important

Serialization may not be appropriate when deploying changes to a staging or production environment because it overwrites items.

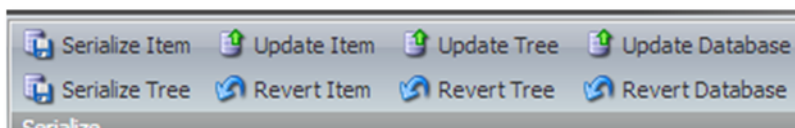


### Walkthrough – Serializing Items

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:



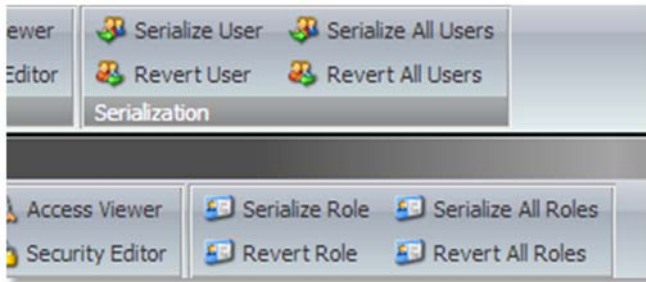
4. Using Sitecore Rocks, demonstrate that items can be serialized by right-clicking and selecting **Tools > Serialization**:





**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**

**Sitecore packages**

Packages allow you to patch in changes

.zip files containing Sitecore items and code files (⚠ security accounts)

Most modules are Sitecore packages

Manually select items or dynamically using a query in Package Designer in the Content Editor

When installing you get a prompt to merge with options or overwrite

**Sitecore Rocks**

**Anti Package**

Allows you to set up dependencies



**Tip**

See the *Package Designer Administration Guide* if you are using the Package Designer interface on the Sitecore Desktop:

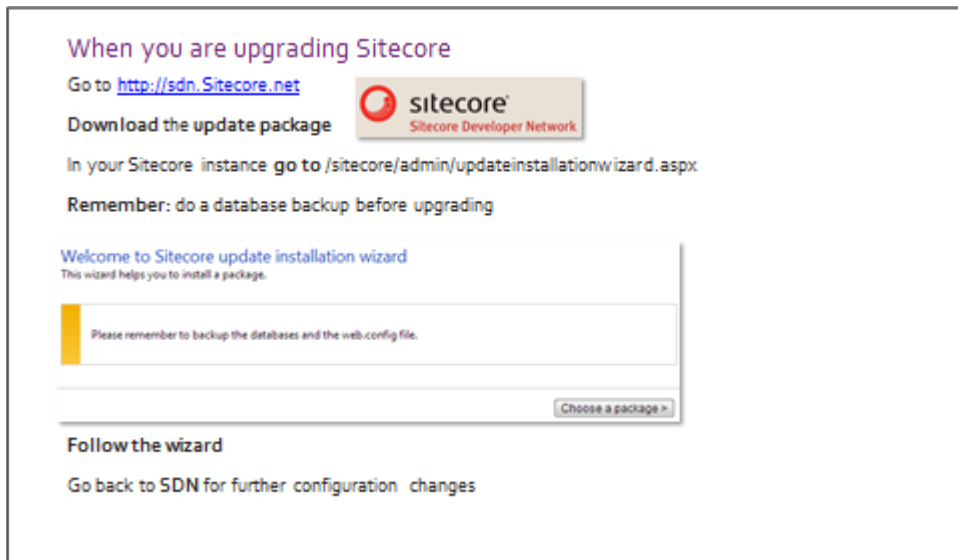
<http://sdn.sitecore.net/Reference/Sitecore%207/Package%20Designer%20Administrator%20Guide.aspx>



**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:

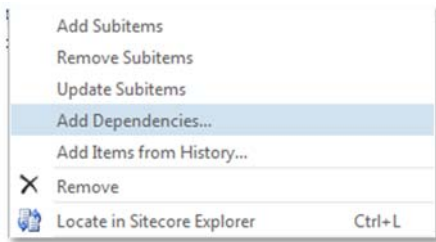
- Create a package using Sitecore Rocks.

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.

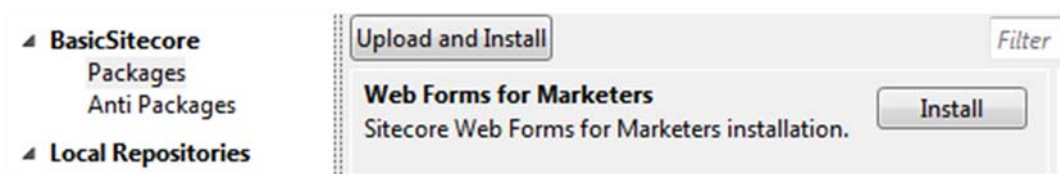
**Note:** you cannot drag and drop files from your solution, because solution files are independent of the Sitecore instance.



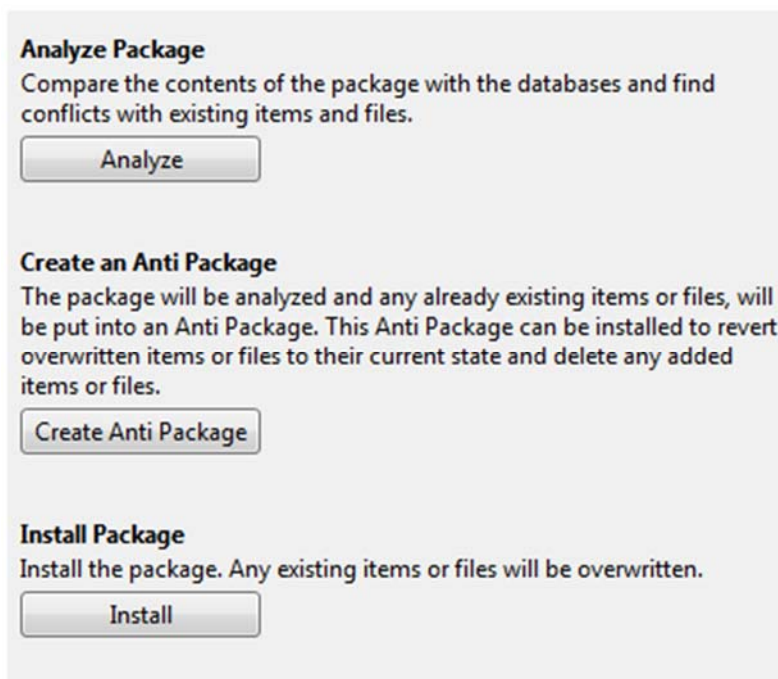
6. Notice that right-clicking on an item gives you options to **add**, **remove**, and **update** its children:



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

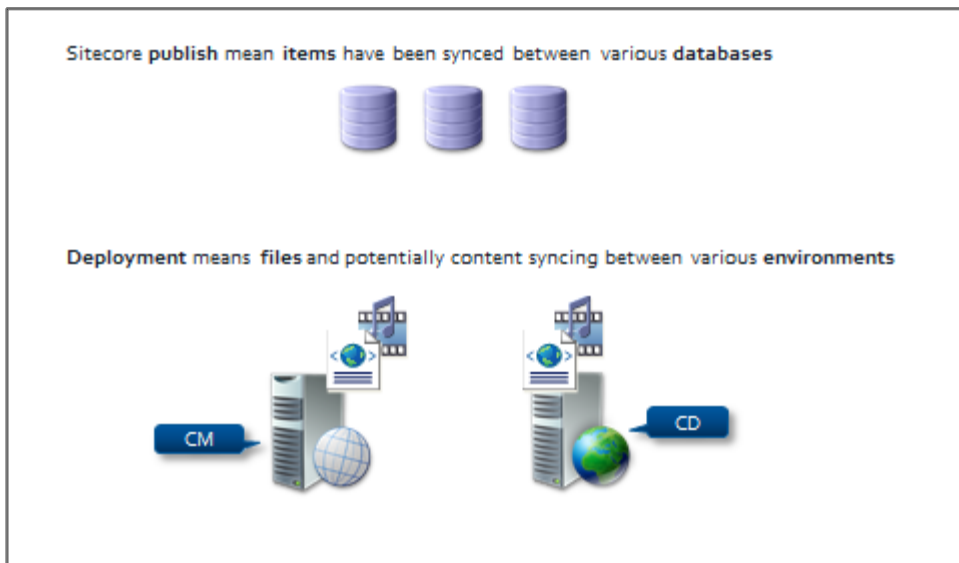
#### Objectives

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

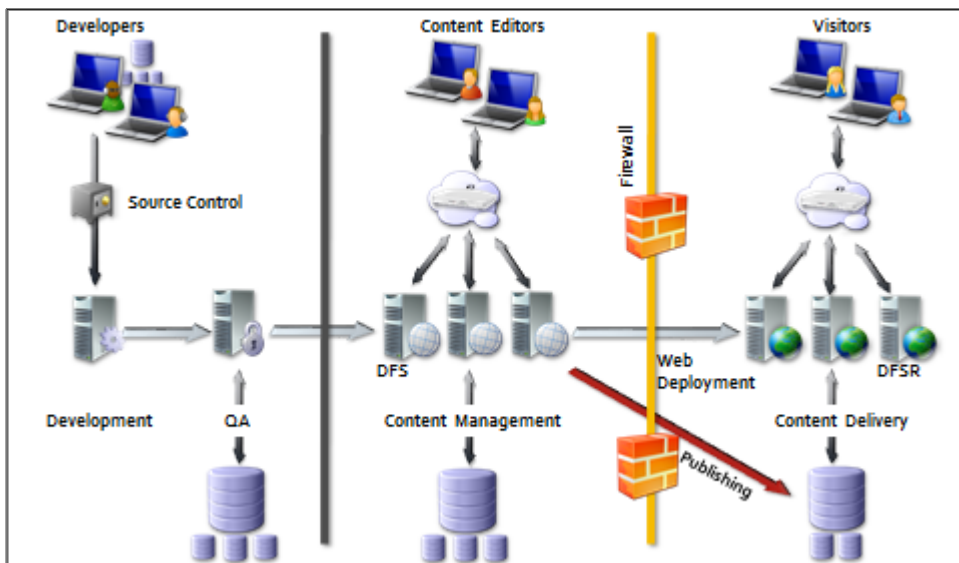
- State the difference between publishing and deployment.
- List some points to remember when deploying to an authoring environment.
- Discuss options for reverse content promotion.

