



sitecore®

Website Development for .NET Developers

Sitecore CEP Platform 7.0

Sitecore Official Curriculum
Course Manual

Website Development for .NET Developers

Sitecore CEP Platform 7.0

Welcome!

Thank you for attending an Official Sitecore Training Course.

We've worked hard to provide you with an outstanding and engaging course that will advance your skills in Sitecore technology.

- **Sitecore Certified Instructors**
Your instructor is a Sitecore technical expert and Instructional expert who has met rigorous standards to be able to deliver your course.
- **All Official Sitecore Training is delivered by Sitecore Certified Instructors.**
- **Sitecore Certification**
Sitecore certification is a valuable credential that demonstrates your skills in Sitecore technology and you will be expected to complete a Sitecore Certification Exam at the end of your training.
- **Post Course Assessment**
Sitecore values your opinion on the courses you take and you will be asked to fill in an assessment questionnaire at the end of your training.

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.

© 2013 Sitecore

Contents

| | | |
|-----------------|---|------------|
| Module 1 | Sitecore Overview | |
| Topic 1.1 | Course Overview | 1-2 |
| Topic 1.2 | What is Sitecore? | 1-5 |
| Topic 1.3 | Sitecore from a User's Perspective | 1-11 |
| Topic 1.4 | Sitecore from a Developer's Perspective | 1-15 |
| Module 2 | Content and Presentation | 2-1 |
| Topic 2.1 | Defining Data Structure | 2-2 |
| Topic 2.2 | Binding Content and Presentation | 2-10 |
| Topic 2.3 | Rendering Content | 2-17 |
| Topic 2.4 | Dynamic Binding | 2-25 |
| Topic 2.5 | Sitecore Rocks | 2-33 |
| Topic 2.6 | Module Summary..... | 2-40 |
| Module 3 | Items Items Items..... | 3-1 |
| Topic 3.1 | Data Template Inheritance | 3-2 |
| Topic 3.2 | Standard Values | 3-8 |
| Topic 3.3 | Insert Options | 3-22 |
| Topic 3.4 | Other Item and Template Properties | 3-27 |
| Module 4 | Sitecore API | 4-1 |
| Topic 4.1 | Basic API Concepts & Retrieving Items | 4-2 |
| Topic 4.2 | Item Links | 4-9 |
| Topic 4.3 | Creating, Deleting and Modifying Items | 4-14 |
| Topic 4.4 | Working with Complex Fields..... | 4-25 |
| Module 5 | Advanced Presentation Concepts..... | 5-1 |
| Topic 5.1 | Reusable Content..... | 5-2 |
| Topic 5.2 | Layout Deltas..... | 5-14 |
| Module 6 | Real World Solutions | 6-1 |
| Topic 6.1 | Introduction to Sitecore Cycling Holidays | 6-2 |
| Topic 6.2 | Application of Familiar Concepts | 6-4 |
| Topic 6.3 | Sitecore Cycling Holidays – Dealing with Larger Sites | 6-16 |
| Topic 6.4 | Sitecore Query | 6-21 |

Contents

| | | |
|------------------|--|-------------|
| Module 7 | Configuring the Page Editor | 7-1 |
| Topic 7.1 | Building for Page Editor | 7-2 |
| Topic 7.2 | Datasource Restrictions | 7-11 |
| Topic 7.3 | Parameters and parameter templates..... | 7-16 |
| Topic 7.4 | Placeholders..... | 7-19 |
| Topic 7.5 | Compatible renderings and allowed controls | 7-26 |
| Topic 7.6 | Advanced Page Editor configuration..... | 7-30 |
| Module 8 | Marketing Functionality | 8-1 |
| Topic 8.1 | Introduction to the CEP..... | 8-2 |
| Topic 8.2 | Engagement Value and Goals | 8-4 |
| Topic 8.3 | Profiling and Personalization | 8-9 |
| Module 9 | Dealing with Your Data..... | 9-1 |
| Topic 9.1 | Item Buckets | 9-2 |
| Topic 9.2 | Search..... | 9-10 |
| Module 10 | Recommended Practices | 10-1 |
| Topic 10.1 | Working with Media | 10-2 |
| Topic 10.2 | Caching and Performance | 10-5 |
| Topic 10.3 | Publishing..... | 10-11 |
| Topic 10.4 | Installing and Scaling Sitecore..... | 10-14 |
| Topic 10.5 | Team Development and the Development Environment | 10-21 |
| Topic 10.6 | How to Deal with Deployment..... | 10-26 |
| Topic 10.7 | Basic Security | 10-28 |
| Topic 10.8 | Workflow..... | 10-30 |
| Module 11 | Optional Topics | 11-1 |
| Topic 11.1 | Branches..... | 11-2 |
| Topic 11.2 | Pipelines and Events | 11-4 |
| Topic 11.2 | Rules..... | 11-9 |
| Module 12 | Support Community | 12-1 |
| Topic 12.1 | Sitecore Community..... | 12-2 |
| Topic 12.2 | Sitecore Support | 12-5 |
| Topic 12.3 | Modules | 12-7 |

About this course

Course Description

This 4 day course teaches learners how to author real world websites using the Sitecore Customer Engagement Platform with Microsoft C# programming.

Audience

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 Best Practices.

Learner 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 and C#
- Microsoft Visual Studio 2012
- Working and interacting with Microsoft SQL Server

Icons used in this manual

**Best Practice**

Adhere to Best Practices whenever possible

**Keyboard Shortcut**

Save time with shortcuts

**Find It**

Additional References on the Sitecore Developer Network

**Important**

Read carefully

**Explanation**

A detailed explanation of a concept

**Code**

A completed code example

**Sitecore API**

Sitecore API reference featured in the exercise

**Knowledge Check**

Master the covered concepts and terminology

**Case Study**

A real world example to illustrate a task

**Tip**

A key feature that is highlighted

Module 1

Sitecore Overview

Contents:

- Course overview
- What is Sitecore?
- Sitecore from an author perspective
- Sitecore from a developer 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
- Discuss the course format
- State the general daily agendas

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 best practices around installation, environments, scaling and optimization
- Extend Sitecore
- Find help and modules



Course format

- Four day course
- 12 modules
1 2 3 4 5 6 7 8 9 10 11 12
- Each module has several topics
Topic 1 Topic 2 Topic 3 Topic x
- Demos and walkthroughs
- Apply labs
- Review
- Extend

Our instructional design follows the **ICARE** model
Introduction
Content
Application - Labs
Review
Extend

Daily agendas – first day

| | | | |
|---|-------|--|------|
| Module 1 Sitecore Overview | 45min | Module 2 Content and Presentation | 3hrs |
| <ul style="list-style-type: none">• Course overview• What is Sitecore?• Sitecore from an author perspective• Sitecore from a developer perspective | | <ul style="list-style-type: none">• Sitecore is just a .NET application• Items, fields and templates• Binding content and presentation• Outputting content• Dynamic binding• Sitecore rocks | |
| Module 3 Items Items Items | 2h30 | Consolidation Labs | |

Second day

| | | | |
|--|------|--|--|
| Module 4 Sitecore API | 3h10 | Module 5 Advanced Presentation Concepts | |
| <ul style="list-style-type: none">• Access control• Basic API concepts• Retrieving items• Item links• Creating, deleting and modifying items• Working with complex fields | | 1h20 | <ul style="list-style-type: none">• Reusable content• Layout deltas |
| Module 6 Real World Solutions | 2h20 | Consolidation Labs | |

Third day

| | |
|---|---|
| Module 7 Configuring the Page Editor 4h25 <ul style="list-style-type: none">• Building for the Page Editor• Datasource restrictions• Parameters and parameter templates• Placeholders• Compatible renderings and allowed controls• Advanced Page Editor configuration | Module 8 Marketing Functionality 1h10 <ul style="list-style-type: none">• Introduction to the CEP• Engagement Value and Goals• Profiling and personalization |
| Module 9 Dealing with Your Data 3h25 <ul style="list-style-type: none">• Validation• Item buckets• Search | Consolidation Labs |

Fourth day

| | |
|--|--|
| Module 10 Recommended Practices 2h40 <ul style="list-style-type: none">• Working with media• Caching and performance• Installing and scaling Sitecore• Team development and environment• Workflow and basic security• Publishing | Module 11 Optional Topics 40min <ul style="list-style-type: none">• Branches• Pipelines and events• Rules |
| Module 12 Support Community 10min <ul style="list-style-type: none">• Sitecore community• Sitecore support• Modules | Assessment Exam and Certification 60min <ul style="list-style-type: none">• Theory exam with 40 questions |

Topic 1.2 What is Sitecore?

Introduction

Objectives

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

- State two key aspects that the Sitecore Customer Engagement Platform blends together
- Explain what Sitecore is and what its minimum requirements are
- Draw a basic architecture diagram
- Draw and explain the developer environment
- Draw and explain the minimum required installation scenario for a production environment
- Explain how authors and developers interact with Sitecore on a day-to-day basis
- State the key responsibility of developers when building a Sitecore implementation

Content

Sitecore is a Customer Engagement Platform

Sitecore is a “blended” solution – it combines web content management with a marketing platform

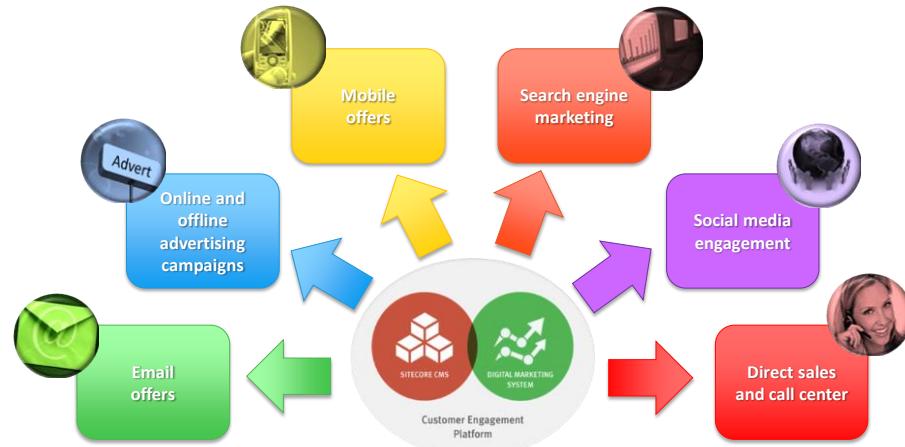
Sitecore can:

- Build and populate a website (**content management**)
- **Respond intelligently** to an individual visitor (**personalization**)
- Integrate **multiple sources of profiling data** for communication
- Consolidate siloed marketing technologies (e.g. web and email)
- Integrate with **external tools and databases** already in use – e.g. Dynamics, SalesForce

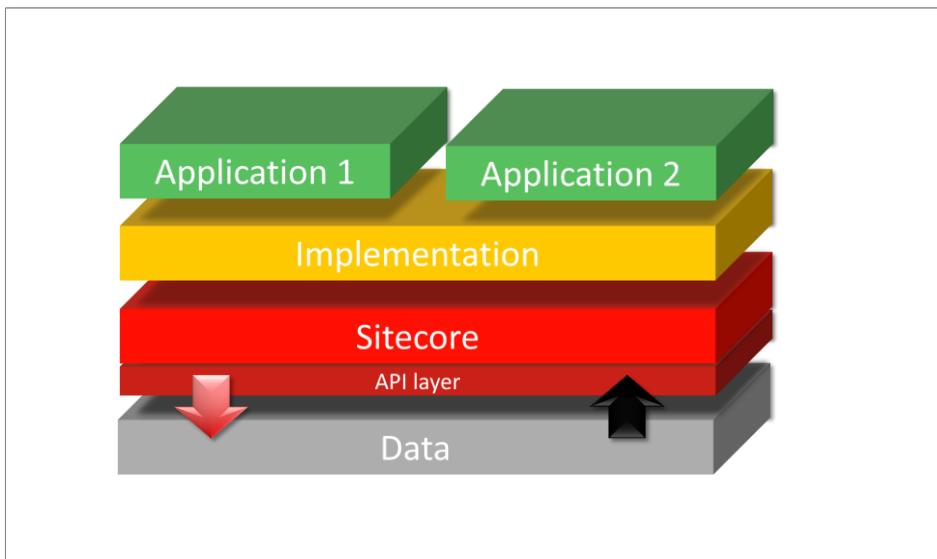


Cross-channel communication

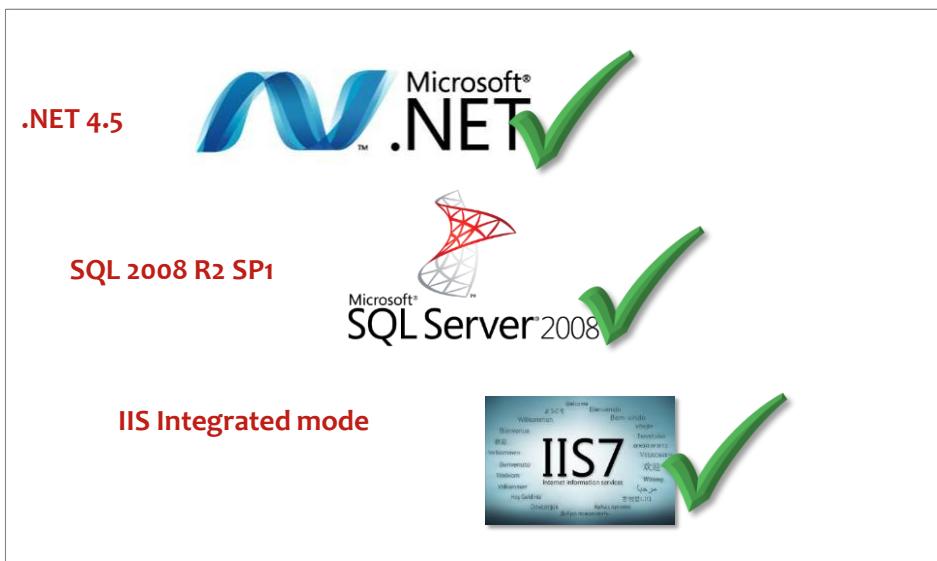
Offering a consistent, individual experience regardless of how visitors interact with your content



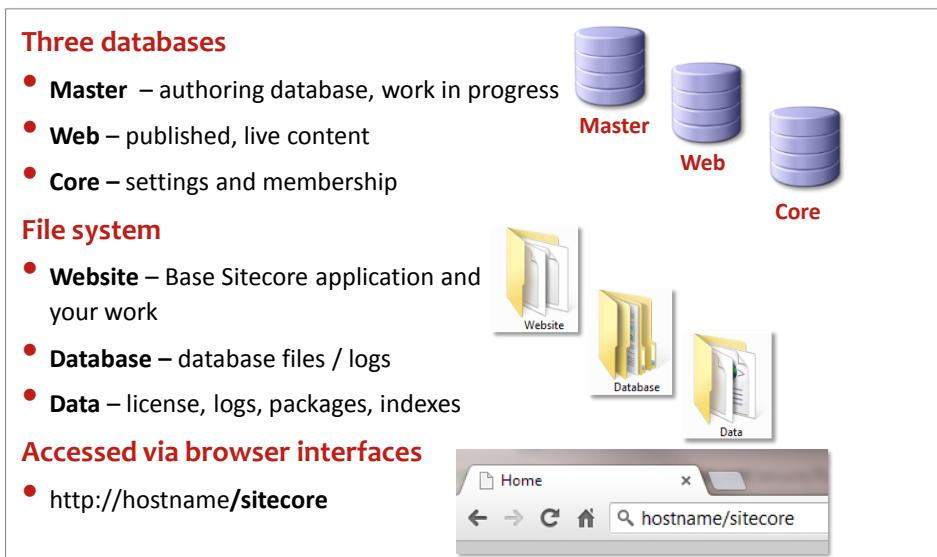
Architecture



Minimum requirements



What gets installed?