### Content

#### Difference Between Publishing and Deployment



#### Deployment Between Environments



## Points to Remember

### When deploying to an authoring environment

You are changing the **authoring environment**, therefore you need to be careful what items you are overriding with your own

Authors can keep updating - schedule a **content freeze**

**Back up** database / file system

Potentially deploy to a **temporary installation** then switch when authors are done editing (e.g. pre-live.mysite.com and mysite.com)

### Reverse content promotion

Update your developer environment with the content managed content

Database backups

Packages



## Topic 9.7 Basic Security

### Introduction

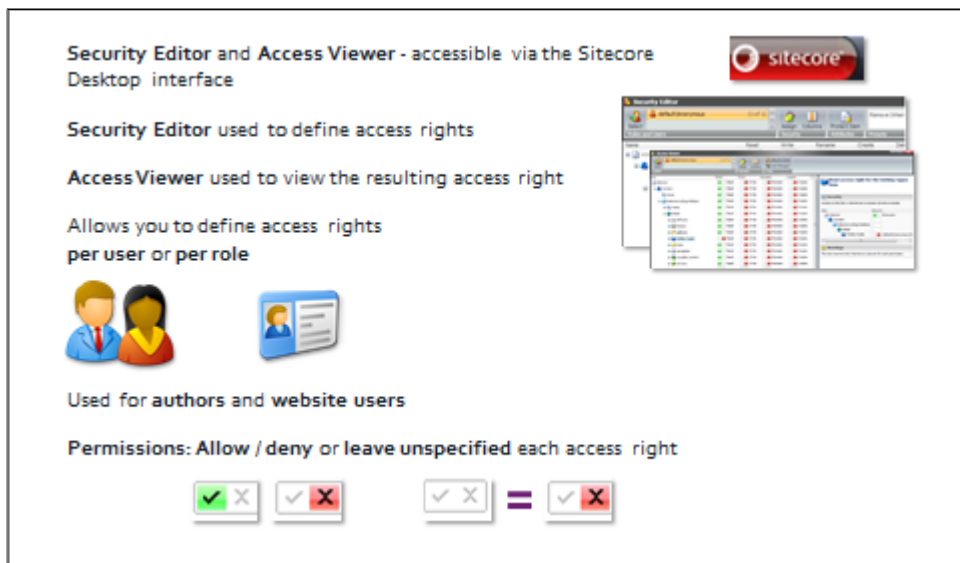
#### Objectives

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



Security Editor and Access Viewer - accessible via the Sitecore Desktop interface

Security Editor used to define access rights

Access Viewer used to view the resulting access right

Allows you to define access rights per user or per role

Used for authors and website users

Permissions: Allow / deny or leave unspecified each access right

Icons:       =



#### 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](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

**Role inheritance**  
Access rights are **cumulative** - a user inherits all access rights from all roles

**Conflicting access rights**  
Explicit **deny** will always take precedence  
Can be overridden at user level

**Client**  
Sitecore comes with **pre-defined roles**  
Restricts what actions an editor can perform e.g. **Sitecore Client Publishing** – allows user to publish

Write access to an item	
Role A	Role B
<input checked="" type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
	=
<input checked="" type="checkbox"/> <input type="checkbox"/>	<input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>



### 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.

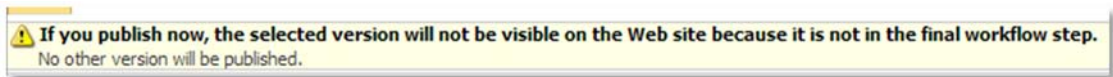


#### Demo – Using a Workflow as an Author

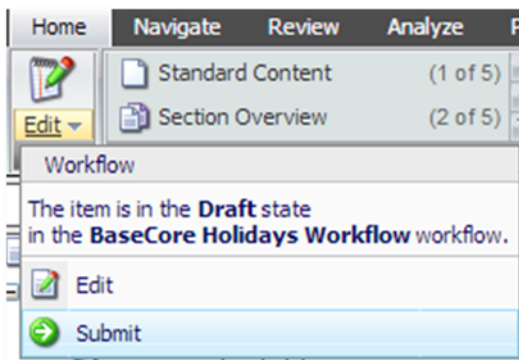
In the following demo, we will:

- 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:

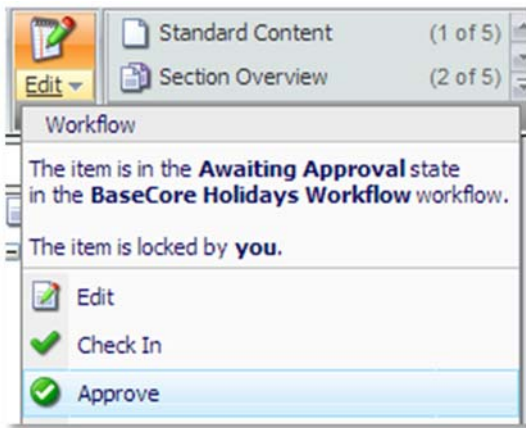


3. Click the arrow under the **Edit** button on the ribbon, and select **Submit**:



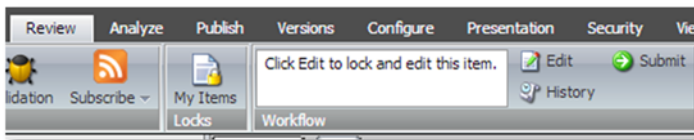
4. Enter a comment when prompted.

- 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.



**i** 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?

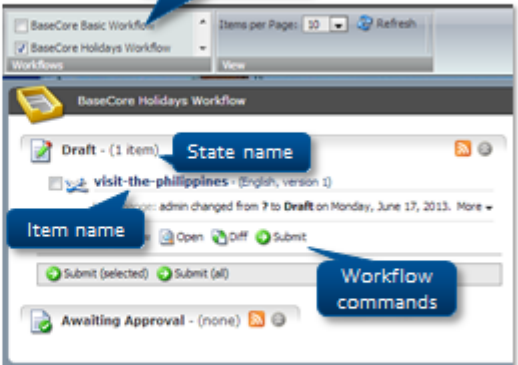
**A process for content approval**

Content needs to be **approved** by different members of the organization

**Consists of:**

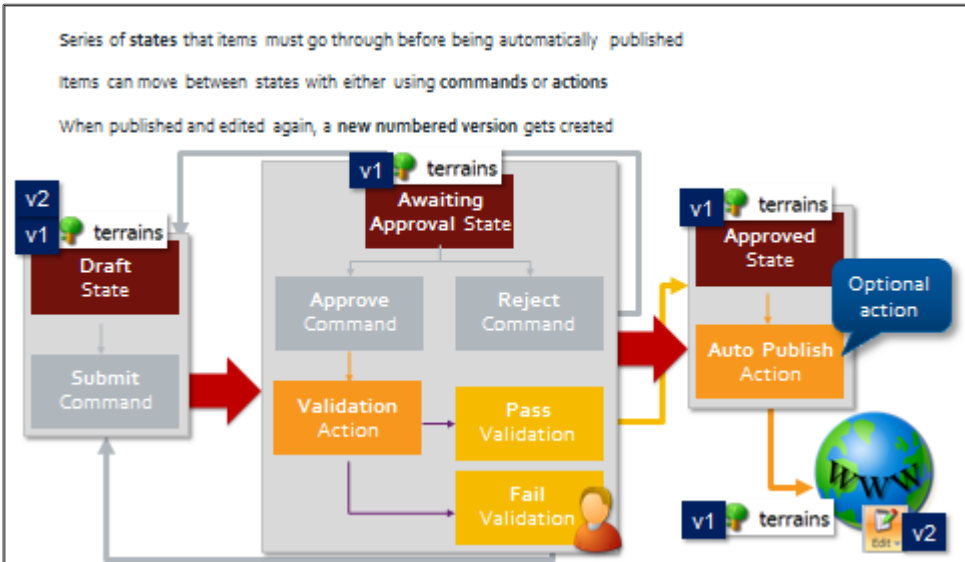
- States that an item moves through
- Each state can have **commands** and **actions**
- Items are **publishable** when they are in a **final state** (can have multiple)

Can be used for content versioning



The screenshot shows the Sitecore workflow management interface. At the top, there are two workflow lists: 'BaseCore Basic Workflow' and 'BaseCore Holidays Workflow'. The 'BaseCore Holidays Workflow' is selected and expanded. It shows a 'Draft' state with one item, 'visit-the-philippines - (English, version 1)'. Below the item, there are buttons for 'Open', 'Diff', and 'Submit'. At the bottom, there are buttons for 'Submit (selected)' and 'Submit (all)'. Callouts point to the 'Workflow name', 'State name', 'Item name', and 'Workflow commands'.

### 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 you 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

**Measure, Personalize and Optimize**

Measure the effectiveness of your site using goals to assign an engagement value to each visit

Personalize content based on user's activity on the site

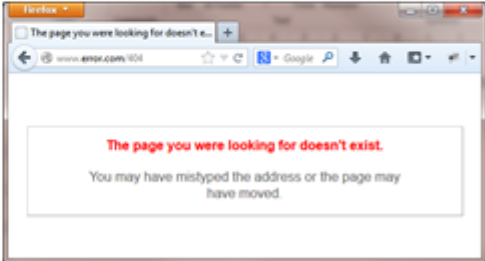
Optimize your website using tests on discreet site functionality

**Track and Report**

Lead tracking

Granular and overview reporting

Site issue reporting (e.g., 404, slow pages)



#### Building for the XP


**How you design and build your site directly affects how effectively a marketer can implement and use the XP's marketing functionality:**

Componentize your pages in order to be able to test and vary particular areas of your pages on the fly

Use datasources to be able to vary the content of your components on the fly

If you want to vary content, it has to be defined on an item – no hard-coding

Most of the work building for the Page Editor is relevant to building for marketing functionality



## I'm Not Using It yet

Even if you are 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


Traffic figures (number of hits) are not a true representation of how much a visitor engaged with your site – e.g., *visiting the homepage (low engagement) versus booking a holiday (high engagement)*

A site that encourages engagement is more effective – e.g., *10,000 visits and 10 bookings versus 1000 visits and 100 bookings*

Sitecore introduces concept of **engagement value** - a number that represents how much quality time a user spent on your website



The more actions a customer performs on your site that have **value** (e.g., request a brochure, fill in the contact form, make booking), the higher their engagement value


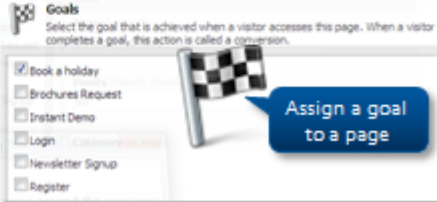


#### Goals

Goals are defined in the Marketing Center and have a **value** associated with them

A goal's value depends on its **value to the business** – a holiday booking is worth more to the business than a brochure request, for example

Goals are either assigned to **pages** (e.g., the contact form thank-you page) **or** **Triggered programmatically** – when a visitor triggers a goal, the engagement value for that visit increases



**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**

**Executive Insight Dashboard**

Overview statistics for management

Average engagement value per visit

Define time period

Average value per visit per entry page, search term, source, and entry page

**Engagement Analytics (real time)**

Granular statistics – view top leads or drill down to particular visits

Customizable reports

Look at path taken through the site, and goals and events triggered

## 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 MinimumVisitsFilter (set to 50 by default) that was set in:

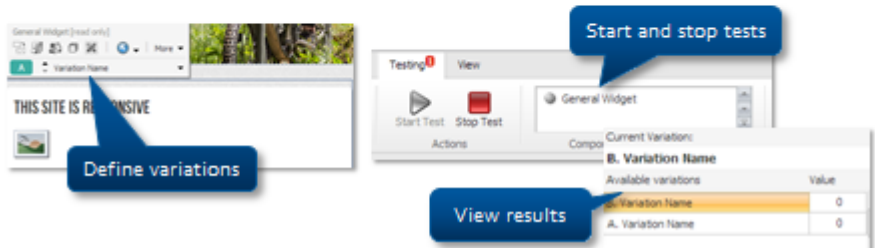
[\sitecore\shell\Applications\Reports\Dashboard\Configuration.config](#)