Walkthrough – What gets installed?

Installation scenarios



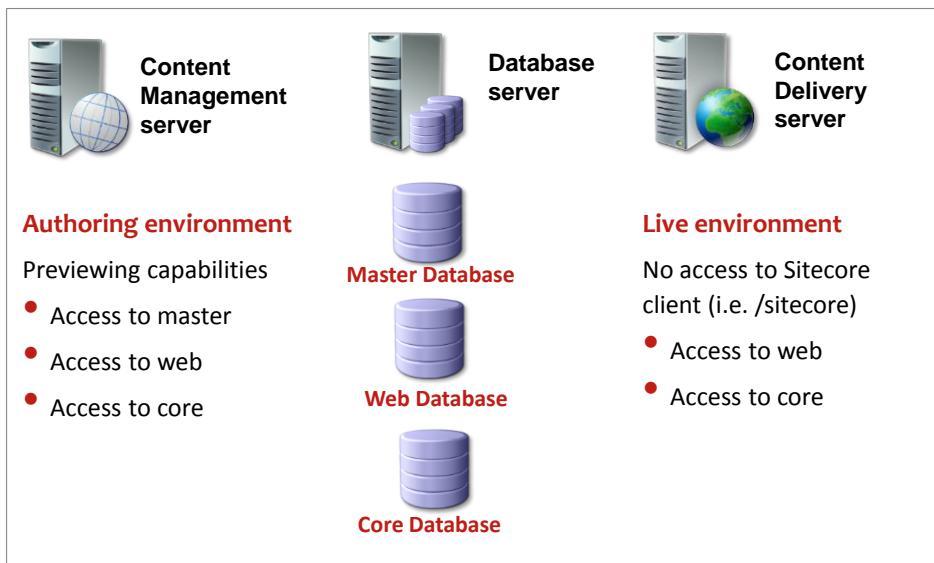
All databases on the same machine

- Master
- Core
- Web

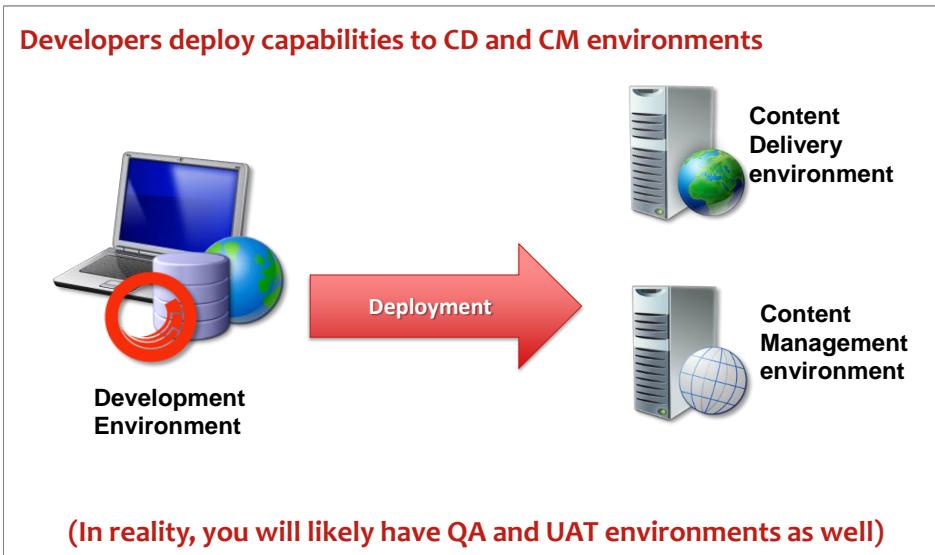
Single IIS site

- Sitecore client (/sitecore)
- Live site (<http://www.sitecore.net>)

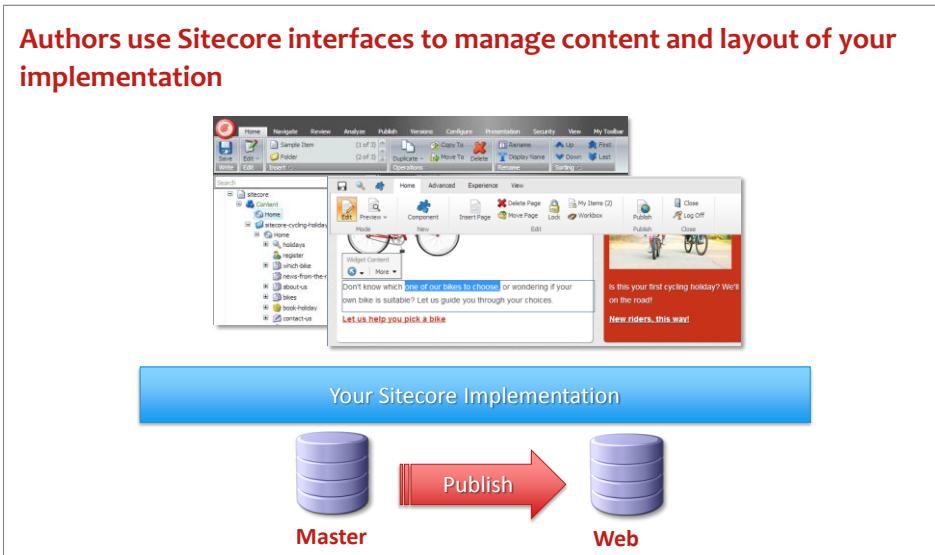
Installation scenarios – recommended minimum



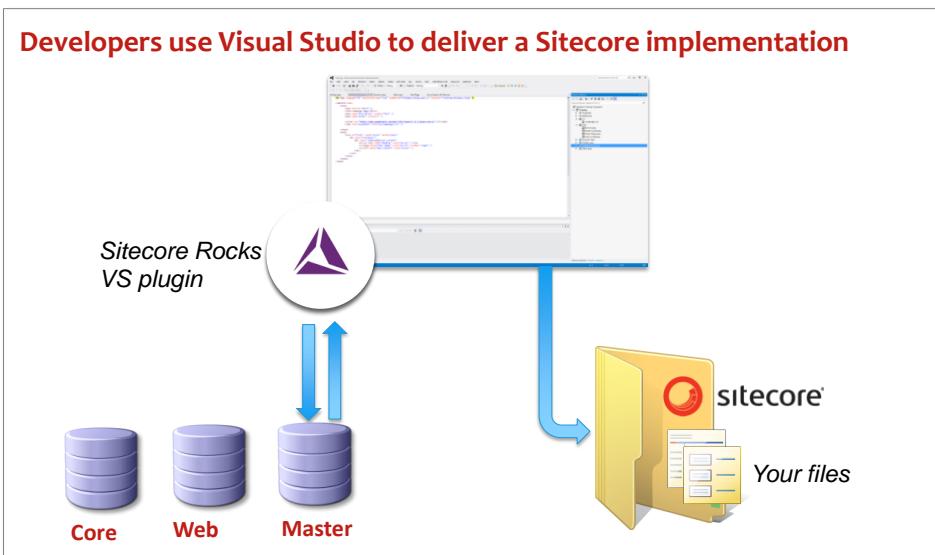
From a system perspective



From an author perspective



From a developer perspective



The developer's responsibility

- You are not building a **website**, you are implementing capabilities that allow Sitecore authors to build and manage a website
- The fundamental aim of this course is to teach you how to build an implementation according to **best practice**
- An implementation built according to best practice will allow marketers to take full advantage of the **content management** as well as the **marketing and engagement analytics** features of Sitecore
- At the end of the day you are empowering your business users



Review

What is Sitecore?

- | | |
|---|--|
| <p>Q Name two key features of the Sitecore CEP:</p> <p>✓ CMS and marketing platform</p> | <p>Q Which databases does a Content Delivery environment have access to?</p> <p>✓ Web and core</p> |
| <p>Q When you install Sitecore, what actually ends up on your machine?</p> <p>✓ A website in the file system and 3 databases</p> | <p>Q Why would you build your implementation with best practices in mind?</p> <p>✓ To allow authors to take advantage of the CMS and the marketing features of Sitecore</p> |
| <p>Q Name the 3 Sitecore databases:</p> <p>✓ Master, core and web</p> | |
| <p>Q Which database contains the 'work in progress' content?</p> <p>✓ Master</p> | |

Topic 1.3 Sitecore from an author 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
- State how Sitecore data is represented to authors
- Name the process by which content is pushed to a live environment

Content



Demo – Sitecore interfaces

1. Open the training site in a browser and append **/sitecore**
2. Log in as the administrator (password: **b**)



Note

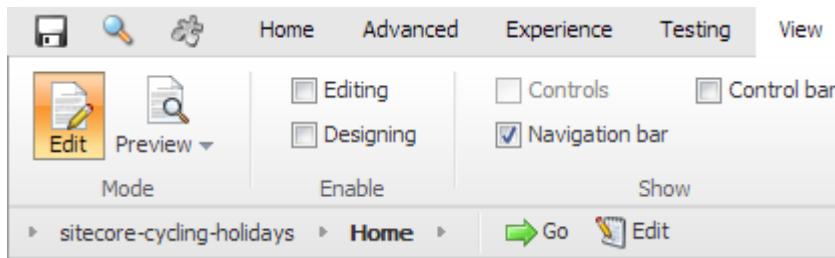
There are three interfaces used for editing: Page Editor, Content Editor and Desktop

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** – common add/delete/move tasks
 - **Experience** – change language
6. Click the **View** tab, notice that both the **Enable** and **Show** checkboxes are not checked
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



1-1 Enabling the Navigation bar

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



Editing

1. In the **View** tab, select the **Editing** checkbox and hover over the various areas of the page – an image, some text, etc. The *Edit Frame* toolbar appears. The editing options available on the toolbar depend upon the type of content (image or text) that was selected



1-2 Floating toolbar for an image

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



Note

The breadcrumbs automatically know what to display

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



Designing

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



Note

You can no longer edit content, but are now selecting areas of functionality on the page. You still get an edit frame, but the options are now things like Move or Delete

4. In the **Home** tab, click **Add Component** (or the corresponding icon next to the actual tab itself). A number of **Add to here** buttons will appear where you can insert a component
5. Click the button at the top of the left-hand column, choose the **Sub Navigation** component, then click **Select**
6. If the list on this page is very long, you might want to give people 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 appeared on the page that lists the titles of all pages, including the one that was just created



Desktop and Content Editor

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. Search for the words **Which bike** using the search bar above the Sitecore tree. The item that was edited earlier will be displayed in the result set
6. Open the item. It is made up of several different types of fields



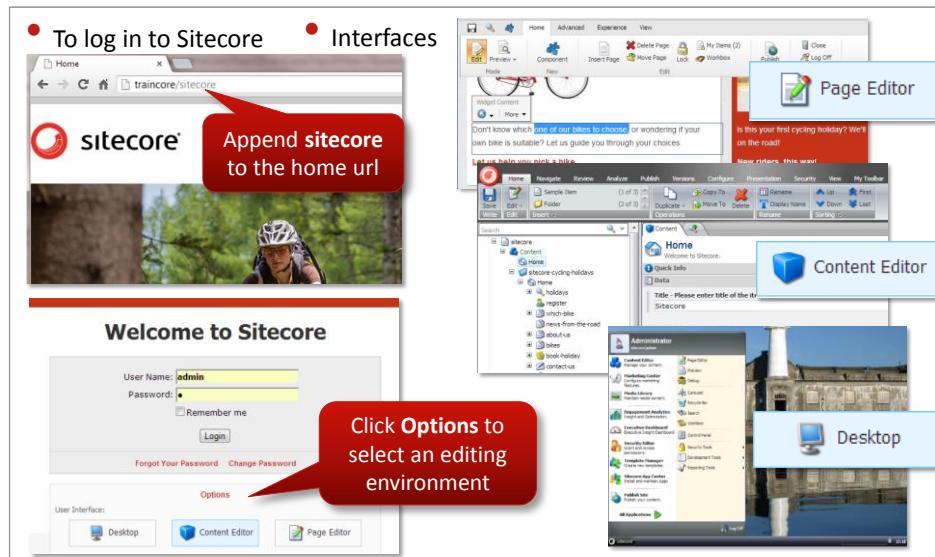
Note

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

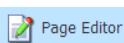
7. Change the **Heading** field to **Choices of bike**, then in the ribbon click **Publish > Preview** to see the changes
8. If you are happy with the changes, click the **arrow** next to **Publish > Publish site**, and choose the **Incremental Publish** option to publish the changed items
9. **Log off** 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

Sitecore as a Web Content Management System

Sitecore provides two main editing tools (Page Editor and Content Editor) for business users to author and design content. It also provides a “Windows”-like Desktop environment, which offers access to tools such as the Security Editor and Engagement Analytics interface



The **Login** screen is reached 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)



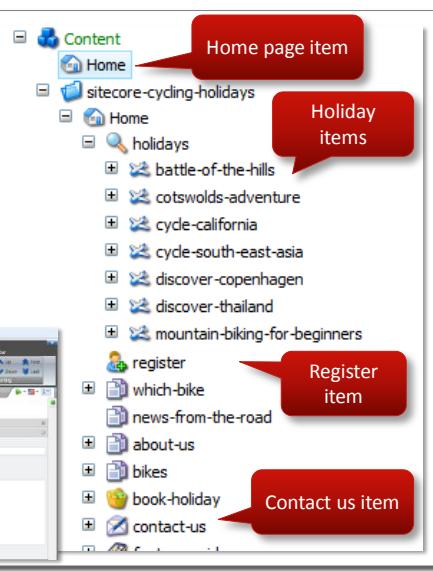
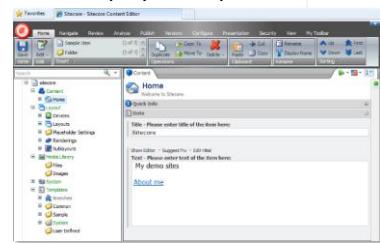
The **Content Editor** allows a business user to work directly with all the data stored in Sitecore. Unlike in the **Page Editor**, authors in this editor environment are editing the actual data excluding any presentation



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
- Sitecore interfaces display all or part of the Content tree



Publishing and the Sitecore databases

Sitecore “**Publishes**” items from the **Master** to the **Web** database



Review

Sitecore from an author perspective

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

Topic 1.4 Sitecore from a developer perspective

Introduction

Objectives

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

- Explain what Sitecore is in terms of a .NET application
- Give at least four examples of standard Sitecore features
- State four drawbacks of a traditional .NET website
- Explain how Sitecore handles a request

Content

Sitecore is a .NET web application

Most projects share a lot of similar groundwork

- End user content and designer tools
- Flexible, hierachal data storage
- Devices like Mobile or Print
- An API to hook into and extend Sitecore
- A media library for binary assets like images & videos
- User and role based access security
- Content workflows for approval
- Content versioning and multiple languages
- Search facility based on provider model



How does Sitecore compare to other methods of building a site?