## Testing

Allows you to test the **effectiveness** of your site by seeing which variation resulted in greater **engagement value** – for example, which homepage offer text resulted in the greatest number of bookings (*note: we are not measuring clicks!*)

Multivariate testing is done at the **component level** – you can change the **component**, the **datasource** or **both**

Tests should be **granular** and test **one thing at a time** – otherwise you cannot even begin to speculate on what resulted in increased engagement



Current Variations	
B. Variation Name	
Available variations	Value
B. Variation Name	0
A. Variation Name	0



## 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.*

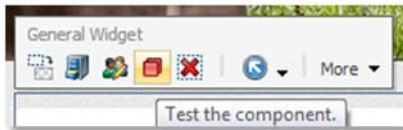


## Demo – Setting Up a Multivariate Test

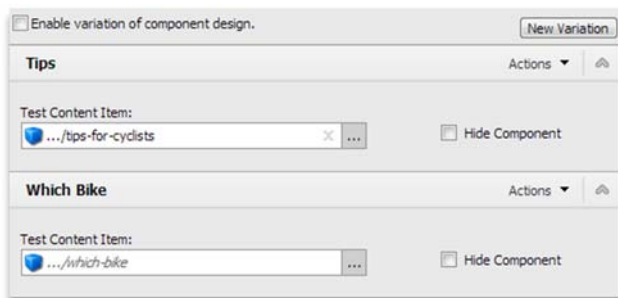
In the following demo, we will:

- 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.

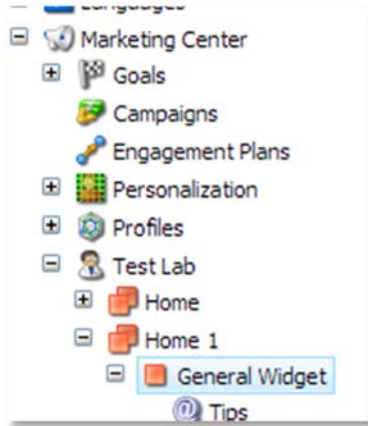


3. Add a new variation (It will default to what is already selected). Then add a second variation and only change the datasource of the component.

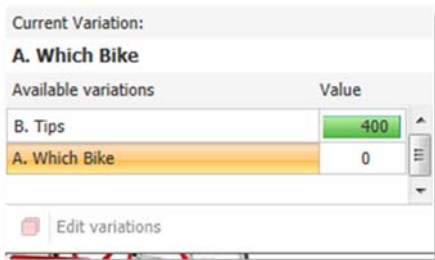




4. Notice that you can scroll between the variations.



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

Content and component type can be personalized to suit the visitor

Change component's datasource, change component type, or hide component.

You can personalize based on goals and events triggered, the profile points a visitor has accrued (next slide), their location, and more.



Very important that you have **componentized** and made good use of **data sources** – otherwise personalization will not work.

Rule determining when to change data source



#### Best Practice

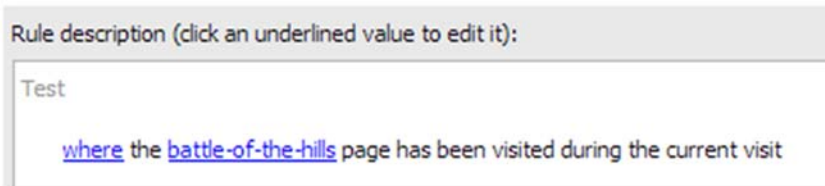
Encourage authors to make content profiling a *routine task* or risk ending up with 40,000 items that require profiling. Make sure that a meaningful profile taxonomy has been created. Defining dimensions, keys, profile cards and pattern cards is a *business decision not a development decision*.

## 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

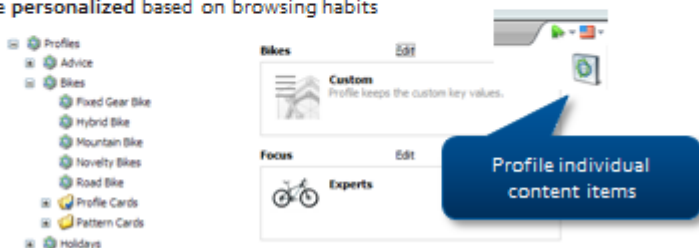
**Score items based on the audience you are targeting**

Business stakeholders design **profiling taxonomy** (configuration done by developers)

**Authors** assign points to items in different categories based on audience being targeted (or bulk-assign using **Profile Cards**)

**Visitors** accrue points based on the content they visit and build up a profile

Content can be **personalized** based on browsing habits



## Demo – Profile Content

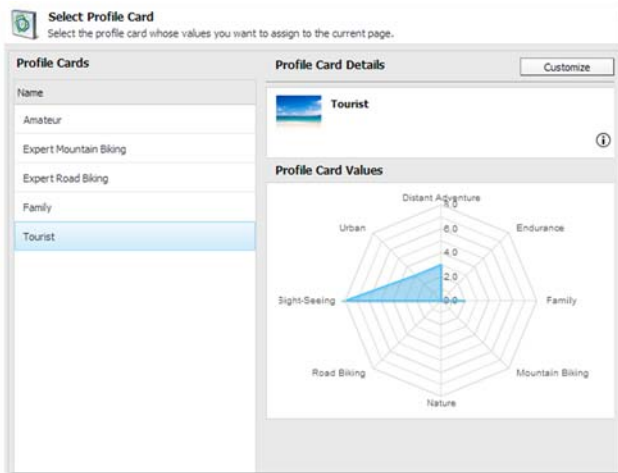
In the following demo, we will:

- 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.

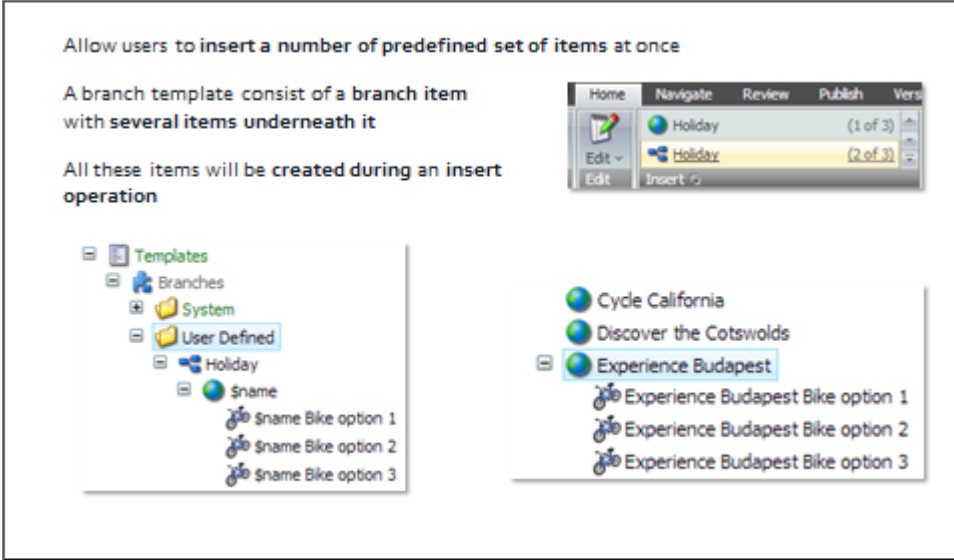
### Content

#### Branch Templates

Allow users to insert a number of predefined set of items at once

A branch template consist of a branch item with several items underneath it

All these items will be created during an insert operation

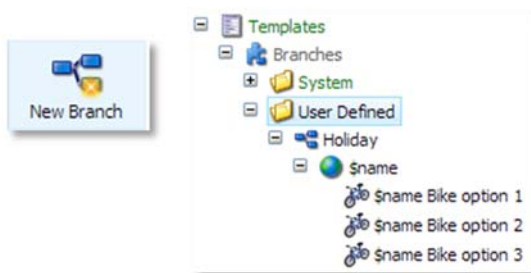


#### Walkthrough: Creating a Branch Template

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 exist, 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`.



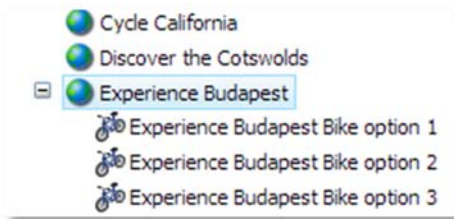


## Knowledge Check

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

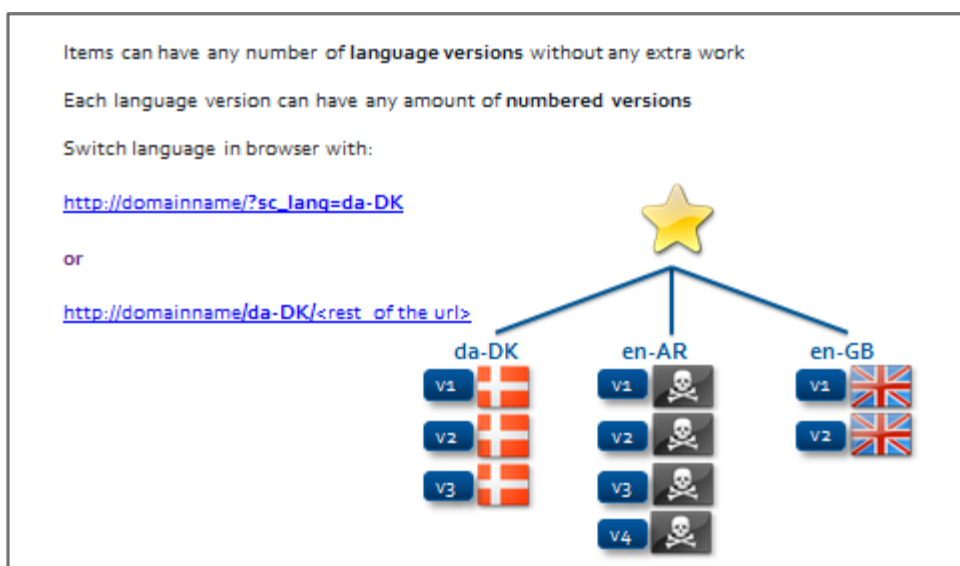
#### Objectives

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.

- In the **Trip Details data template**, create a new field called **Country**, which is **Unversioned** (single value per language).