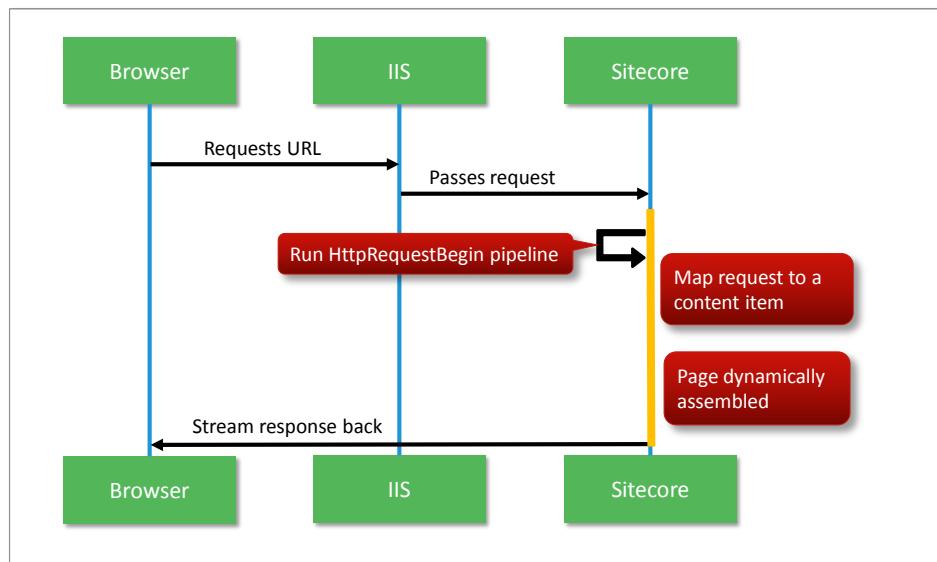
| Method | Challenges |
|--|---|
| <p>A .NET website re-uses user controls and stores page data in a database</p> <ul style="list-style-type: none"> ● A database of objects (e.g. products) with unique IDs and associated content ● An .aspx page per ‘type’ of object with a query string – e.g. StandardContent.aspx?id=5 or Product.aspx?id=27 – you could re-write the URLs ● Re-use of user controls – e.g. a ‘featured product’ control might be given the ID of a product to display, and that control can then be used anywhere ● You might have a backend that allows editors to create new objects (e.g. a new product) and edit information | <ul style="list-style-type: none"> ● Inflexible layout – what happens when you want to create a new page type, or have a page that looks slightly different? (e.g. a news article page without a side bar, or a standard content page without the featured product user control – like ‘Terms & Conditions’) ● Still relying on the file system to generate your URLs and potentially having to rewrite them ● Requires development to extend the system, even if all you want to do is a slightly different incarnation of the same page ● Editors have no control over layout ● Backend interfaces need to be developed and maintained |

How does Sitecore solve these problems?

- Total separation of content and presentation – authors write content, developers define presentation components
- Content and presentation is dynamically assembled at runtime & rendered on demand
- Writing content and determining how it should be rendered are isolated tasks, performed by different people



How Sitecore handles a request



Knowledge Check

What systems or methods have you used and what were the limitations?



Knowledge Check

How does it help Developers if Editors have more control over appearance and site content?

Review

Sitecore from a developer perspective

Q When a request comes in, Sitecore...

- ✓ Maps URL to an item in the content tree and dynamically assembles presentation

Q Name two foundation features that Sitecore provides out of the box

- ✓ Content versioning, multi-language support, devices (adaptive design) support

Extend

View example Sitecore implementations

- Visit various Sitecore Site of The Year Winners

The grid displays six website examples built with Sitecore:

- www.easyjet.com
- www.jabra.com
- www.dornbracht.com
- www.atomic.com
- www.carnationgroup.com
- www.sgs.com

Module 2

Content and Presentation

Contents

- Defining data structure
- Binding content and presentation
- Rendering content
- Dynamic binding
- Sitecore Rocks

Topic 2.1 Defining data structure

Introduction

Objectives

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

- Explain the process of defining a data structure in Sitecore
- Create and edit content

Content

Items, fields and data templates

An item

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

The screenshot shows the Sitecore Experience Editor for a 'Basic Information' item named 'Holidays'. The editor interface includes tabs for 'Basic Information', 'Main Content', and 'Main Image'. The 'Basic Information' tab is active, displaying fields for 'Heading' (set to 'Holidays'), 'Main Content' (containing the text 'Book a great holiday with us.'), and 'Main Image' (linked to '/Images/main-image' which shows a photo of two people riding bicycles). Red callouts highlight the 'Field Section' above the heading field and the 'Various fields' below the main content and main image fields.

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

Data Template defining field sections and fields

The screenshot shows the Sitecore Data Template configuration interface for a 'Basic Information' template. It displays three fields: 'Heading' (Single-Line Text), 'Main Content' (Rich Text), and 'Main Image' (Image). A red callout points to this configuration area with the text 'Data Template defining field sections and fields'.



Demo – Creating content

Creating a data template

1. Log into the Sitecore **Desktop** as admin (password: b)
2. Click the Sitecore button in the bottom left corner of the screen and open the **Template Manager**
3. Right-click on a **User Defined** item and select Insert > New Template
4. Call the data template **Holiday Listing**
5. Create the Basic Information fields as detailed in **Figure 2-1 Holiday Listing data template**
6. Choose an icon for the template with one of these two methods:
 - Click on the larger version of the icon in the **Content** tab
 - Click **Configure > Icon**

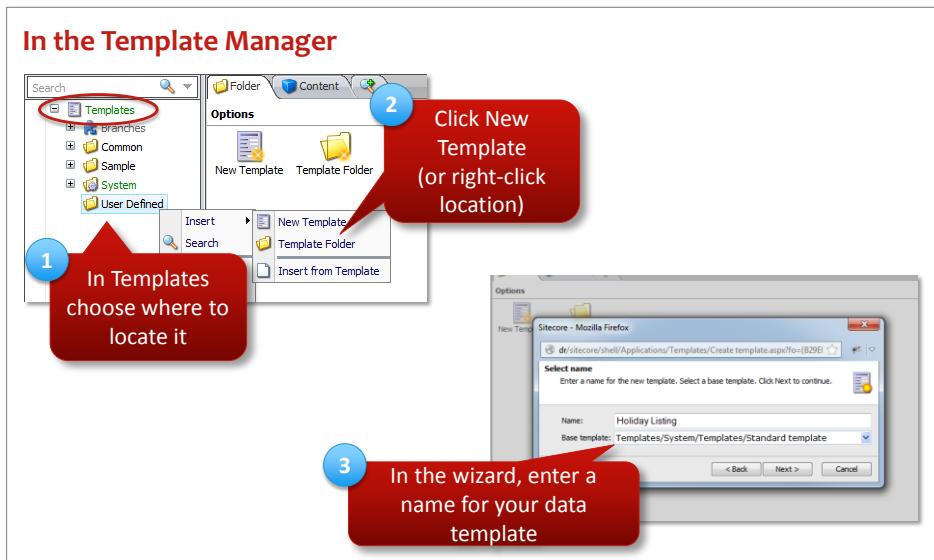
The screenshot shows the Sitecore Data Template configuration interface for a 'Basic Information' template. It displays three fields: 'Heading' (Single-Line Text), 'Main Content' (Rich Text), and 'Main Image' (Image). This is the same configuration shown in the previous screenshot but from a different perspective.

Figure 2-1 Holiday Listing data template

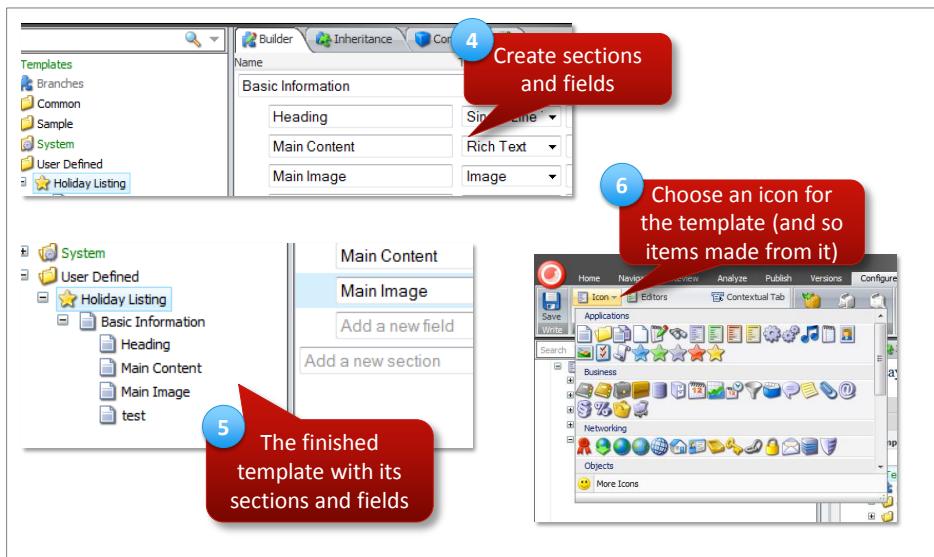
Creating an item

1. Right-click on the **/sitecore/content/Home** item and choose the **Insert from Template** option from the resulting drop-down menu
2. Search for or browse to the **Holiday Listing** data template
3. Type **holidays** into the **Item Name** field and click the **Insert** button
4. Create a few more items based on the Holiday Listing data template (e.g. Cycling Holidays, Family Friendly Holidays, or Exotic Holidays). Each item uses the same data template, but may contain different information)
5. Populate each of the three fields in the new Holidays item with content
6. Show the Quick Info field section and that you can navigate to the data template through it

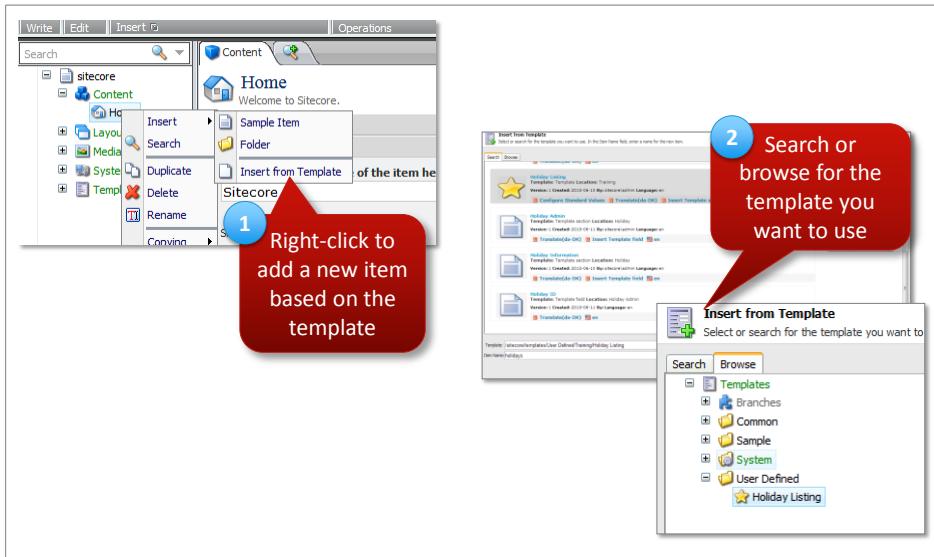
Creating a data template – steps 1 to 3 of 5



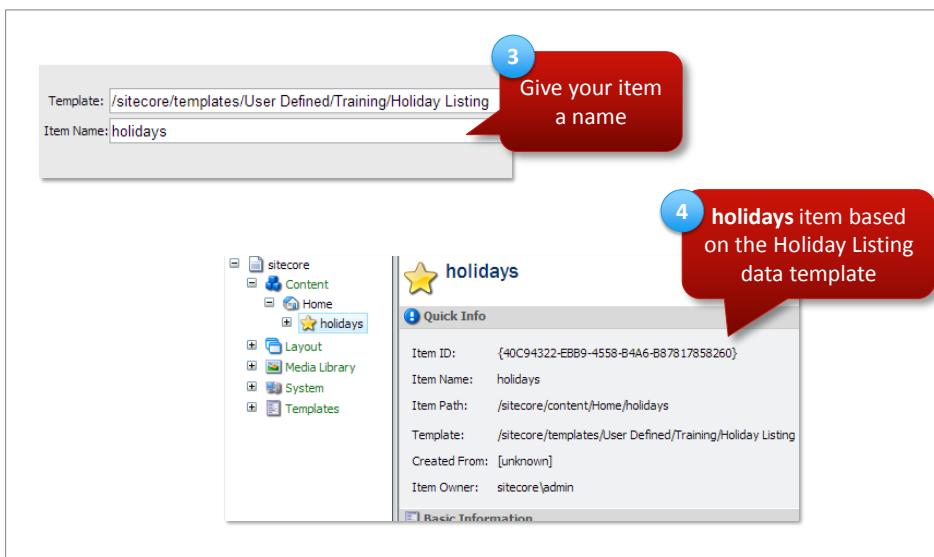
Steps 4 to 5 of 5



Creating an item – steps 1 to 2 of 4



Steps 3 to 4 of 4



Apply – Topic 2.1 – 15 min

Setting the scene

- In an ideal world, you would start a project by **designing your data**
- However, developers are often given **HTML, wireframe, or design** to work from
- Resist the urge to code it up straight away!!**
- Take a **content first** approach and **infer data structure**

Battle of the hills

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking](#) and [adventures](#). Loren ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Non nobis.



Vestibulum admetum est poset, non laoreet voleat esse. Sed non pulvinar vero. In est nunc, adipiscing pharetra ornare. Etiam volutpat ante, vestibulum posueris. Sed ut nulli ut felis hendrerit, volutpat at vel tortor. Suspendisse ad nisl, neman nec tenuis, interdum et malesuada fames ac turpis semper. Donec ultricies, nisl consetetur adipiscing elit. Ut at mollis eros. Suspendisse potest. Donec semper, nisl. Donec ultricies, nisl. Donec ultricies aliquet quis in. Tertius in hac habitatione placet dicitur. Proin tempus ultrices volt, sit amet aliquam nunc aliquet eu.

Sed non pulvinar vero. In est nunc, adipiscing pharetra ornare. Etiam volutpat ante, vestibulum posueris. Sed ut nulli ut felis hendrerit, volutpat at vel tortor. Suspendisse ad nisl, neman nec tenuis, interdum et malesuada fames ac turpis semper. Donec ultricies, nisl consetetur adipiscing elit. Ut at mollis eros. Suspendisse potest. Donec semper, nisl. Donec ultricies, nisl. Donec ultricies aliquet quis in. Tertius in hac habitatione placet dicitur. Proin tempus ultrices volt, sit amet aliquam nunc aliquet eu.

Date: 20th April 2014

Price per person: £2000 per person

Cycling holidays

Our holidays

Cycle London

Mountain Biking in 3 Days

Discover Liverpool

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking](#) and [adventures](#). Loren ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Non nobis.



Vestibulum admetum est poset, non laoreet voleat esse. Sed non pulvinar vero. In est nunc, adipiscing pharetra ornare. Etiam volutpat ante, vestibulum posueris. Sed ut nulli ut felis hendrerit, volutpat at vel tortor. Suspendisse ad nisl, neman nec tenuis, interdum et malesuada fames ac turpis semper. Donec ultricies, nisl consetetur adipiscing elit. Ut at mollis eros. Suspendisse potest. Donec semper, nisl. Donec ultricies, nisl. Donec ultricies aliquet quis in. Tertius in hac habitatione placet dicitur. Proin tempus ultrices volt, sit amet aliquam nunc aliquet eu.

Sed non pulvinar vero. In est nunc, adipiscing pharetra ornare. Etiam volutpat ante, vestibulum posueris. Sed ut nulli ut felis hendrerit, volutpat at vel tortor. Suspendisse ad nisl, neman nec tenuis, interdum et malesuada fames ac turpis semper. Donec ultricies, nisl consetetur adipiscing elit. Ut at mollis eros. Suspendisse potest. Donec semper, nisl. Donec ultricies, nisl. Donec ultricies aliquet quis in. Tertius in hac habitatione placet dicitur. Proin tempus ultrices volt, sit amet aliquam nunc aliquet eu.



Create a data template and an item

In the following labs, you will:

- Create a new data template called Holiday Listing
- Create a an item based on this data template
- Populate the fields with sample content

Lab A. Create a data template called Holiday Listing under the User Defined folder

You will find a blank installation of Sitecore on your student machine. You will be given the URL by your instructor. Throughout this manual, the hostname for the blank Sitecore instance will be referred to as <http://training>, if your hostname is different then please adjust the URLs accordingly

1. Navigate to the **login** page by going to <http://training/sitecore>
2. Click the **options** link to view the available interfaces
3. Log in to the Sitecore **Desktop** as **admin** (password: **b**)
4. Click the **Sitecore** button and open the **Template Manager**
5. Navigate to the **User Defined** folder item

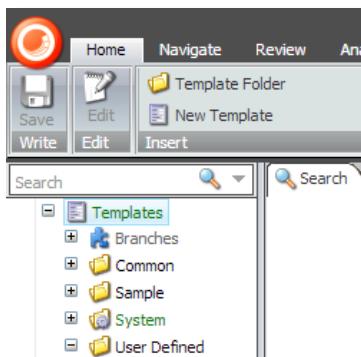


Figure 2-2 User Defined folder

6. Click the **New Template** option (You can also right-click and select **Insert > New Template**), the template wizard opens
7. On the Select Name Dialog box enter **Holiday Listing**

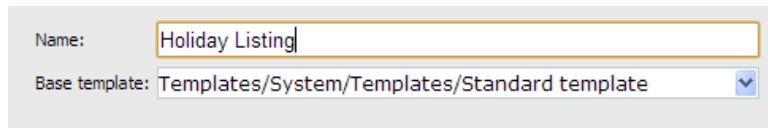


Figure 2-3 Naming the template

8. Click **Next, Next and Finish**
9. Using the **Template Manager**, add a new field section called **Basic Information**
10. Create three fields: **Heading**, **Main Content**, and **Main Image** (as shown below)

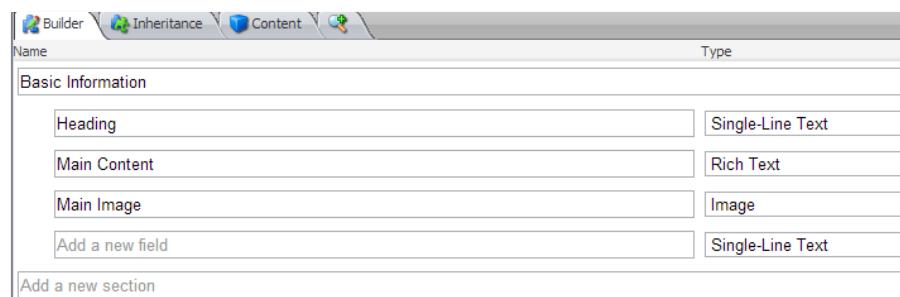


Figure 2-4 Field names to use

11. **Save** your work with the save command in top left of the ribbon



Tip

To save use **CTRL+S**

12. To assign an icon to your data template go to **Configure > Appearance > Icon** and assign the **Gold Star**

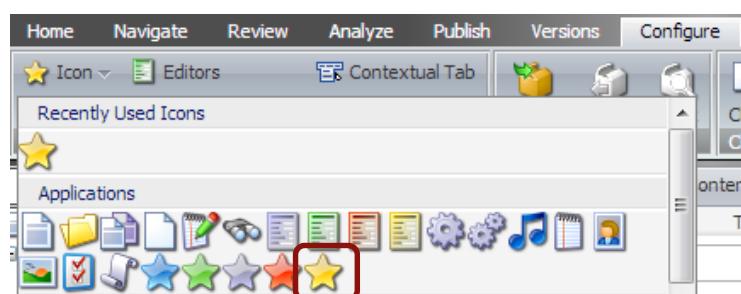


Figure 2-5 Choose the gold star icon

Lab B. Create a new item based on the Holiday Listing data template

1. In the **Content Editor**, right-click the **Home** option
2. In the context menu select **Insert> Insert from Template**. You are now ready to add a new item under the Home item



Figure 2-6 Insert from template

3. In the **Insert from template Search** tab, enter the name of the data template you created in the previous lab (*Holiday Listing*)
4. Click the **Search magnifying lens** button or click **Enter**
5. **Select the Holiday Listing data template**
6. Enter **holidays** in the **Item Name** field
7. Click the **Insert** button

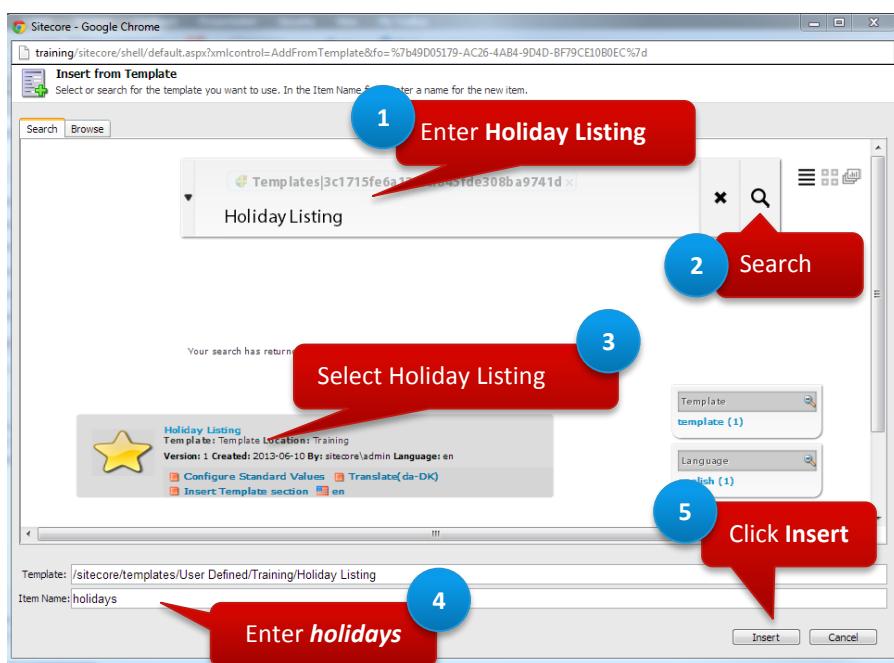


Figure 2-7 Creating a new item from a data template

8. Enter **Holidays** in the **Heading** text field
9. Enter some text in the **Main Content** text field. (“Book a great holiday with us”)

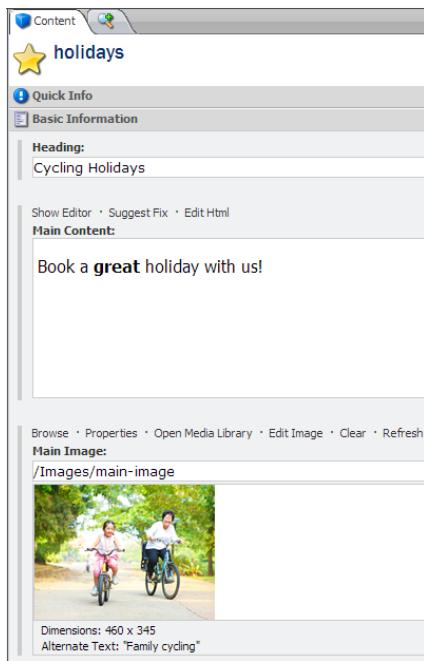


Figure 2-8 Editing the Holidays item fields



Note

Use the Show Editor button to add text into the Page Content rich text field

10. Before you can insert an image into the **Main Image** field, you must upload some to the Media Library.
Select the Media Library item content tree
11. Select the **Images** folder item in the **content tree**
12. Make sure that the **folder tab** is selected and click the **Upload Files (Advanced)** icon

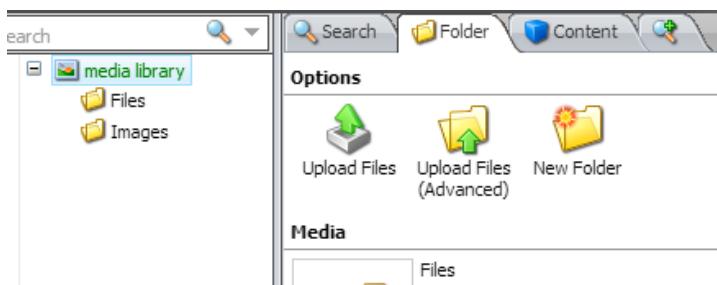


Figure 2-9 Choose Upload Files (Advanced)

13. Go to the **Images.zip** file in the **student resources folder** (*WND Labs > Module 2 > Topic 2.1 > Lab B > Images.zip*), select it, and click the **OK** button
14. Select the **Unpack ZIP Archives** checkbox and click the **Upload** button

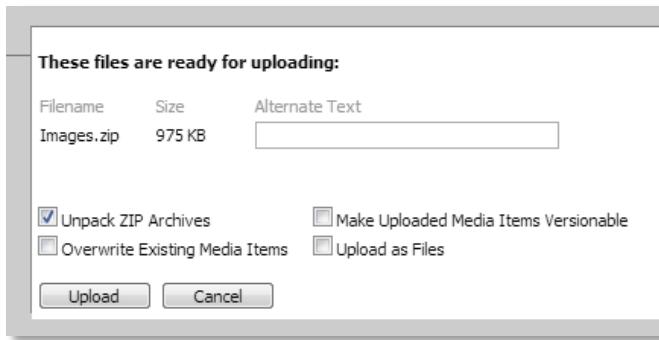


Figure 2-10 Advanced upload options

15. In the **Content Editor**, select the **holidays** item you were editing in steps 8-9
16. Click the **Browse** button that is next to the **Main Image** field, which will open the Media Browser
17. In the browser, select one of the images you uploaded in **/media library/Images** and click the **OK** button
18. **Save**



Walkthrough – Layout Not Found

1. **Smart publish** the database to make sure that both media items and content items exist in the web database by clicking on the **Publish tab** in the ribbon, selecting **Publish site** option and following the wizard
2. In another **browser** navigate to the item you just created (<http://training/holidays>)
3. Sitecore displays an error stating that **the layout for the requested document was not found**
4. Review the instructions that determine how an item should be rendered in the browser

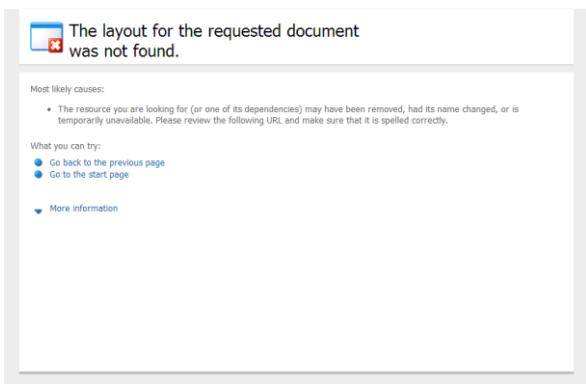


Figure 2-11 Layout not found

Review

Defining data structure

- | | |
|--|---|
| <p>Q What is an item made up of?</p> <ul style="list-style-type: none">✓ Field sections and fields <p>Q All fields have a...</p> <ul style="list-style-type: none">✓ Field type <p>Q Some examples of field types are...</p> <ul style="list-style-type: none">✓ Single-line text, rich text, image, date <p>Q What determines an item's type?</p> <ul style="list-style-type: none">✓ Its data template <p>Q Which interface do you use to create data templates?</p> <ul style="list-style-type: none">✓ Template Manager | <p>Q How do you create an item?</p> <ul style="list-style-type: none">✓ Right-click, Insert from Template <p>Q Where can you find out which data template an item is based on?</p> <ul style="list-style-type: none">✓ Quick Info section on that item <p>Q Why is it important to assign an icon to data templates?</p> <ul style="list-style-type: none">✓ To help authors distinguish between different types of content and differentiate between different types of items in a large content tree |
|--|---|

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 Binding content and presentation

Introduction

Objectives

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

- Create a project environment in Visual Studio
- Configure presentation details in Sitecore
- Bind presentation details to items

Content

Working with Visual Studio

Before we can work with the presentation layer, we need to create a Visual Studio solution to house our files. You already have a pre-made Visual Studio solution on your machine, however we will show you how to set one up



Demo - Creating a project

1. Open **Visual Studio 2012** as **administrator**
2. From the **FILE** menu, click the **New Project....**
3. Select the **Installed/Templates/Visual C#/Web ASP.NET Empty Web Application**
4. Enter **Training** in the **Name** text field
5. Verify the **Create Directory for solution** checkbox is checked and click the **OK** button