- Enter a default value for the **Country** field.
- 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.
- Fill in the **Danish field** with **USA**.
- 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

**Defining fields on the data template**

- Unversioned field – single value per language**
- Shared field – single value for all numbered versions in all languages**
- Versioned field (default) – different values for languages and numbered versions**

**i** **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

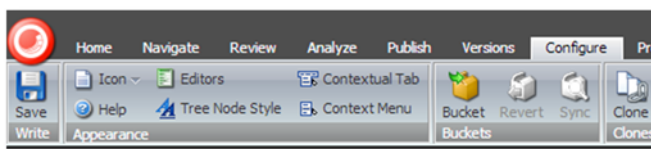
In this demo you will see that fields can have helper text added to them to help authors fill in fields.

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 Content 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.

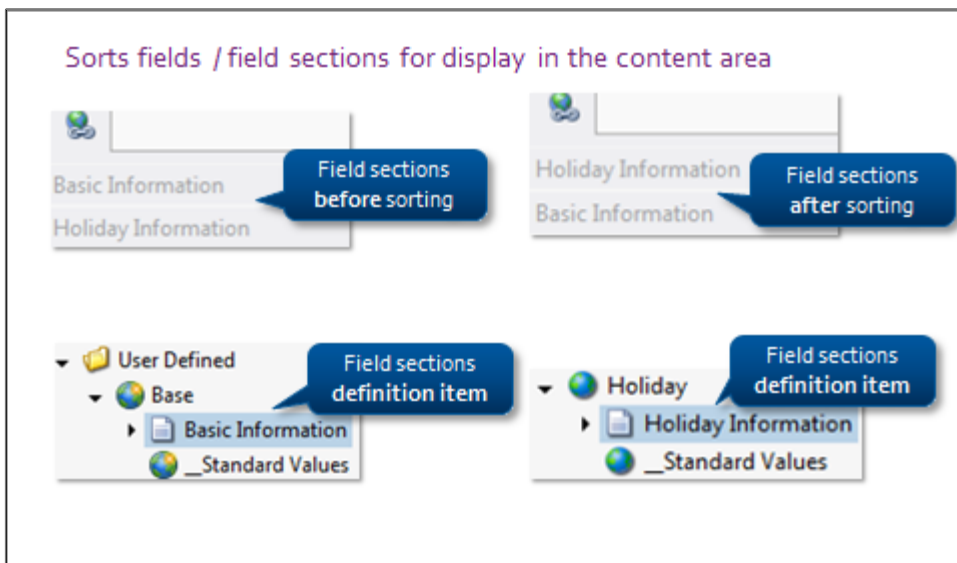


## Title Field

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

To view the `__Sortorder` field ensure the standard fields is ticked

In the Content Editor in View tab

In Sitecore Rocks right click in the Editor Pane

\_\_Sortorder  
20

\_\_Sortorder  
0

### Help Text

Add helper text to field names for authors

With help text

The helper fields are in the standard fields

Without help text

Standard fields:  
\_\_Long description  
\_\_Short description

### Title Field

Change the field name that is displayed to authors

For developers the field name stays the same

Holiday Admin  
Holiday ID

Field definition item

Data  
Type  
Integer

Title  
Booking reference number

Title field

Holiday Admin  
Booking reference number [shared, standard value]:  
1

What the user sees

**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.

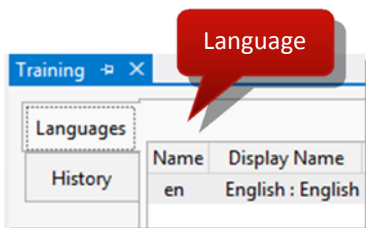
**Apply – Topic 11.2 – 35 min****Working with Versions, Sorting, Help Fields, and Field Source**

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**

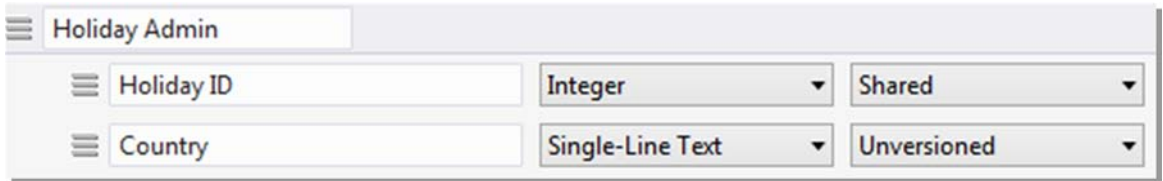
1. Go back to the <http://BasicSitecore/> site you created on day one. Expand the **BasicSitecore** instance node in Sitecore Rocks.
2. Using **Sitecore Rocks**, right-click the **master database** and select **Tools > Language**.
3. Right-click in the content pane and select the **Add New Language...** 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.

- 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.
- 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).



- Enter a default values for the **Holiday ID** and the **Country** fields in the Standard fields.
- 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.
- Fill in the **Danish** field with **USA**.
- Switch to **English v#1** (different value).
- Create **V#2 in Danish** by right-clicking the **Editor** pane and then select the **Versions > Add Version** option.

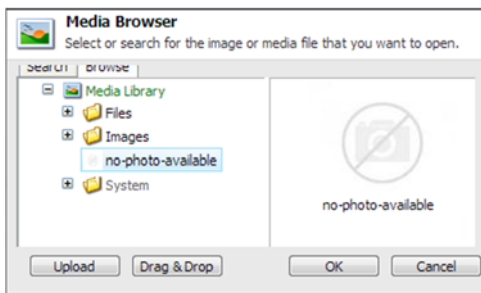


- Switch between **DK v#1** and **DK v#2** (they should be the same value – USA).

**Lab B. Field Source**

In this lab, you will specify a location of where your authors can select images.

- In the **Content Editor**, navigate to a content item – such as **Explore Holland** at: */sitecore/content/Home/Family Holidays/Explore Holland*.
- Change the image and note the browsing location.



- Close that and navigate to the **Explore Holland** data template using the **Quick Info** section.
- Remember **we are inheriting** that image field from the Base data template, so navigate to **Base**.



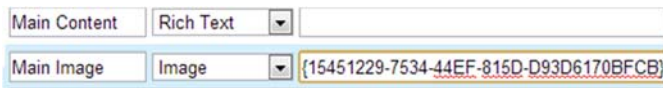
- Change the **field source** to be the **location of a folder in media library**.



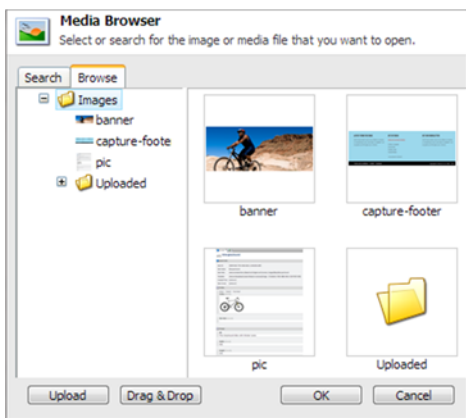
### Tip

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.

- In the source field type `/sitecore/media library/Images` or use the **GUID** of that item.



- 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.

- View the `__Standard Values` item for the **Trip Details** data template.
- Note** the order of the fields `Holiday ID` and `Country` in the **Holiday Admin section**. We are going to swap their order.
- Double-click the **Holiday ID** item at: `/sitecore/templates/User Defined/Trip Details/Holiday Admin/Holiday ID`.
- Right-click inside the **editing** pane to view the **standard fields**.
- Change the `__Sortorder` field to `20`.
- Double-click on the **Type** item at: `/sitecore/templates/User Defined/Trip Details/Holiday Admin/Country`.
- Change the `_Sortorder` field to `200`.
- Right-click again to clear the **Standard Fields** checkbox.
- 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

Pipelines break down complex operations into multiple, configurable steps

Pipelines are defined in /configuration/sitecore/pipelines in the web.config

Pipelines are defined in 2 places 

<processors /> - UI processes (e.g. uiCloneItems, uiDeleteItems)

<pipelines /> - system processes (e.g. httpRequest Pipeline)

One of the most important pipelines is the httpRequest Pipeline, which is invoked by the httpApplication.BeginRequest event

The httpRequest Pipeline performs tasks such as resolving sites and language

Each of the steps is specified as a separate <processor /> within the pipeline

```
<httpRequestBegin>
  <processor type="Sitecore.Pipelines.PreprocessRequest.CheckIgnoreFlag, Sitecore.Kernel" />
  <processor type="Sitecore.Pipelines.HttpRequest.StartMeasurements, Sitecore.Kernel" />
  <processor type="Sitecore.Pipelines.HttpRequest.IgnoreClas, Sitecore.Kernel" />
  <processor type="Sitecore.Pipelines.HttpRequest.SiteResolver, Sitecore.Kernel" />
  <processor type="Sitecore.Pipelines.HttpRequest.UserResolver, Sitecore.Kernel" />
  <processor type="Sitecore.Pipelines.HttpRequest.DatabaseResolver, Sitecore.Kernel" />
  <processor type="Sitecore.Pipelines.HttpRequest.BeginDiagnostics, Sitecore.Kernel" />
</httpRequestBegin>
```

#### Processor

Each step in a pipeline is a <processor /> with a type in the format of Namespace, Assembly

```
<processor type="Sitecore.Pipelines.HttpRequest.ItemResolver, Sitecore.Kernel" />
<processor type="Sitecore.Pipelines.HttpRequest.DeviceSimulatorResolver, Sitecore.Kernel" />
<processor type="Sitecore.Pipelines.HttpRequest.LayoutResolver, Sitecore.Kernel" />
```

Steps are executed in the order specified in the web.config

#### All processors

Implement the Process() method

Accept arguments that are particular to the pipeline they serve

```
namespace Training.Utilities.BaseCore.Pipelines
{
  public class CascadeDataSource : InsertRenderingsProcessor
  {
    public override void Process(InsertRenderingsArgs args)
    {
      // Pipeline code
    }
  }
}
```

**InsertRenderingsProcessor**

## 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 the `Process()` method 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

An event is something that happens that you can respond to

For example an item being saved or a version being added etc.

You can subscribe to events

Events can be extended and overridden with your own event handlers, or you can write your own

Events are defined in /configuration/sitecore/events in the web.config


Each event can have any number of <handler /> nodes that specify a type and method:

```
<event name="item:copied">
  <handler type="Sitecore.Links.ItemEventHandler, Sitecore.Kernel" method="OnItemCopied" />
</event>
```

## Subscribing to Events and Using Event Handlers

Create a class with a method that accepts a sender object and EventArgs

Depending on what event the handler subscribes to, the content of the EventArgs changes – in this example, the EventArgs is an item

 SDN – Using Events

```
public class LayoutInheritance
{
  public void OnItemSaving(object sender, EventArgs args)
  {
    // Event handler code
  }
}
```

Event handler that is executed when an item is saved

Add your handler to the appropriate event, specifying the namespace, assembly, and method:

```
<event name="item:saving">
  <handler type="Sitecore.Tasks.ItemEventHandler, Sitecore.Kernel" method="OnItemSaving" />
</event>
```



### 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

A rule consists of conditions and actions

If a specific condition evaluates to true then a certain action gets triggered – just like an *if* statement

**Rule 1**

where the item name contains Title

and where the item language is equal to DE

set data source to off-the-beaten-path

In Module 10, you used conditional renderings, which utilized the rules engine for personalization - If a user has 5 points, change the datasource of a component

**Technique** Actions ▾

**Condition** Edit  Hide Component

where the value of Technique profile key is greater than 5

**Personalize Content:**

.../expert-holidays

Rules are part of the core Sitecore functionality and can be used everywhere e.g. conditional insert options (if a user is \_\_\_\_, show \_\_\_\_)

#### How Does a Rule Work for Authors?

**Conditions and actions**

User select any number of conditions and actions

Evaluated in succession

If end result is true, actions are executed

**Rule 1**

where the item name contains Title

and where the item language is equal to DE

set data source to off-the-beaten-path

**Rule Set Editor**

Select the conditions and actions first, then specify the values in the description.