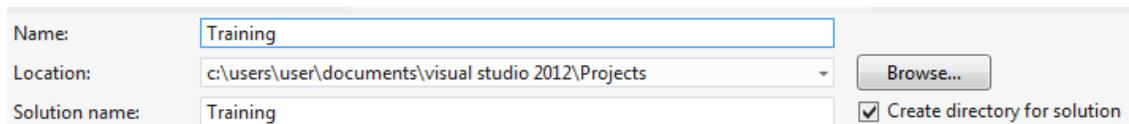


Figure 2-12 Creating a new project folder outside of the webroot

6. Replace the project **web.config** with the one in your **Sitecore web root**, which may be at **C:\inetpub\wwwroot\Training\Website**
7. Create a folder called **App_Config** in your project
8. Copy the **ConnectionString.config** file from your **Sitecore web root**, which may be at **C:\inetpub\wwwroot\Training\Website\App_Config**, into the new **App_Config** folder in your project
9. Create an **empty** folder called **Libraries** in your project
10. Copy **Sitecore.Kernel.dll** from your **Sitecore web root**, which may be at **C:\inetpub\wwwroot\Training\Website\bin**, into the **Libraries** folder
11. Add a reference to the **Sitecore.Kernel.dll** (set Copy Local to **true**) – **DO NOT** reference the version in your web root make sure to **Browse...** for the Libraries folder instance in your Visual Studio 2012 Projects folder



Walkthrough – Create a Visual Studio publish

You will be working in a pre-made training solution – which might be located in:
D:\Users\Admin\Documents\Visual Studio 2012\Projects\Training

We need to create a Visual Studio publish which will save, build and copy files from your Visual Studio solution into the **Sitecore web root**

- For the duration of this course we will be calling a Visual Studio publish – **Visual Studio deploy / deploy** so as not to confuse the terminology with a Sitecore publish

- Locate and **open** your **training solution** which might be located in **D:\Users\Admin\Documents\Visual Studio 2012\Projects\Training**
- Right click on your **Training project** and select **Publish...**

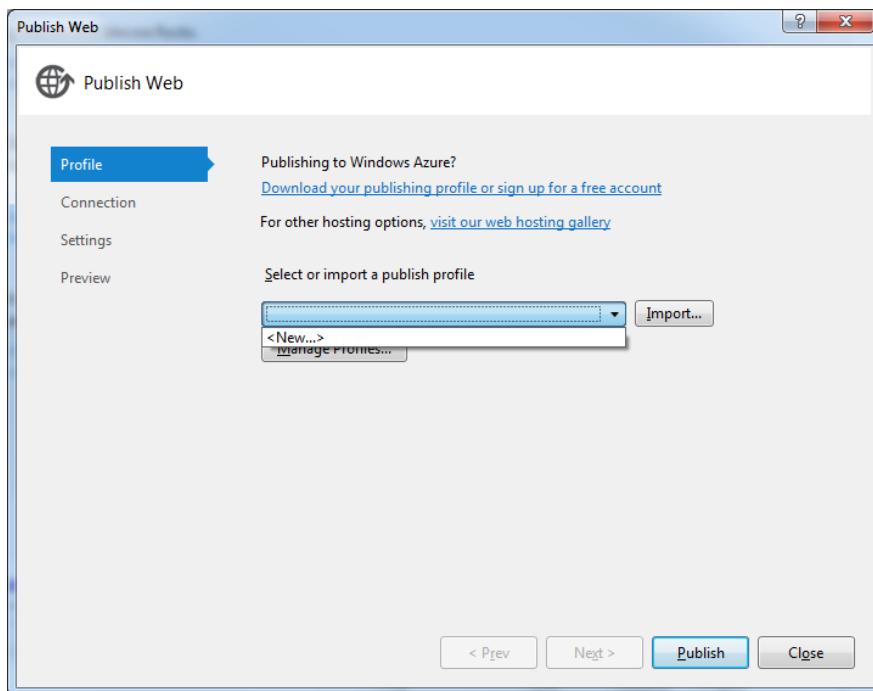


Figure 2-13 Set up the Visual Studio publish profile

- In the dropdown, select **<New...>** and name the publish profile **Training** and click **OK**
- Choose **File System** as your **publishing method**, and click the ellipsis (...) next to **Target location**
- Choose **IIS** in the top left-hand corner and select your **Training** site and click **Open** (if you don't have this option, find the web root folder –e.g. **C:\inetpub\wwwroot\Training\Website**)

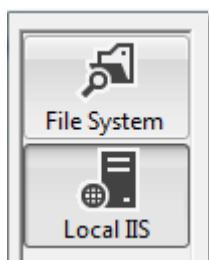


Figure 2-14 Select IIS as the target location

- Click **Next**
- Choose **Debug** in the **Configuration** dropdown

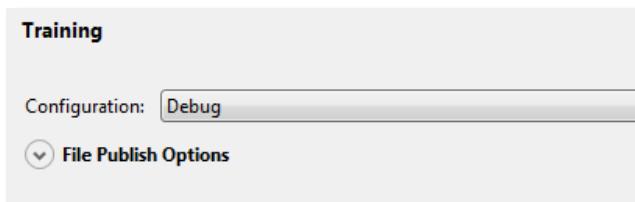


Figure 2-15 Debug configuration

8. Click **Next and Publish**

Walkthrough – Create Web Form and deploy

In the following labs you will:

- Create a web form in Visual Studio
- Deploy your project (copy the files from your local location into the web root)
- Browse to the file URL
- Test that intellisense works

1. Using Visual Studio, right-click on the solution root item and select the **Add > New Item...** option from the resulting menu
2. Add a **.NET Web Form** into your project
3. Name it **Static.aspx**
4. Populate Static.aspx with some static content, for example:

```
<h1>I will be static!</h1>
<p>Content here</p>
```
5. Type **<sc:** to check for **intellisense**



Tip

If intellisense does not work, do a **Clean** on your solution and restart it

6. **Deploy** the Visual Studio **solution** by 1 of 2 options:
 - Right-click on the project, choose Publish, and click publish again (long way)
 - Right-click in the empty space in your Visual Studio toolbar, and enable **Web One Click Publish**

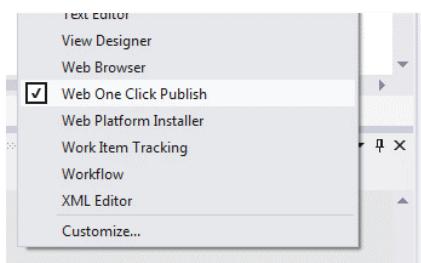


Figure 2-16 Create Visual Studio publish shortcut

- You can then use the shortcut button:

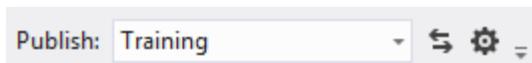


Figure 2-17 Training Visual Studio publishing profile

7. Browse to your new page at <http://training/Static.aspx>

Resolving Static.aspx

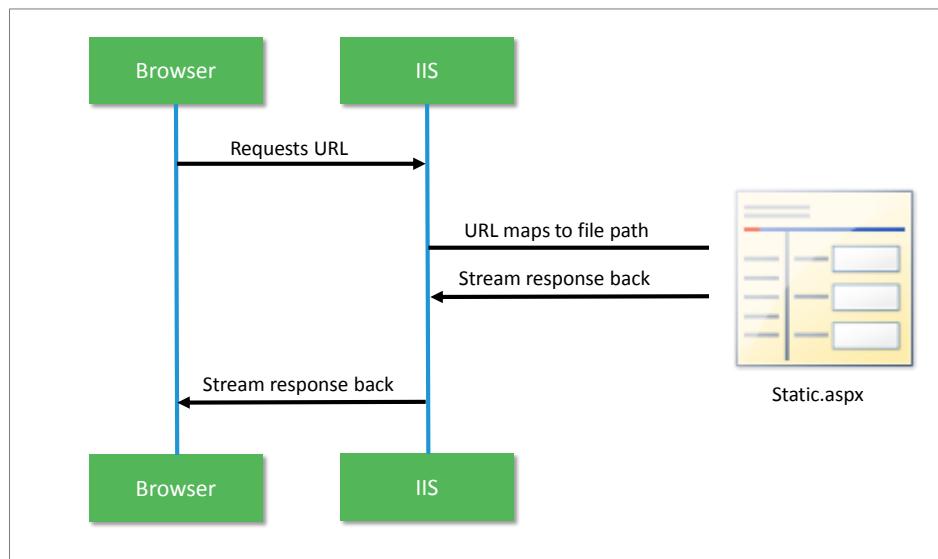


Figure 2-18 Static.aspx



Demo – part 1 – create a layout definition item

1. Keep Static.aspx around for comparison and create a new web form in exactly the same way (in the project root)
2. Name the new web form **Dynamic.aspx**
3. Populate Dynamic.aspx with some **static content**, for example:

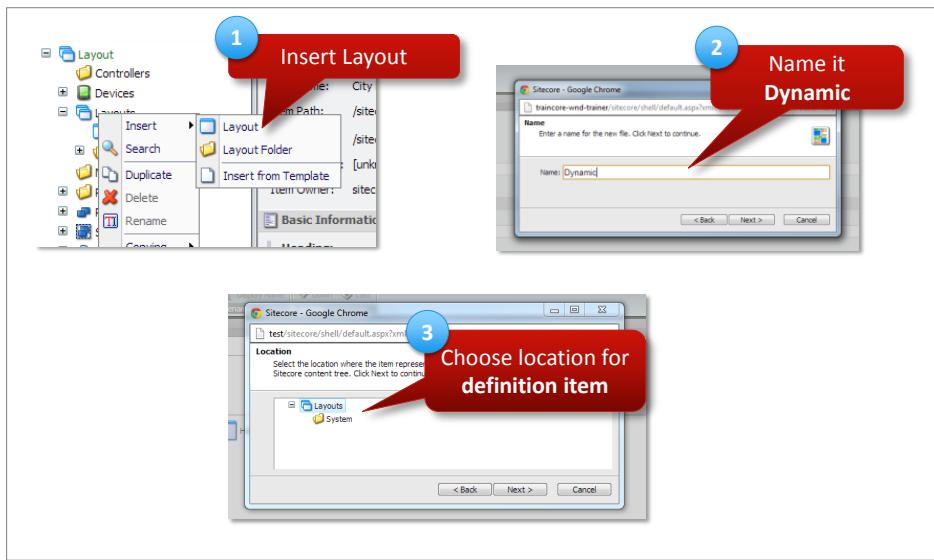
```
<h1>I will be dynamic!</h1>
<p>Content here</p>
```
4. Type in **<sc** to verify that **intellisense** works, if it doesn't do a **Clean** on the solution, you may need to **restart** it as well
5. **Deploy** the Visual Studio solution
6. In the **Content Editor**, right- click on **/sitecore/Layout/Layouts** and select **Layout**
7. Go through the sublayout creation wizard. Specify a **name (Dynamic)**, **location (Layouts)**, and **file location** (layouts folder is fine however – this is redundant and will be overwritten by the file in your Visual Studio solution)
8. Change the **path field** of the Dynamic layout definition item to point to the **Dynamic.aspx** in the **web root**
9. Switch to **Grid Designer** tab and notice the static binding



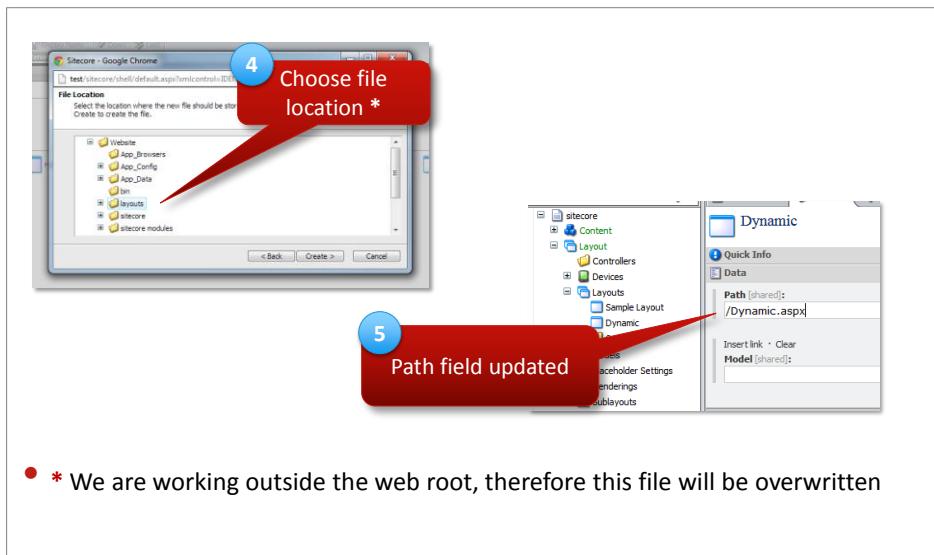
Demo – part 2 – associate layout and content

1. Navigate to the **/holidays** item
2. Click the **Presentation tab > Details command**
3. For the **Default device** click **[no layout specified]**
4. Choose the **Dynamic** layout definition item in the **dropdown**
5. Click the **OK** button on all dialogs
6. **Preview** the page by selecting the **Publish tab** in the ribbon and click **Preview**
7. You should now see the content of **Dynamic.aspx**

Create a layout definition item – steps 1 to 3 of 5



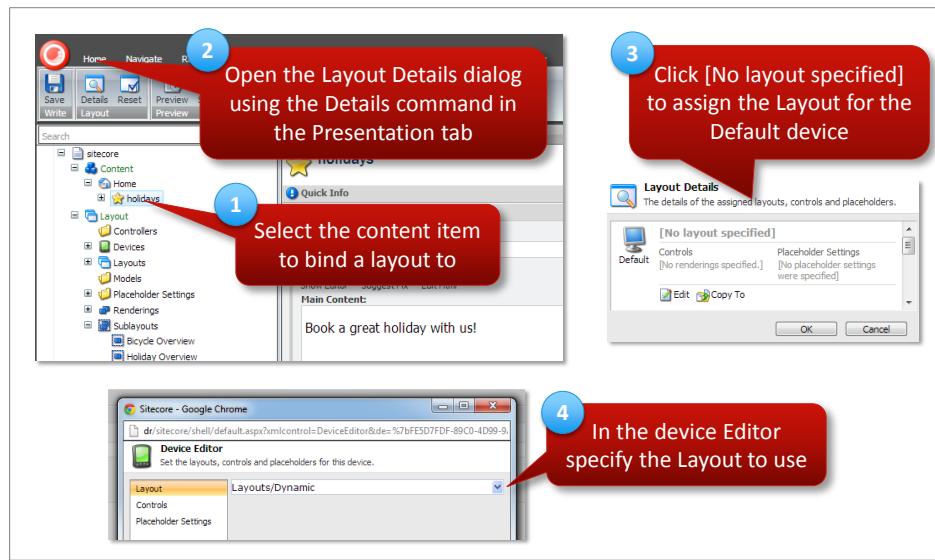
Steps 4 to 5 of 5



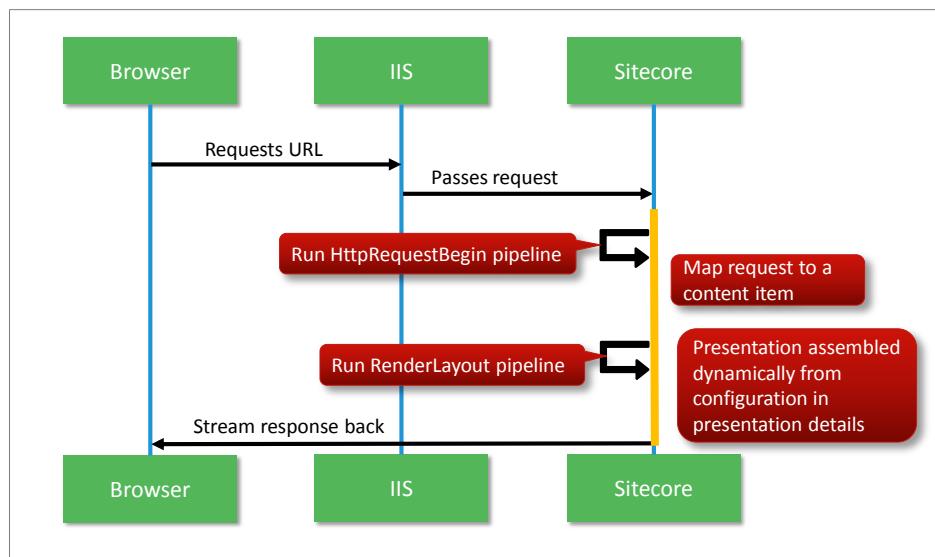
Note

Why did we Insert from Template rather than click the Layout option in the list? Clicking the layout option launches a wizard that creates an .aspx file for you. However, it creates the file in the Sitecore web root, but we are working in a project outside the web root. We will show more efficient ways of doing this later in the training.