Select the conditions for the rule:

- where the page index contains to number
- where the item ID contains to value
- where the level of the item contains to number
- where the item name contains to value
- where the item template is specific template
- where the item language contains to value
- where the item has a layout
- where the parent name contains to value
- where the parent template is specific template
- where the item is in database

Select the actions for the rule:

- hide rendering
- process multi-state text
- set data source to [value]
- set parameters to [value]
- set placeholder to [value]
- set rendering to [value]

Actions:

- run specific script



## 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.

### Rule 1

where the item name is equal to Links  
 add Simple Link insert option

### 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

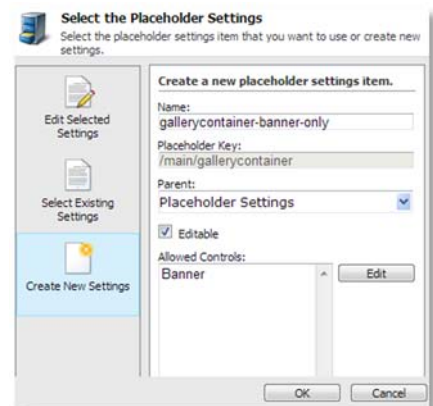


#### Demo: Create Placeholder Setting Override

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.



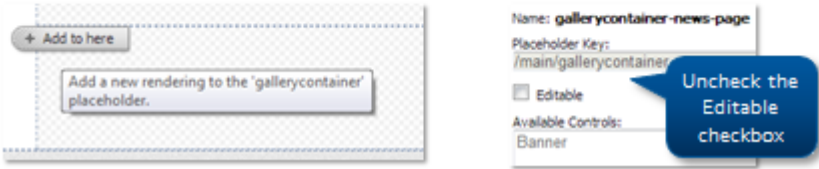
## What are Placeholder Overrides Used For?

You may want to keep the same placeholder on various pages throughout your website, but to allow different controls into that placeholder dependent on which item it's on

Placeholders have an Editable checkbox as do components

Make the placeholder not editable – restricting authors in Page Editor Design mode

For example the Placeholder Setting for the gallerycontainer allows a variety of components to be added – you may want to make that placeholder not editable on the News Article page –placeholder override gallerycontainer-news-page



### **i** 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

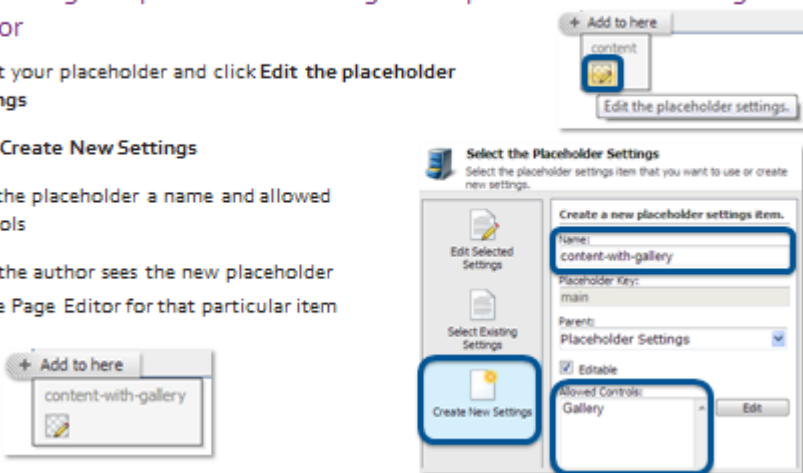
Overriding the placeholder settings for a placeholder in the Page Editor

Select your placeholder and click **Edit the placeholder settings**


Click **Create New Settings**

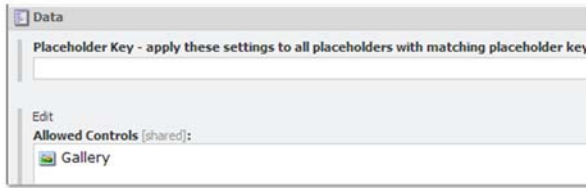
Give the placeholder a name and allowed controls

Now the author sees the new placeholder in the Page Editor for that particular item



## 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

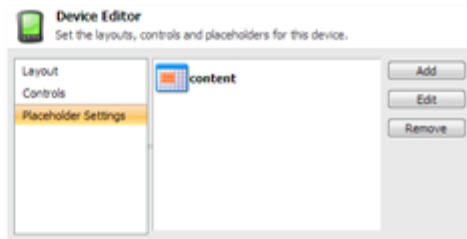
## Where to Override Placeholder Settings

### 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.

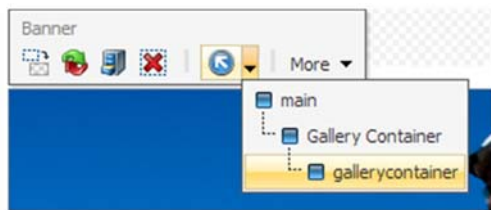
#### A scenario for overriding a placeholder

- 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.

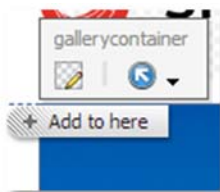
You want to set this restriction for all holiday pages - globally.

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**.

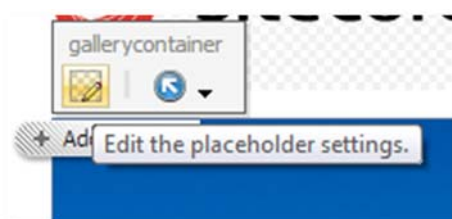
1. Open **Traincore**. In the Page Editor navigate to: */Sitecore-cycling-holidays/Home/holidays/battle-of-the-hills*
2. Make sure you are in **Design** mode.
3. On the holiday page, click the **Banner** component, and navigate to the **gallerycontainer** placeholder using the arrow icon.



4. Click **Add to here**.

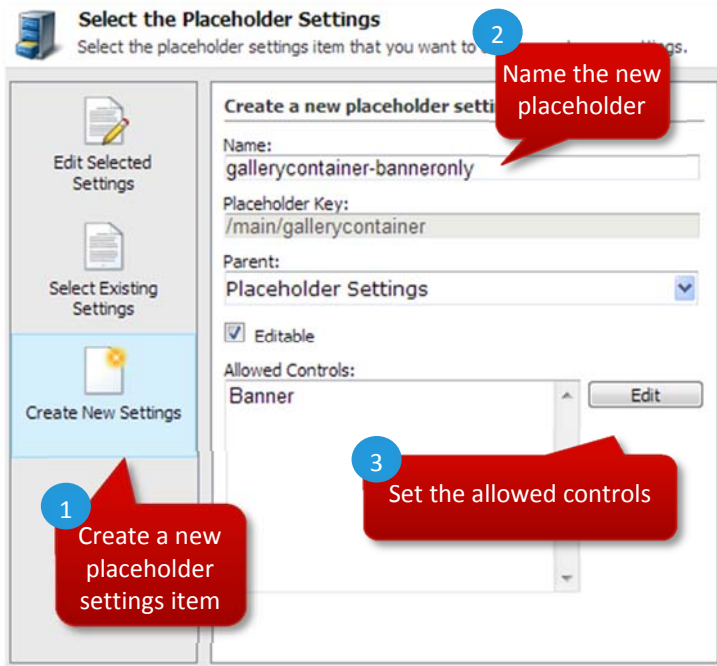


5. As an author you are allowed to add only a Gallery or a Banner component into this placeholder – click **cancel** – think of it as Insert Options for a placeholder.
6. Select the **gallerycontainer** placeholder again and, this time, click the command on the **ribbon** to **Edit the placeholder settings**.

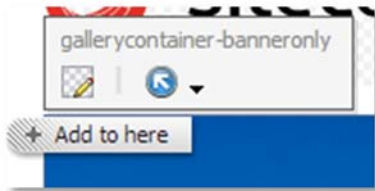


7. Click the **Create New Settings** tab.

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.

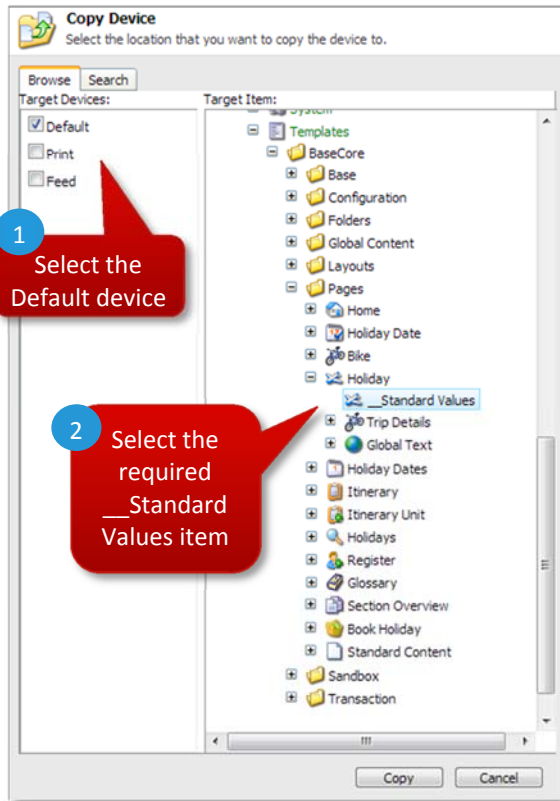
**Lab B. Copy the Placeholder Override to the Standard Values**

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*.

- Click **Copy** then **OK** in the **Layout Details** dialog.



- If you add a new holiday page, it will automatically get the **gallerycontainer-banneronly** placeholder.