Associating layout and content



How Sitecore handles a request



Apply – Topic 2.2 – 15 min



Create a layout for the holidays page

In the following labs you will:

- Create a file on the file system
- Deploy your solution
- Create a layout definition item in Sitecore – by bypassing the wizard because we don't want to create the file on the file system as we already have one
- Bind that definition item to the file on the file system
- Assign that layout item to the holidays item you created earlier
- Finally, you will preview your page

Lab A. Create a web form and a layout definition item

1. In **Visual Studio**, create a new web form called **HolidayListing.aspx**
2. **Deploy** the Visual Studio solution
3. Switch to Sitecore and using the **Content Editor**, navigate to **/sitecore/Layout/Layouts/**
4. **Right-click** on the **layouts** item and click the **Insert > Insert from Template** option in the context menu
5. Click the **Browse** tab
6. Create an item based on **/Templates/System/Layout/Layout**, and name it **Holiday Listing**

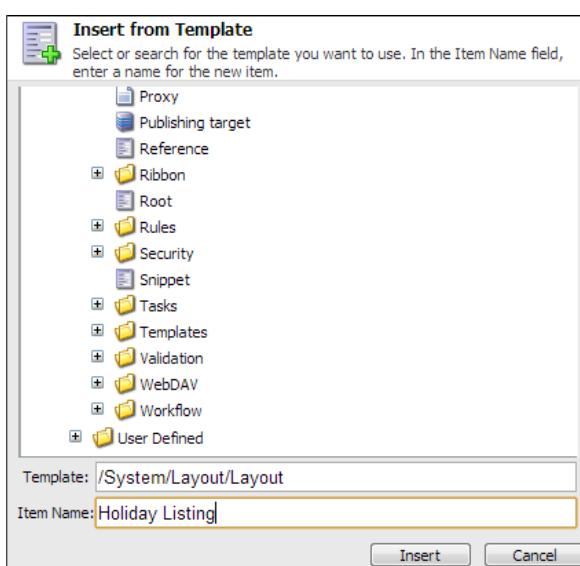


Figure 2-19 Create Holiday Listing layout definition item

7. Click the **Content** tab and in the **Path** field, type the path to the **/HolidayListing.aspx** you created in a previous lab, then **save** your work

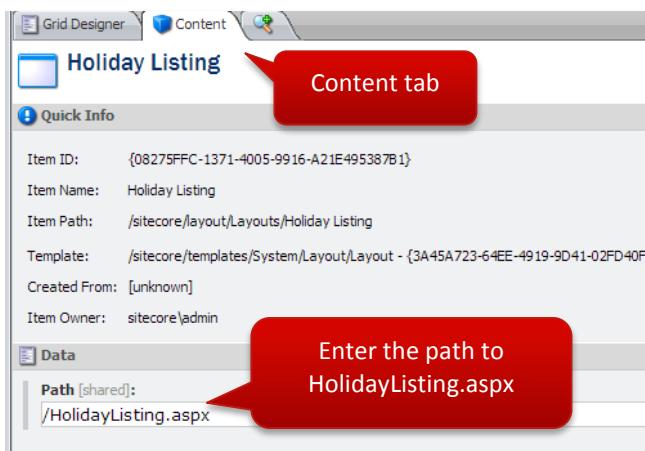


Figure 2-20 Enter the path to the .aspx page

8. You can ensure that you have correctly configured the layout path with the **Grid designer** tab, for now we have an empty <div>



Figure 2-21 Grid Designer

Lab B. Add HTML to the HolidayListing.aspx

1. Some static HTML has been provided in the **student resource folder** (*WND Labs > Module 2 > Topic 2.2 > Lab B > HTML > campaign-page-landing.html*)
2. **Copy** the static **HTML** from the above location **into** the **HolidayListing.aspx**
3. Make sure you **merge** the **HTML** to **HolidayListing.aspx**; do **not overwrite** the **existing page directive** at the top of the file
4. From the same student resource folder copy the accompanying **/css** and **/img** folders into your solution

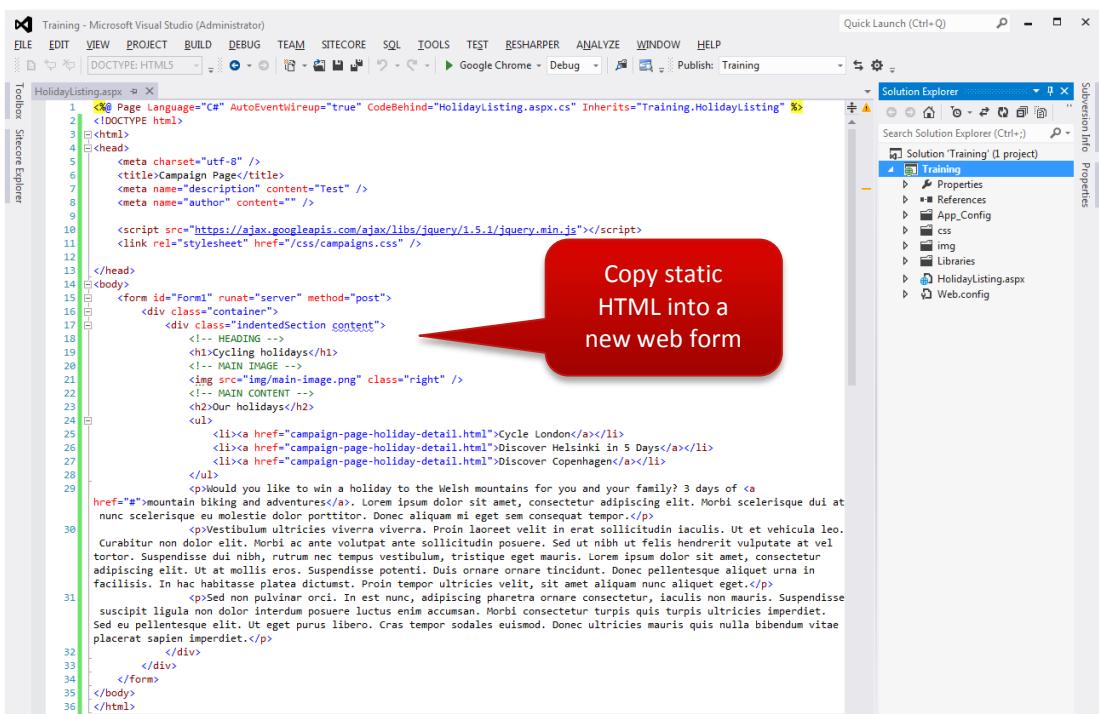


Figure 2-22 Resulting HolidayListing.aspx and Training solution

5. **Deploy** your solution

Lab C. Bind the Holiday Listing layout definition item to the holidays item

1. In the **Content Editor**, navigate to the **holidays** item you created under `/sitecore/content/home/`
2. In the **ribbon**, select the **Presentation** tab and click the **Details** command



Figure 2-23 Presentation details command

3. Click **[No layout specified]** next to the **Default** device

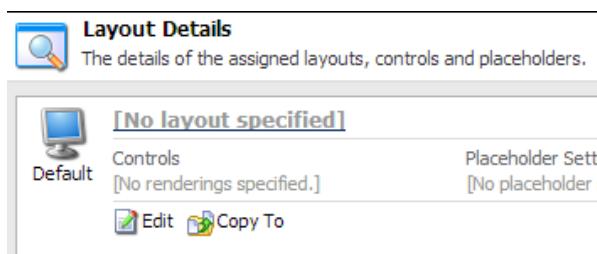


Figure 2-24 Open Device Editor

4. Select the **Holiday Listing** item from the drop down

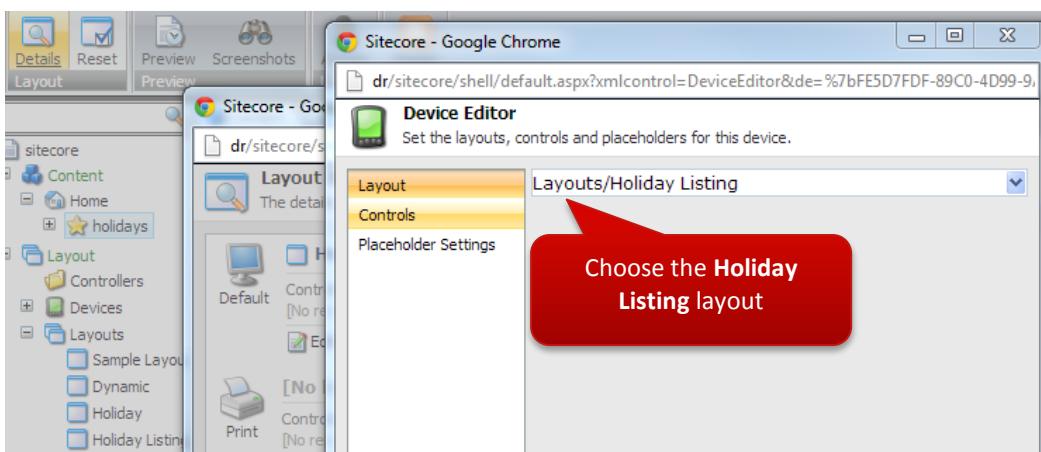
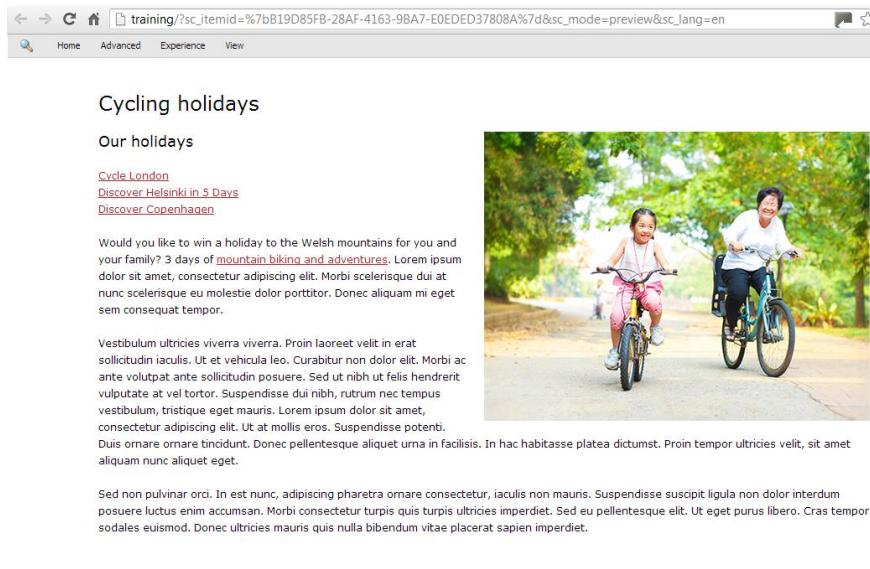


Figure 2-25 Choose the Holiday Listing layout

5. Click the **OK** button in all windows and **save** the holidays item
6. **Preview** the <http://training/holidays> item (in the ribbon, **Publish** tab > **Preview**) and confirm that the dummy content you entered into HolidayListing.aspx is displayed



The screenshot shows a web browser window displaying a Sitecore website. The URL in the address bar is `training/?sc_itemid=%7bB19D85FB-28AF-4163-9BA7-E0EDED37808A%7d&sc_mode=preview&sc_lang=en`. The page title is "Cycling holidays". A sidebar on the left contains links: "Our holidays", "Cycle London", "Discover Helsinki in 5 Days", and "Discover Copenhagen". The main content area contains placeholder text about winning a holiday to the Welsh mountains, followed by a large image of a young girl and an older woman riding bicycles on a path. Below the image is more placeholder text.

2-26 Finished page

Review

Binding content and presentation

- | | |
|---|--|
| <p>Q In the context of Sitecore, a web form is known as a...</p> <p>✓ Layout</p> | <p>Q How do you configure an item's presentation details?</p> <p>✓ Select item, Presentation tab > Details</p> |
| <p>Q How do you make Sitecore aware of a layout on the file system?</p> <p>✓ A layout definition item that points to the path of the layout.</p> | <p>Q State one advantage of working in a Visual Studio project outside the Sitecore web root.</p> <p>✓ Your solution is portable.</p> |
| <p>Q How many layouts can you assign to a single item?</p> <p>✓ One per device</p> | |

Topic 2.3 Rendering Content

Introduction

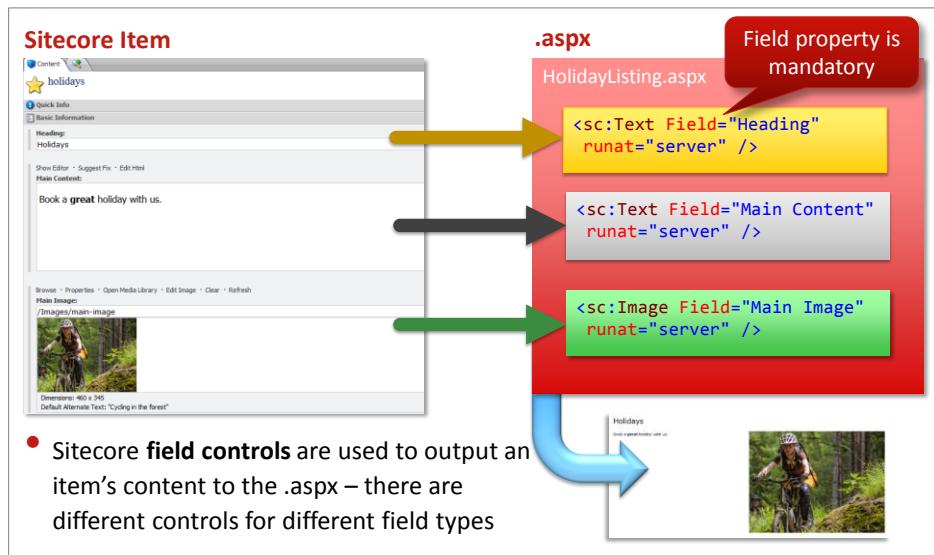
Objectives

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

- Render the content of an item to the browser
- Statically componentize a rendered page

Content

How do you output content?



Sitecore control parameters

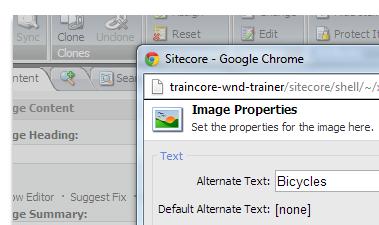
Depending on the control, Sitecore offers a different set of options

- Dynamic resizing of images using **MaxWidth** and **MaxHeight** (caches an image of that size on the server):

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

Remember! Your settings in code trumps author settings

- If you set an image **Alt** text in code, an author's changes in the Content Editor will **not** take effect – make sure this is not going to be interpreted as a bug





Walkthrough – Outputting Content

In this walkthrough, we will:

- Replace the static content in the .aspx with the Sitecore controls that will output Sitecore stored content
- Deploy
- Make edits in the Page Editor

Make sure that the controls are added between the `<form>` tags as shown in [2-27 HTML after replacing content with Sitecore controls](#).

- In Visual Studio, open the `HolidayListing.aspx`
- Replace the **image tag** `` with a **Sitecore image control** `<sc:Image>` that will output the **Main Image** field (**note** the `CssClass` property):



Code Sample – Outputting the Sitecore image field

```
<sc:Image Field="Main Image" CssClass="right" runat="server" />
```

- Replace the **heading** `<!-- HEADING -->` and **content** `<!-- MAIN CONTENT -->` with **Sitecore text controls** `<sc:Text>`
- Set the **Field** property to *Heading* and *Main Content* respectively:



Code Sample – Outputting Sitecore text fields

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

- Make sure that you do not have any dummy content remaining. Your file should look something like this:

```
<form id="Form1" runat="server" method="post">  
    <div class="container">  
        <div class="indentedSection content">  
            <!-- HEADING -->  
            <h1><sc:Text Field="Heading" CssClass="right" runat="server" /></h1>  
            <!-- MAIN IMAGE -->  
            <sc:Image Field="Main Image" CssClass="right" runat="server" />  
            <!-- MAIN CONTENT -->  
            <sc:Text Field="Main Content" runat="server" />  
        </div>  
    </div>  
</form>
```

2-27 HTML after replacing content with Sitecore controls

- Deploy the Training solution and preview the <http://training/holidays> page through the **Content Editor** by selecting the **holidays** item, in the ribbon select the **Publish tab** and click **Preview**
- The page should now display the content you originally entered into the Sitecore Holidays item.



Figure 2-28 Published as an item

8. In the ribbon switch to **editing mode** by clicking on the **Home** tab and click **Edit**



Figure 2-29 Switch to edit mode in Page Editor

9. **Edit the Heading field**

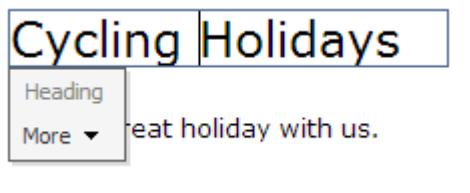


Figure 2-30 Edit text field

10. Now that you are in **Page Editor editing mode** you are able to **edit fields** that are being **rendered with Sitecore controls** i.e. `<sc:text />`, `<sc:image>` etc. we will cover mode field types and Sitecore controls at a later stage

Componentize into user controls as usual

As with any software project, the **D(on't) R(epeat) Y(ourself)** principle applies – separate your pages into re-useable components:

- Just as you would in a regular .NET project, **componentize functionality** so that it can be re-used

Holiday Listing
Componentized

Holiday Overview
Componentized



Walkthrough – Componentize HolidayListing.aspx

In Visual Studio, create a new user control called Introduction.ascx and bind it to HolidayListing.aspx

1. In your **Visual Studio** Training solution, create a **user control** in the project root with the other files and name it ***Introduction.ascx***
2. **Cut** the static **HTML** with Sitecore user controls **from HolidayListing.aspx** and **paste** it into **Introduction.ascx** (cut this div `<div class="indentedSection content">` and everything inside it)



Code Sample – HTML sample

```
<div class="container">
    <!-- Start Cut -->
    <!-- .. static HTML and user controls .. -->
    <!-- End Cut -->
</div>
```

3. Make sure there is **no HTML** inside the `<div class="container">` in HolidayListing.aspx, this is where you will be including Introduction.ascx
4. You can register your user control as you normally would using a `<%@ Register %>` directive at the top of the HolidayListing.aspx, however we are going to use a Sitecore control to do it:



Code Sample – Statically binding a user control

```
<sc:Sublayout Path="/Introduction.ascx" Runat="server" />
```

- Any user control (.ascx) can be included on a page in this way. We return to sublayouts later on in the module. Your HolidayListing.aspx should look something like this:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="HolidayListing.aspx.cs" Inherits="Training.HolidayListing" %>