**i** 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
- Module 2 – Defining Data
- Module 3 – Presentation
- Module 4 – Sitecore API
- Module 5 – Advanced Presentation Concepts
- Module 6 – Real World Solutions
- Module 7 – Configuring the Page Editor
- Module 8 – Dealing with Your Data
- Module 9 – Recommended Practices
- Module 10 – Marketing Functionality
- Module 11 – Optional Topics

## Module 1 Sitecore Overview

### Topic 1.2 What Is Sitecore

- |  |  |
|--|--|
| <p>Q Name two key components that make up the Sitecore XP:</p> <ul style="list-style-type: none"><li>✓ Experience CMS and marketing platform</li></ul>                           | <p>Q What should you do before you contact support?</p> <ul style="list-style-type: none"><li>✓ Refer to <i>HelpDesk best practices</i></li></ul>                                      |
| <p>Q Name three sites that you can refer to for more information and module downloads.</p> <ul style="list-style-type: none"><li>✓ SDN, Knowledgebase, and Marketplace</li></ul> | <p>Q What are your responsibilities as a developer?</p> <ul style="list-style-type: none"><li>✓ Build according to recommended practices</li><li>✓ Don't fight the framework</li></ul> |

### Topic 1.3 Sitecore from an Author's Perspective

- |   |   |
|---|---|
| <p>Q Name 3 Sitecore interfaces:</p> <ul style="list-style-type: none"><li>✓ Desktop, Content Editor, Page Editor</li></ul>   | <p>Q Name the process by which items are synchronized between the master and web database:</p> <ul style="list-style-type: none"><li>✓ Publishing</li></ul> |
| <p>Q Everything in Sitecore is an...</p> <ul style="list-style-type: none"><li>✓ Item</li></ul>                               | <p>Q Which Page Editor mode would I use to add a new component?</p> <ul style="list-style-type: none"><li>✓ Design mode</li></ul>                           |
| <p>Q An item is NOT a...</p> <ul style="list-style-type: none"><li>✓ File</li></ul>   | <p>Q Which Page Editor mode would I use to add an image or edit text?</p> <ul style="list-style-type: none"><li>✓ Editing mode</li></ul>                    |
| <p>Q An item's URL is determined by...</p> <ul style="list-style-type: none"><li>✓ Its position in the Content tree</li></ul> |   |



## Topic 1.4 Sitecore from an Developer's Perspective

- |   |   |
|---|---|
| <p><b>Q</b> When a request comes in, Sitecore...</p> <ul style="list-style-type: none"><li>✓ Maps URL to an item in the content tree and dynamically assembles presentation</li></ul> | <p><b>Q</b> Name two <b>foundation features</b> that Sitecore provides out of the box</p> <ul style="list-style-type: none"><li>✓ Content versioning, multi-language support, devices (adaptive design) support</li></ul> |
| <p><b>Q</b> Name the <b>three databases</b> that are installed by default:</p> <ul style="list-style-type: none"><li>✓ Master, web, core</li></ul>                                    | <p><b>Q</b> Name the <b>marketing database</b>:</p> <ul style="list-style-type: none"><li>✓ Analytics</li></ul>   |

## Module 2 Defining Data

### Topic 2.1 Defining Data Structure

- |   |   |
|---|---|
| <p><b>Q</b> An item's <b>type</b> is determined by the _____ used to create it.</p> <ul style="list-style-type: none"><li>✓ Data template</li></ul> | <p><b>Q</b> Where can you find out which data template an item is based on?</p> <ul style="list-style-type: none"><li>✓ Quick Info section on that item</li></ul> |
| <p><b>Q</b> What is an item made up of?</p> <ul style="list-style-type: none"><li>✓ Field sections and fields</li></ul>                             | <p><b>Q</b> Some examples of field types are...</p> <ul style="list-style-type: none"><li>✓ Single-line text, rich text, image, date</li></ul>                    |
| <p><b>Q</b> All fields have a...</p> <ul style="list-style-type: none"><li>✓ Field type</li></ul>   |   |

## 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`, `$id`, `$parentid`, `$parentname`, `$time`, `$now`

## Topic 2.4 Insert Options

### Q What are insert options?

- ✓ A list of item types (data templates) that can be created under a specific item.

### Q Where should insert options be configured?

- ✓ On the data template's standard values. *This can be overridden at item level.*

### Q Why is it a good idea to configure insert options on standard values?

- ✓ All items sharing that data template will have the same insert options, allowing authors to build a tree that is  $n$  levels deep.

## Module 3 Presentation

### Topic 3.1 Presentation Is Dynamic and Modular

### Q What are presentation details?

- ✓ Configuration that determines what an item looks like when requested by the browser.

### Q Describe how Sitecore resolves a page in comparison to a static HTML site.

- ✓ Normally, a URL points to an HTML file. In Sitecore, a request maps to an item, and the page is **dynamically assembled** from smaller pieces.

### Topic 3.2 Preparing to Build

Q What is the name of the main Sitecore .dll?

- ✓ Sitecore.Kernel.dll

Q Where is it recommended that you create the Visual Studio project to work with your Sitecore instance?

- ✓ Outside the web root (*more on this in Module 6 and Module 9*)

### Topic 3.3 Creating a Layout

Q In a Sitecore context, what is the name given to an .aspx file?

- ✓ A layout.

Q By what mechanism is the layout definition connected to the layout file?

- ✓ The Path file on the layout item points to the location of the .aspx file.

Q How do you configure an item's presentation details in Sitecore Rocks?

- ✓ Right-click > Tasks > Design Layout or Ctrl+U. You can also use Commandy.

Q How many layouts can you assign to a single item?

- ✓ One – *per device!*

### 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

Q Name three specialized Sitecore field controls

✓ `<sc:Text />`, `<sc:Date>`, `<sc:Image />`

Q How can Sitecore help you dynamically resize images?

✓ Set `MaxWidth` and/or `MaxHeight` on the `sc:Image` tag.

Q Name the one mandatory property that must be specified on the above controls

✓ `Field` (note: on `<sc:FieldRenderer />`, the equivalent is `FieldName`)

## Module 4 Sitecore API

### Topic 4.1 Basic API Concepts

Q In which .dll can you find the majority of the API?

✓ `Sitecore.Kernel`

Q When Sitecore makes a request, what is the name of the static class that is assembled?

✓ `Sitecore.Context`

Q Name four properties that you might get from `Sitecore.Context`:

✓ `Context user`, `language`, `database`, and `item`

Q When you are looking at a page in Page Editor mode, what is the context database?

✓ `Master`

## Topic 4.1 (continued) Retrieving Items

- |  |   |
|--|---|
| <p>Q Name the method used to retrieve items?</p> <p>✓ <code>GetItem()</code></p>   | <p>Q How should you compare two items in code?</p> <p>✓ Using their IDs</p> |
| <p>Q Items can be retrieved by path or...</p> <p>✓ By ID</p>   |   |
| <p>Q What does the <code>Sitecore.Context.Item.Database</code> property return in the context of a visitor?</p> <p>✓ web</p> |   |

## Topic 4.2 Item Links

- |   |  |
|---|--|
| <p>Q What method do you use to retrieve an item's URL?</p> <p>✓ <code>LinkManager.GetItemUrl()</code></p>                             | <p>Q What object can you pass into the <code>GetItemUrl()</code> method to customize the way your item's URL is rendered?</p> <p>✓ <code>AUrlOptions</code> object</p> |
| <p>Q Why should you not use <code>GetDynamicUrl()</code> for your site's front end?</p> <p>✓ Unreadable 'developer' URL, uses IDs</p> | <p>Q Where can you customize how URLs is rendered globally?</p> <p>✓ In the <code>LinkManager</code> section of the <code>web.config</code></p>                        |

### Topic 4.3 Creating, Deleting and Modifying Items

- |  |   |
|--|---|
| <p><b>Q</b> Sitecore security is an extension of...</p> <ul style="list-style-type: none"> <li>✓ Standard .NET membership</li> </ul> <p><b>Q</b> All code is executed in the context of...</p> <ul style="list-style-type: none"> <li>✓ The current user</li> </ul> <p><b>Q</b> What affects whether or not a piece of code will run?</p> <ul style="list-style-type: none"> <li>✓ That user's permissions</li> </ul> <p><b>Q</b> What permissions does extranet\anonymous lack by default?</p> <ul style="list-style-type: none"> <li>✓ Permission to create or edit items</li> </ul> | <p><b>Q</b> How can you force a piece code to run even though the currently logged in user does not have the appropriate permissions?</p> <ul style="list-style-type: none"> <li>✓ Recommended practice is to use the <b>UserSwitcher</b> to run in the context of a user that does have permission. The <b>SecurityDisabler</b> may also be used.</li> </ul> |
|--|---|

- |  |  |
|--|--|
| <p><b>Q</b> When creating or editing an item, the code must be executed with the appropriate security rights. Name two ways that this can be done.</p> <ul style="list-style-type: none"> <li>✓ UserSwitcher or SecurityDisabler</li> </ul> <p><b>Q</b> How do you put an item into an editing state?</p> <ul style="list-style-type: none"> <li>✓ <code>Item.Editing.BeginEdit();</code></li> </ul> <p><b>Q</b> Why should you create / edit items in the master rather than the web database?</p> <ul style="list-style-type: none"> <li>✓ Web overwritten by publish</li> </ul> | <p><b>Q</b> What property do you use to update?</p> <ul style="list-style-type: none"> <li>✓ <code>Item.Editing.EndEdit();</code></li> </ul> <p><b>Q</b> Why should you not output a field's value straight to the screen?</p> <ul style="list-style-type: none"> <li>✓ Not editable in Page Editor and complex fields contain custom XML</li> </ul> <p><b>Q</b> Which field types are suitable for editing using the <code>.Value</code> property?</p> <ul style="list-style-type: none"> <li>✓ Simple text fields – e.g. Single-Line Text</li> </ul> |
|--|--|

### Topic 4.4 Working with Complex Fields

- |   |   |
|---|---|
| <p><b>Q</b> Why should you never output raw value to screen?</p> <ul style="list-style-type: none"> <li>✓ Not editable in Page Editor, complex field types will not make sense – e.g., image</li> </ul> <p><b>Q</b> Name the best suited field object for the following field types: single-line text, treelist, droplink, general link.</p> <ul style="list-style-type: none"> <li>✓ Field, MultilistField, ReferenceField, LinkField</li> </ul> | <p><b>Q</b> What method should you use to render the contents of text, date, image, and link fields to the screen (and why)?</p> <ul style="list-style-type: none"> <li>✓ <code>FieldRenderer.Render()</code> – Page Editor support, transforms custom XML, transforms RichText links</li> </ul> <p><b>Q</b> Why can't you render a multilist field to the page using <code>.Render()</code>?</p> <ul style="list-style-type: none"> <li>✓ It contains IDs, not readable content</li> </ul> |
|---|---|



## Module 5    Advanced Presentation Concepts

### Topic 5.1    Datasources

- |   |   |
|---|---|
| <p><b>Q</b> Setting a component datasource allows that component to ____.</p> <ul style="list-style-type: none"><li>✓ Output content from a content item elsewhere in the tree.</li></ul>   | <p><b>Q</b> What do parameters allow you to do?</p> <ul style="list-style-type: none"><li>✓ Allows properties to be set per instance of a component</li></ul> |
| <p><b>Q</b> How can you force a Sitecore control to output the content of an item's datasource?</p> <ul style="list-style-type: none"><li>✓ Set <code>DataSource</code> property in code-behind</li><li>✓ Set <code>Item</code> property in code-behind</li><li>✓ Set <code>DataSource</code> property on the control</li></ul> | <p><b>Q</b> Which property do you use to retrieve parameters?</p> <ul style="list-style-type: none"><li>✓ <code>.Parameters</code> property</li></ul>         |

- |   |  |
|---|--|
| <p><b>Q</b> What utility converts parameter lists to a <code>NameValueCollection</code>?</p> <ul style="list-style-type: none"><li>✓ <code>WebUtil.ParseUrlParameters()</code></li></ul>        | <p><b>Q</b> Why would you set the component and the datasource on the standard values?</p> <ul style="list-style-type: none"><li>✓ When you want all items to have that component with that particular datasource</li></ul>                                |
| <p><b>Q</b> Why would you set the datasource on a component on the item itself?</p> <ul style="list-style-type: none"><li>✓ When your component and its datasource appear on one page</li></ul> | <p><b>Q</b> Why would you set the component on the standard values and override the datasource on the item?</p> <ul style="list-style-type: none"><li>✓ When your component has to appear on every page but has a different datasource each time</li></ul> |

## Topic 5.2 Layout Deltas

Q When does a layout delta get created?

- ✓ When presentation coming from the standard values of an item's template is overridden on that item.

Q How does the XML of a layout delta differ from the XML specified on a template's standard values?

- ✓ Layout deltas only contain differences between an item's standard value presentation and any changes made by the author on the item.

The standard value XML contains all presentation details.

## Module 6 Real World Solutions

### Topic 6.1 Familiar Concepts

Q What can you do with methods that could be used across multiple Sitecore projects?

- ✓ Abstract code to a generic utilities project

Q Why can't you preview items under Global?

- ✓ They do not have presentation details

Q Why is the Settings item outside any individual site folder structure?

- ✓ Values shared across all sites

Q What kind of content can you display using a General Widget?

- ✓ Promotions, announcements – anything that could be re-used across the site

Q Name 2 devices you may want to target with different presentation details:

- ✓ Tablets and mobiles

## Topic 6.2 Dealing with Larger Sites

- |  |   |
|--|---|
| <p><b>Q</b> What is the difference between default site language and default client language?</p> <ul style="list-style-type: none"><li>✓ Site is for visitors, client is for authors</li></ul>                                | <p><b>Q</b> What can you do to avoid getting an empty item in your list when getting an item's children?</p> <ul style="list-style-type: none"><li>✓ Check that a version exists in the context language</li></ul>  |
| <p><b>Q</b> How do visitors to your site change the site's language?</p> <ul style="list-style-type: none"><li>✓ You have to build language switching for the front-end yourself – using a query string, for example</li></ul> | <p><b>Q</b> What are some of the dangers of a multi-site implementation?</p> <ul style="list-style-type: none"><li>✓ Code base quickly gets messy, licensing implications – see Multi-site slide for more</li></ul> |

## Topic 6.3 Sitecore Query

- |  |   |
|--|---|
| <p><b>Q</b> What will <code>.return()</code> return?</p> <ul style="list-style-type: none"><li>✓ The context item – such as, the one you are currently viewing</li></ul> | <p><b>Q</b> Why shouldn't you query the entire content tree?</p> <ul style="list-style-type: none"><li>✓ Expensive – particularly if you are iterating through all descendants</li></ul>              |
| <p><b>Q</b> Name two axes available in Sitecore query:</p> <ul style="list-style-type: none"><li>✓ <code>ancestor-or-self::</code>, <code>parent::</code></li></ul>      | <p><b>Q</b> Unless you are doing a very simple query on a very limited area of your Sitecore tree, what should you use instead?</p> <ul style="list-style-type: none"><li>✓ Sitecore search</li></ul> |

## Module 7 Configuring the Page Editor

### Topic 7.1 Datasource Restrictions

- |   |   |
|---|---|
| <p><b>Q</b> What does the <i>Datasource Location</i> field automatically do in the Page Editor?</p> <p>✓ Opens the <b>Select the Associated Content</b> dialog when authors add a new component</p> <p><b>Q</b> What happens if the <i>Datasource Location</i> and the <i>Datasource Template</i> fields are filled in?</p> <p>✓ The <i>Select the Associated Content</i> dialog opens and authors can create a new content item using the template specified</p> | <p><b>Q</b> What happens if the <i>Datasource Location</i> field is empty, but the <i>Datasource Template</i> field is not?</p> <p>✓ The dialog doesn't open automatically, but when the author requests it, they see the whole tree with irrelevant items ghosted.</p> |
|---|---|

### Topic 7.2 Parameters and Parameter Templates

- |  |   |
|--|---|
| <p><b>Q</b> What is the benefit of rendering parameter templates to authors?</p> <p>✓ They reduce error - authors do not have to specify the <b>parameter name</b>, and they can choose values from dropdowns.</p> <p><b>Q</b> What data template does your new data template need to be based on?</p> <p>✓ Standard Rendering Parameters template</p> | <p><b>Q</b> If you want your component to use this new parameters template, where do you assign it?</p> <p>✓ On the component definition item</p> |
|--|---|

### Topic 7.3 Placeholder Restrictions

- |  |  |
|--|--|
| <p><b>Q</b> When you add <code>&lt;sc:placeholder key="myKey" runat="server" /&gt;</code> to your sublayout, what must you do for the placeholder to be selectable in the Page Editor?</p> <p>✓ Create a placeholder settings definition item in Sitecore.</p> | <p><b>Q</b> What else do these placeholder settings items allow you to specify for authors?</p> <p>✓ Which components authors can insert into a placeholder.</p> |
|--|--|

### Topic 7.3 (continued) Compatible Renderings / Allowed Controls

- |  |  |
|--|--|
| <p><b>Q</b> When you add <code>&lt;sc:placeholder key="myKey" runat="server" /&gt;</code> to your sublayout, what must you do for the placeholder to be selectable in the Page Editor?</p> <p>✓ Create a placeholder settings definition item in Sitecore.</p> | <p><b>Q</b> What else do these placeholder settings items allow you to specify for authors?</p> <p>✓ Which components authors can insert into a placeholder.</p> |
|--|--|

### Topic 7.4 Advanced Page Editor Configuration

- |  |  |
|--|--|
| <p><b>Q</b> Once Custom Experience Buttons are assigned, where do they show up?</p> <p>✓ On a component / field and displayed in the Page Editor</p> <p><b>Q</b> Name two examples of Custom buttons:</p> <p>✓ Field Editor and WebEdit button</p> <p><b>Q</b> On what type of definition item can you assign Custom Experience buttons?</p> <p>✓ A component definition item or a field definition item</p> | <p><b>Q</b> What do Edit Frames do?</p> <p>✓ 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</p> <p><b>Q</b> In which database do you create the Custom Experience buttons and edit frames?</p> <p>✓ Core</p> |
|--|--|

## Module 8 Dealing with Your Data

### Topic 8.1 Item Buckets

- |   |   |
|---|---|
| <p><b>Q</b> Use a bucket when...</p> <ul style="list-style-type: none"> <li>✓ You do not care about having a hierarchical item structure and/or you have a large number of items</li> </ul> <p><b>Q</b> In order to store an item in a bucket, you must...</p> <ul style="list-style-type: none"> <li>✓ Make that item or the data template's standard values it is based on <b>bucketable</b></li> </ul> <p><b>Q</b> After making changes to the bucketability of the items in a bucket, you must...</p> <ul style="list-style-type: none"> <li>✓ Sync the bucket</li> </ul> | <p><b>Q</b> What do facets allow you to do?</p> <ul style="list-style-type: none"> <li>✓ Progressively apply filters (based on fields) to narrow down your result set</li> </ul> <p><b>Q</b> How do you create a bucket?</p> <ul style="list-style-type: none"> <li>✓ Select the item you want to turn into a bucket and click the 'Bucket' command in the Configure tab</li> </ul> |
|---|---|

### Topic 8.2 Search

- |   |  |
|---|--|
| <p><b>Q</b> What defines how Sitecore items should be indexed?</p> <ul style="list-style-type: none"> <li>✓ Index configuration files in /App_Config/Include</li> </ul> <p><b>Q</b> Search is built using a provider model. What does this mean?</p> <ul style="list-style-type: none"> <li>✓ You can plug in whatever search provider you want</li> </ul> <p><b>Q</b> What syntax do you use to query an index?</p> <ul style="list-style-type: none"> <li>✓ LINQ</li> </ul> | <p><b>Q</b> What is the name of Sitecore's default search result class?</p> <ul style="list-style-type: none"> <li>✓ <code>SearchResultItem</code></li> </ul> <p><b>Q</b> When building your own search result class, how do you account for fields with spaces?</p> <ul style="list-style-type: none"> <li>✓ Decorate with <code>[IndexField("Page Heading")]</code></li> </ul> <p><b>Q</b> What method should you use to return a rich results object?</p> <ul style="list-style-type: none"> <li>✓ <code>.GetResults();</code></li> </ul> |
|---|--|

- |   |  |
|---|--|
| <p><b>Q</b> You can index fields by type or...</p> <ul style="list-style-type: none"> <li>✓ Name</li> </ul> <p><b>Q</b> You can include/exclude by individual field names or...</p> <ul style="list-style-type: none"> <li>✓ Item templates</li> </ul> <p><b>Q</b> Which attribute must be set to 'YES' in order for values to be stored in the index?</p> <ul style="list-style-type: none"> <li>✓ <code>storageType</code></li> </ul> <p><b>Q</b> Which two search providers does Sitecore ship with?</p> <ul style="list-style-type: none"> <li>✓ Lucene and Solr</li> </ul> | <p><b>Q</b> If you wanted to store the number of comments a news article has, what type of field might you use to perform the calculation?</p> <ul style="list-style-type: none"> <li>✓ A computed field</li> </ul> <p><b>Q</b> Why should you tune your index configuration and not index everything by default?</p> <ul style="list-style-type: none"> <li>✓ Large, unwieldy index as your solution grows</li> </ul> |
|---|--|

## Module 9 Recommended Practices

### Topic 9.1 Working with Media

- |  |   |
|--|---|
| <p><b>Q</b> How and where is media stored?</p> <ul style="list-style-type: none"> <li>✓ In the database as a regular item based on a template</li> </ul>                             | <p><b>Q</b> Can you override the types and properties for the media types?</p> <ul style="list-style-type: none"> <li>✓ Yes</li> </ul>  |
| <p><b>Q</b> What template is the Pdf media type using?</p> <ul style="list-style-type: none"> <li>✓ PDF</li> </ul>   | <p><b>Q</b> What is the default Sitecore extension for media items?</p> <ul style="list-style-type: none"> <li>✓ .ASHX</li> </ul>   |
| <p><b>Q</b> Are those templates configurable?</p> <ul style="list-style-type: none"> <li>✓ Yes</li> </ul>  | <p><b>Q</b> What property do you need to make a change to so that media is displayed with the real file ending?</p> <ul style="list-style-type: none"> <li>✓ <code>Media.RequestExtension</code></li> </ul> |
| <p><b>Q</b> In which file can you change what template Sitecore uses for that particular media item?</p> <ul style="list-style-type: none"> <li>✓ <code>Web.Config</code></li> </ul> |   |

### Topic 9.2 Caching and Performance

- |  |  |
|--|--|
| <p><b>Q</b> What are the 3 layers that the the item cache mechanism have?</p> <ul style="list-style-type: none"> <li>✓ Item, Data and Prefetch</li> </ul>              | <p><b>Q</b> Cache settings are defined in what file?</p> <ul style="list-style-type: none"> <li>✓ <code>web.config</code></li> </ul>   |
| <p><b>Q</b> What operation clears the HTML cache?</p> <ul style="list-style-type: none"> <li>✓ Publishing</li> </ul>   | <p><b>Q</b> Name 3 places where caching options can be defined</p> <ul style="list-style-type: none"> <li>✓ Definition item, standard values or on per instance of the component</li> </ul>    |
| <p><b>Q</b> Which cache is populated when the application starts?</p> <ul style="list-style-type: none"> <li>✓ Prefetch cache</li> </ul>                               | <p><b>Q</b> What 2 modes in Page Editor show profile, cache settings and HTML output for individual components?</p> <ul style="list-style-type: none"> <li>✓ Profile and Debug mode</li> </ul> |
| <p><b>Q</b> What is the path for the .aspx page that clears all cache?</p> <ul style="list-style-type: none"> <li>✓ <code>/Sitecore/admin/cache.aspx</code></li> </ul> |  |

### Topic 9.3 Publishing

<p><b>Q</b> Name three publishing modes</p> <ul style="list-style-type: none"> <li>✓ Incremental , smart and re-publish</li> </ul>	<p><b>Q</b> Publishing restrictions can be applied to _____ , _____ and _____</p> <ul style="list-style-type: none"> <li>✓ Numbered versions of items , the item itself and targets</li> </ul>
--	--

### Topic 9.4 Installing and Scaling Sitecore

<p><b>Q</b> List some benefits of using Sitecore Installer</p> <ul style="list-style-type: none"> <li>✓ Checks prerequisites, correct DLLs, logs, remove an existing installation</li> </ul>	<p><b>Q</b> Name some features of Sitecore Instance Manager</p> <ul style="list-style-type: none"> <li>✓ Packages, customize, extend, reinstall &amp; remove instances</li> </ul>
<p><b>Q</b> Can you do a manual installation?</p> <ul style="list-style-type: none"> <li>✓ Yes</li> </ul>	<p><b>Q</b> Name some features of Sitecore Rocks</p> <ul style="list-style-type: none"> <li>✓ Packages, pick steps to include and remove instances</li> </ul>
<p><b>Q</b> What is the minimum recommendation for production environment?</p> <ul style="list-style-type: none"> <li>✓ CM with core, master and web</li> <li>✓ CD with core and web</li> <li>✓ Separate DB server</li> <li>✓ Use scaling guide</li> </ul>	<p><b>Q</b> Where is the Include folder?</p> <ul style="list-style-type: none"> <li>✓ App_Config</li> </ul>
	<p><b>Q</b> Where can you view the merged web.config?</p> <ul style="list-style-type: none"> <li>✓ /sitecore/admin/showconfig.aspx</li> </ul>

### Topic 9.5 Team Development and Development Environment

<p><b>Q</b> Name two development models</p> <ul style="list-style-type: none"> <li>✓ Inside and outside the web root</li> </ul>	<p><b>Q</b> What are Sitecore packages and what do they contain?</p> <ul style="list-style-type: none"> <li>✓ zip files that contain items and code files</li> </ul>
<p><b>Q</b> Which is the best practice?</p> <ul style="list-style-type: none"> <li>✓ Development outside the web root</li> </ul>	<p><b>Q</b> What is the name of the .aspx that you go to when upgrading Sitecore?</p> <ul style="list-style-type: none"> <li>✓ /sitecore/admin/updateinstallationwizard.aspx</li> </ul>
<p><b>Q</b> What are the options to source control your items?</p> <ul style="list-style-type: none"> <li>✓ You can serialize your items and files or use TDS</li> </ul>	



## Topic 9.6 How to Deal with Deployment

- |   |   |
|---|---|
| <p><b>Q</b> What is the difference between Sitecore publish and deployment</p> <ul style="list-style-type: none"> <li>✓ Publishing synchronizes items between various databases, deployment synchronizes environments</li> </ul> <p><b>Q</b> How would you update your developer environment with the authored environment?</p> <ul style="list-style-type: none"> <li>✓ DB backups/packages</li> </ul> | <p><b>Q</b> Name some points to remember when deploying to the authors environment</p> <ul style="list-style-type: none"> <li>✓ Content freeze, DB backups, be careful what items you override, optionally deploy to a temp location</li> </ul> |
|---|---|

## Topic 9.7 Security

- |   |   |
|---|---|
| <p><b>Q</b> Which application is used to define access rights?</p> <ul style="list-style-type: none"> <li>✓ Security Editor</li> </ul> <p><b>Q</b> Which application is used to view the resolved access rights</p> <ul style="list-style-type: none"> <li>✓ Access Viewer</li> </ul> <p><b>Q</b> You can assign access rights to ____ and ____</p> <ul style="list-style-type: none"> <li>✓ Users and roles</li> </ul> | <p><b>Q</b> Can access be explicitly given to a user to override the role access right?</p> <ul style="list-style-type: none"> <li>✓ Yes</li> </ul> <p><b>Q</b> Name three permissions that are applied to access rights</p> <ul style="list-style-type: none"> <li>✓ Allow, deny and not specified = deny</li> </ul> |
|---|---|

## Topic 9.8 Workflows

- |   |   |
|---|---|
| <p><b>Q</b> What are workflows used for?</p> <ul style="list-style-type: none"> <li>✓ Content approval, versioning and tracking</li> </ul> <p><b>Q</b> Items go through a series of ____</p> <ul style="list-style-type: none"> <li>✓ States</li> </ul> <p><b>Q</b> Each state can contain certain ____ and ____</p> <ul style="list-style-type: none"> <li>✓ Commands and actions</li> </ul> | <p><b>Q</b> When is an item publishable?</p> <ul style="list-style-type: none"> <li>✓ In the final state</li> </ul> <p><b>Q</b> When is a new numbered version created?</p> <ul style="list-style-type: none"> <li>✓ When the published item is edited again</li> </ul> |
|---|---|

## Module 10 Marketing Functionality

### Topic 10.1 Introduction to the XP

- |   |  |
|---|--|
| <p><b>Q</b> Name three key marketing features offered by Sitecore XP:</p> <ul style="list-style-type: none"><li>✓ Measure engagement using goals, personalize visitor experience, optimize by testing</li></ul> | <p><b>Q</b> Why should you build to support marketing functionality from the very beginning?</p> <ul style="list-style-type: none"><li>✓ Clients often want to leverage marketing functionality in Phase 2 – be prepared for them or face awkward rebuilds</li></ul> |
|---|--|

### Topic 10.2 Engagement Value and Goals

- |  |  |
|--|--|
| <p><b>Q</b> Engagement is measured using...</p> <ul style="list-style-type: none"><li>✓ Goals</li></ul> <p><b>Q</b> Give an example of a business-critical goal in the context of the Sitecore Cycling Holidays company.</p> <ul style="list-style-type: none"><li>✓ Booking a holiday</li></ul> <p><b>Q</b> What reporting interface would you use to see trends over time?</p> <ul style="list-style-type: none"><li>✓ Executive Insight Dashboard</li></ul> | <p><b>Q</b> Where would you go to see a detailed report on a visit?</p> <ul style="list-style-type: none"><li>✓ Engagement Analytics reports</li></ul> <p><b>Q</b> In what way does componentization and the use of data sources support testing?</p> <ul style="list-style-type: none"><li>✓ Tests set up to change component or component data source out of the box – hardcoded components and/or content not supported</li></ul> |
|--|--|

## Topic 10.3 Profiling and Personalization

Q What does it mean to *profile content*?

- ✓ Assigning points in certain categories based on the audience a content item is aimed at.

Q How does componentization and use of data sources support personalization?

- ✓ As a visitor navigates around the site, components can be added / hidden or their data changed to match a particular pattern card or profile key score

## Module 11 Optional Topics

### Topic 11.1 Branch Templates

Q What function do branch templates have?

- ✓ Allow authors to create an item and a list of child items in one click

## Topic 11.2 Other Item and Template Properties

- |  |   |
|--|---|
| <p><b>Q</b> Items exist in a ___ and ___ version.</p> <ul style="list-style-type: none"> <li>✓ Language and numbered version</li> </ul> <p><b>Q</b> What does unversioned for field versioning mean?</p> <ul style="list-style-type: none"> <li>✓ Single value per language</li> </ul> <p><b>Q</b> What <i>does shared for field versioning</i> mean?</p> <ul style="list-style-type: none"> <li>✓ Single value for all numbered versions in all languages</li> </ul> <p><b>Q</b> What is the default field versioning?</p> <ul style="list-style-type: none"> <li>✓ Versioned – different values for all languages and numbers</li> </ul> | <p><b>Q</b> What is the field called where you set up field sorting?</p> <ul style="list-style-type: none"> <li>✓ Sortorder</li> </ul> <p><b>Q</b> Where is this field found?</p> <ul style="list-style-type: none"> <li>✓ Standard fields</li> </ul> <p><b>Q</b> On what type of item do you do field sorting?</p> <ul style="list-style-type: none"> <li>✓ On the field definition item</li> </ul> <p><b>Q</b> Name two fields that you can use to help a multi-lingual audience</p> <ul style="list-style-type: none"> <li>✓ Long Description and Short Description fields - also part of the standard fields</li> </ul> |
|--|---|

## Topic 11.3 Pipelines and Events

- |   |  |
|---|--|
| <p><b>Q</b> What are pipelines</p> <ul style="list-style-type: none"> <li>✓ Break down operations into multiple configurable steps</li> </ul> <p><b>Q</b> Where are pipelines defined?</p> <ul style="list-style-type: none"> <li>✓ In the web.config as &lt;processors /&gt; and &lt;pipelines /&gt;</li> </ul> <p><b>Q</b> In what folder on the file system do you store your pipeline configuration files?</p> <ul style="list-style-type: none"> <li>✓ The include folder</li> </ul> | <p><b>Q</b> What method do all processors implement?</p> <ul style="list-style-type: none"> <li>✓ Process()</li> </ul> <p><b>Q</b> What are the 3 options for creating pipelines</p> <ul style="list-style-type: none"> <li>✓ Add processor to existing pipeline</li> <li>✓ Override and extend existing processors</li> <li>✓ Create new pipelines that are called from code</li> </ul> |
|---|--|

- 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

- Q A rule consists of a \_\_\_\_\_ and \_\_\_\_\_
  - ✓ Condition and action
- Q In what other scenarios can rules be used, besides personalization?
  - ✓ E.g. Insert Options
- Q What should the final result be for a condition for the action to be executed?
  - ✓ True

## Topic 11.5 Placeholder Overrides

- Q Why would you override a placeholder settings item?
  - ✓ If you have a particular page that needs to display different allowed controls in a certain placeholder – you would override the placeholder settings item with the required allowed controls