<!DOCTYPE html>
<html>
    <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">
                <sc:Sublayout ID="Sublayout1" Path="/Introduction.ascx" Runat="server" />
            </div>
        </form>
    </body>
</html>
```

Figure 2-31 HolidayListing.aspx code

5. **Deploy** the Training solution and **preview** your **holidays** item through the **Content Editor**
6. Although you have componentized your page, the **final output** should remain **unchanged**

Apply – Topic 2.3 – 15 min



Create a data template and some items based on It

In the following labs, you will:

- Create a Holiday data template with a number of fields
- Create a number of items based on the Holiday data template and populate them with data

Lab A. Create a data template

1. Log in to the Sitecore **desktop** as **admin** (password: **b**)
2. Click on the **Sitecore** start button and open the **Template Manager**
3. Select the **User Defined** folder in the Folder tab click **New Template**

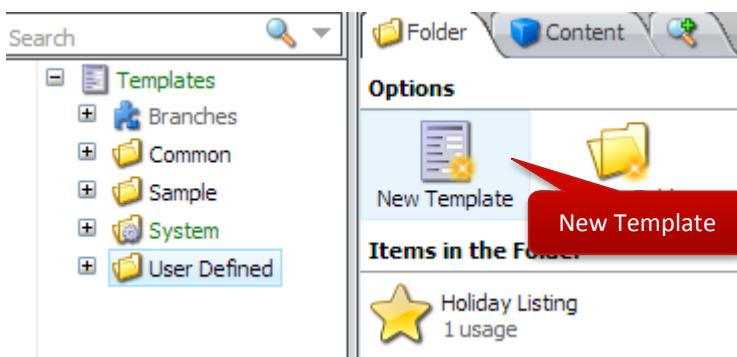


Figure 2-32 Create a new template

4. Name this template **Holiday**
5. Create the **field sections** (**Basic Information** and **Holiday Information**) and fields defined in the table below:

| Field Name | Field Type |
|----------------------------|------------------|
| Basic Information | |
| Heading | Single-line text |
| Main Content | Rich Text |
| Main Image | Image |
| Holiday Information | |
| Price per person | Single-line text |
| Start date | Date |

6. **Save**
7. Assign the globe icon to the data template with the **Icon command** in the **Configure** tab in the **ribbon** (You will find the globe icon in the Networking section. More icons are available by clicking the **More Icons** option)



Figure 2-33 Assign an icon

Lab B. Add new items based on the holiday data template

1. Open the **Content Editor** from the **Desktop** by using the **start** button
2. Select the **sitecore/content/Home/holidays** item you created earlier, in the **Home** tab in the **ribbon** click **Insert from Template**

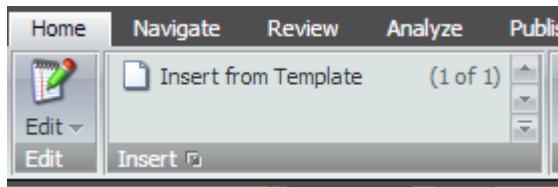


Figure 2-34 Insert from template in the ribbon

3. Insert **three items** based on the **Holiday** data template – name them **Explore Holland**, **Discover the Cotswolds** and **Visit Thailand**, see **Figure 2-7 Creating a new item from a data template**
4. Use the data in the following table as a guide to completing your items (choose any image from the Media Library), if pressed for time **fill in at least 2** of the 3 items created

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

| Discover the Cotswolds | |
|------------------------|-----------------------------------|
| Basic Information | |
| Heading | Discover The Cotswolds |
| Main Content | Discover The Cotswolds |
| Main Image | <i>Any from the media library</i> |
| Holiday Information | |
| Price per person | \$800 (or use any value) |
| Start date | 5/1/2013 (or enter any date) |

| Visit Thailand | |
|---------------------|-----------------------------------|
| Basic Information | |
| Heading | Visit Thailand |
| Main Content | Visit Thailand |
| Main Image | <i>Any from the media library</i> |
| Holiday Information | |
| Price per person | \$1000 (or use any value) |
| Start date | 6/12/2013 (or enter any date) |

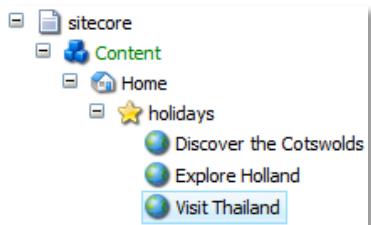


Figure 2-35 Finished items

Review

Rendering content

- | | |
|--|--|
| <p>Q Name three specialized Sitecore field controls</p> <p>✓ <sc:Text />, <sc:Date>, <sc:Image /></p> <p>Q Name the one mandatory property that must be specified on the above controls</p> <p>✓ <i>Field (note: on <sc:FieldRenderer />, the equivalent is FieldName)</i></p> | <p>Q What control can you use to statically include a .NET user control in an .aspx?</p> <p>✓ <sc:Sublayout /></p> <p>Q Why is it important to componentize functionality?</p> <p>✓ <i>To reduce duplication</i></p> |
|--|--|

Extend

Presentation Component API Cookbook

<http://sdn.sitecore.net/Reference/Sitecore%206/Presentation%20Component%20API%20Cookbook.aspx>

Topic 2.4 Dynamic binding

Introduction

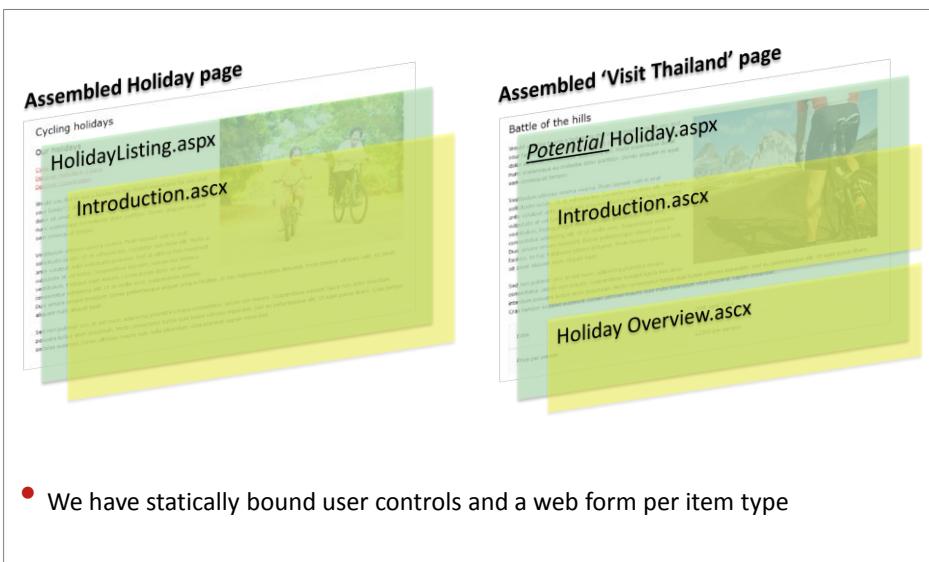
Objectives

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

- Dynamically assemble a page from components
- Name four different types of components
- Know how to use a single layout for an entire site

Content

Potential setup

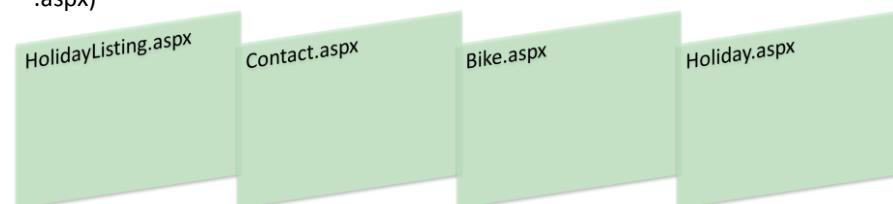


Knowledge Check

What issues do you foresee with the current setup?

The problem

- Every new page type would require **its own layout** (and therefore a separate .aspx)



- Potential **duplication of functionality** – e.g. headers, footers, <head> (*general .NET solutions get around this with master pages and placeholders*)
- If you want to re-order controls or change their properties, you have to do it **on each individual web form**
- Any **changes** to page structure have to be done **by a developer in Visual Studio**

Sitecore's solution – using placeholders

Holiday.aspx

<div>

Battle of the hills

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking and adventures](#). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi sollempne dum at nunc consequatur eu molestie dolor porttitor. Donec aliquam mi eget sem consequatur tempus.

Vestibulum ultrices viverra viverra. Proin laoreet velit in erat sollicitudin laus. Ut et velutina leo. Curabitur non dolor etc. Morbi ac ante volutpat ante sollicitudin posse. Sed ut nibh ut felis hendrerit vestibulum. Tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi sollempne dum at nunc consequatur eu molestie dolor porttitor. Donec aliquam mi eget sem consequatur tempus.

Duis ornare ornare tristique. Donec pellentesque aliquet urna in feaces. In hac habitasse platea dictumst. Proin tempus ultrices velit, sit amet aliquam nunc aliquet eu.

Sed non pulvinar eros. In est manu, adipiscimus pharetra ornare.

[Introduction.aspx](#)



Sed non pulvinar eros. In est manu, adipiscimus pharetra ornare.

[Introduction.aspx](#)

Etiam ultrices viverra viverra. Proin laoreet velit in erat sollicitudin laus. Ut et velutina leo. Curabitur non dolor etc. Morbi ac ante volutpat ante sollicitudin posse. Sed ut nibh ut felis hendrerit vestibulum. Tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi sollempne dum at nunc consequatur eu molestie dolor porttitor. Donec aliquam mi eget sem consequatur tempus.

Duis ornare ornare tristique. Donec pellentesque aliquet urna in feaces. In hac habitasse platea dictumst. Proin tempus ultrices velit, sit amet aliquam nunc aliquet eu.

Sed non pulvinar eros. In est manu, adipiscimus pharetra ornare.

[Introduction.aspx](#)

</div>

HolidayListing.aspx

<div>

Placeholder key="Main"

Battle of the hills

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking and adventures](#). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi sollempne dum at nunc consequatur eu molestie dolor porttitor. Donec aliquam mi eget sem consequatur tempus.

Vestibulum ultrices viverra viverra. Proin laoreet velit in erat sollicitudin laus. Ut et velutina leo. Curabitur non dolor etc. Morbi ac ante volutpat ante sollicitudin posse. Sed ut nibh ut felis hendrerit vestibulum. Tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi sollempne dum at nunc consequatur eu molestie dolor porttitor. Donec aliquam mi eget sem consequatur tempus.

Duis ornare ornare tristique. Donec pellentesque aliquet urna in feaces. In hac habitasse platea dictumst. Proin tempus ultrices velit, sit amet aliquam nunc aliquet eu.

Sed non pulvinar eros. In est manu, adipiscimus pharetra ornare.

[Introduction.aspx](#)

Etiam ultrices viverra viverra. Proin laoreet velit in erat sollicitudin laus. Ut et velutina leo. Curabitur non dolor etc. Morbi ac ante volutpat ante sollicitudin posse. Sed ut nibh ut felis hendrerit vestibulum. Tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi sollempne dum at nunc consequatur eu molestie dolor porttitor. Donec aliquam mi eget sem consequatur tempus.

Duis ornare ornare tristique. Donec pellentesque aliquet urna in feaces. In hac habitasse platea dictumst. Proin tempus ultrices velit, sit amet aliquam nunc aliquet eu.

Sed non pulvinar eros. In est manu, adipiscimus pharetra ornare.

[Introduction.aspx](#)

</div>

2-37 Static Binding

- Instead of statically binding components (`<sc:Sublayout />`), declare a **placeholder** with a unique key and **dynamically bind reusable components**

What are placeholders?

- Placeholders** are Sitecore web controls
- They are identified by an attribute called **Key**
- They can be added to an .aspx or .ascx, and define a point where components can be **dynamically bound**
- Placeholders are added directly into code:

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

Layout

<div>

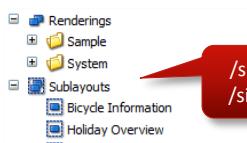
Placeholder (**key =xxx**)

</div>

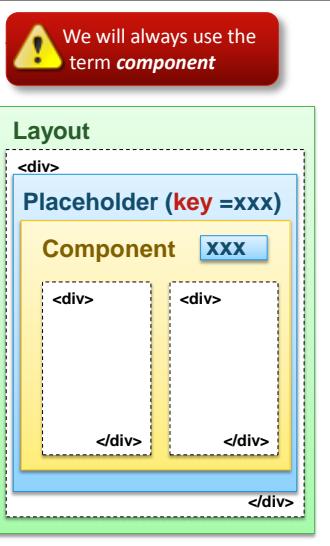
What is a component?

We deliberately use the word **component** to describe a re-useable chunk of functionality:

- Pages are assembled from discrete units of functionality – collectively known as **components**
- A **component** can be a:
 - Sublayout** (Sitecore's term for **user control**)
 - XSLT Rendering**
 - Web control**
 - MVC Rendering**
- Like layouts, components consist of a **definition item in Sitecore** and a **file on the file system**



/sitecore/Layout/Sublayouts
/sitecore/Layout/Renderings



Important

Throughout the development of Sitecore the words component, rendering and control are used interchangeably; however, we will always use the term **component**



Demo – part 1 – create a web form with a placeholder

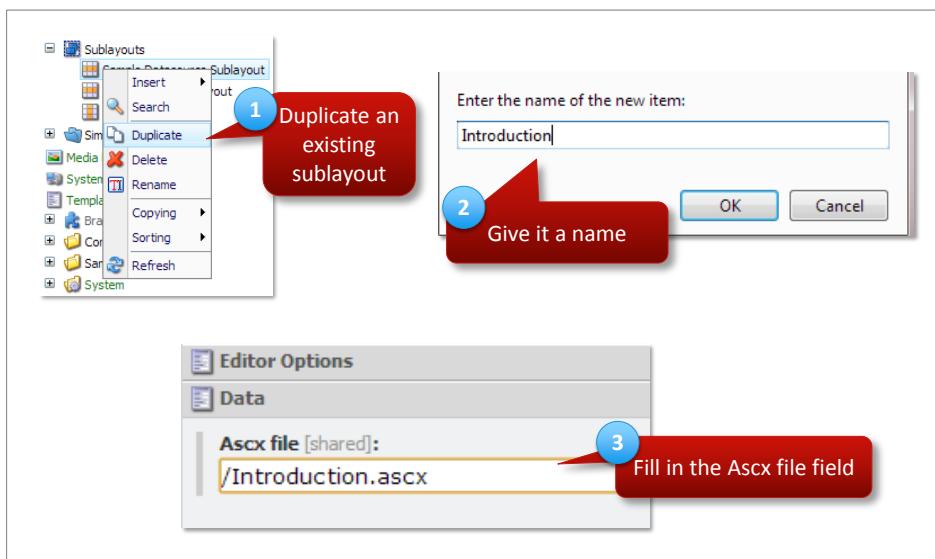
1. Create a new web form called **Main.aspx**
2. **Copy** all **HTML** and code from **Holiday Listing.aspx** into **Main.aspx**
3. **Replace** the **statically bound component** with a Sitecore **placeholder** and set its key to **main**
4. **Deploy** the solution



Demo – part 2 – create definition items and bind content to presentation

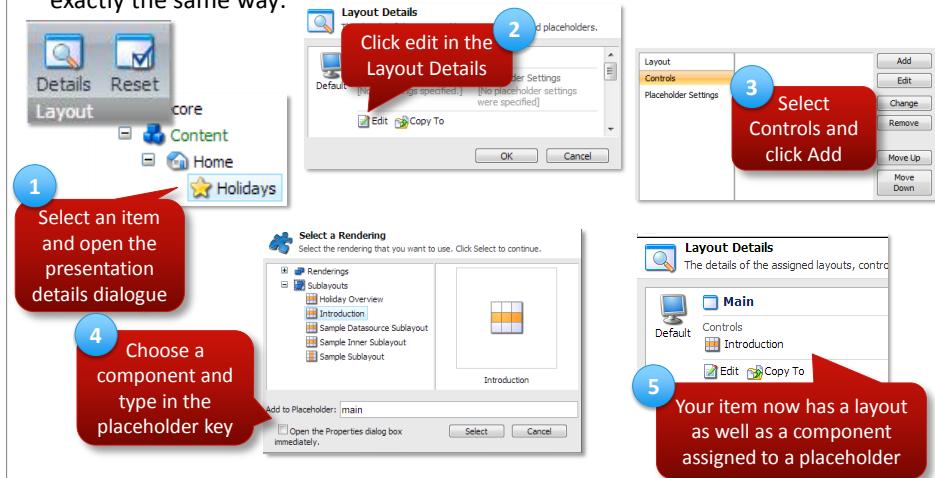
1. In the **Content Editor** create a **layout definition item** called **Main** for the **Main.aspx** – **duplicate** an existing layout
2. Make sure you fill in the Path field on the item **/Main.aspx**
3. **Save**
4. Switch to the **Grid Designer** tab to see if the **binding** has happened
5. Create another **sublayout definition item** called **Introduction**, by **duplicating** an existing definition item
6. Make sure you fill in the **Ascx file** field in the data field section on the item **/Introduction.ascx**
7. **Save**
8. Switch to the **Grid Designer** tab to see if the **binding** has happened
9. Navigate to the **holidays** item and click the **Presentation Details** button
10. Click the **Edit** button next to the Default device and select the **Main** layout option in the drop-down menu
11. Select the **Controls** option in the left-hand menu and click the **Add** option
12. Find and select the newly created **Introduction** component and bind it to the **main** placeholder
13. **Preview** the **holidays** item to see the image, heading and text that was there when we used the Dynamic layout
14. Show **design mode** that you can now select the component

Creating a component definition item



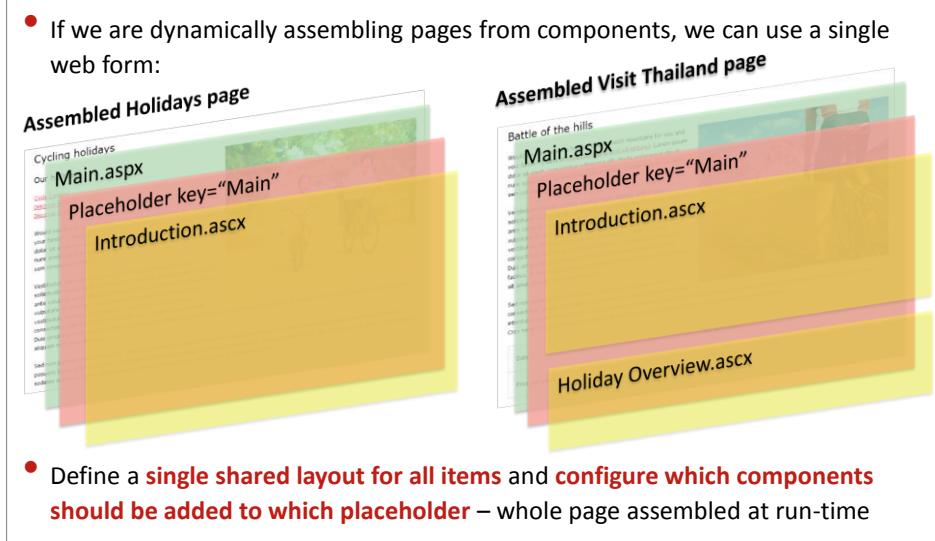
Dynamically bind component

- All components, regardless of type, are added to an item's presentation details in exactly the same way:



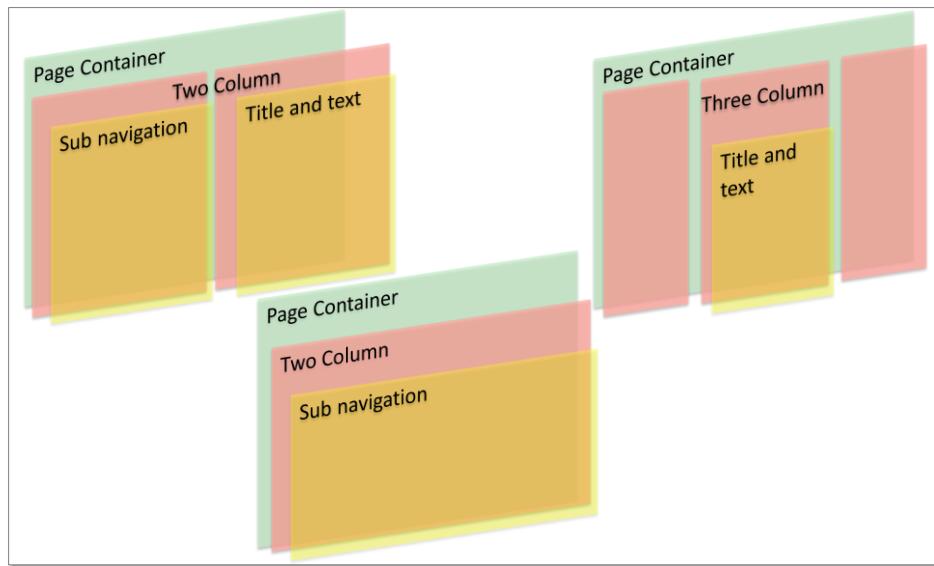
One layout for all items

- If we are dynamically assembling pages from components, we can use a single web form:



**Best Practice**

You would typically have **one layout per site, per device**. (i.e., one layout for web, one layout for mobile, one layout for print)

A more realistic use of dynamic binding**Apply – Topic 2.4 – 30 min****Create and bind presentation components**

In the following labs we will be refactoring our implementation to use one layout for the Default device. You will be creating a new layout and a new sublayout, using a placeholder control for dynamic binding, as well as binding all the presentation components to your items

Lab A. Create a new layout with placeholder

1. In **Visual Studio**, create a layout called **Main.aspx**
2. Copy all the HTML from the **HolidayListing.aspx** and **paste** it into **Main.aspx** (as before do not copy or replace the page directives)
3. In **Main.aspx**, **replace** the single **statically bound** Sitecore component (**Introduction.ascx**) with a placeholder and give it a key of **main** – see below for the syntax

**Code Sample – Statically binding a user control**

```
<sc:PlaceHolder Key="main" Runat="server" />
```

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Main.aspx.cs" Inherits="Training.Main" %>
<!DOCTYPE html>
<html>
<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">
            <sc:Placeholder Key="main" runat="server" />
        </div>
    </form>
</body>
</html>

```

Figure 2-38 Final Main.aspx

4. Save your solution

Lab B. Create a new user control called Holiday Overview

Holiday Overview.ascx will contain a table showing details about the holiday. We will be putting it into a separate user control because you may want to re-use it to display details somewhere other than on holiday items.

1. In Visual Studio, create a new user control called **HolidayOverview.ascx**
2. Copy the contents from the holiday-overview file in the student resource folder (*WND Labs > Module 2 > Topic 2.4 > HTML > holiday-overview.html*) to be used in the Holiday Overview control
3. Paste this static HTML into the **Holiday Overview.ascx** file under the page directive

Lab C. Output Sitecore field values

Replace the static values in the table cells with dynamic content using Sitecore controls

- Use Sitecore controls to output the fields that you defined in *Topic 2.3 > Lab A* into the Holiday Overview user control

1. For the **Start date** and **Price per person** you can use a Sitecore **Date** and Sitecore **Text** control respectively, see below for code sample



Code Sample – Outputting Sitecore fields

```

<sc:Date Field="Start date" Runat="server" />
<sc:Text Field="Price per person" Runat="server" />

```

```

<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="HolidayOverview.ascx.cs"
Inherits="Training.HolidayOverview" %>



| Date             | <sc:Date Field="Start date" Runat="server" />                  |
|------------------|----------------------------------------------------------------|
| Price per person | <sc:Text ID="Text1" Field="Price per person" Runat="server" /> |


```

Figure 2-39 Final HTML for HolidayOverview.ascx

2. Deploy your solution

Lab D. Create layout and sublayout definition Item

To make Sitecore aware of the aspx and ascx just created, you need to create definition items for them in Sitecore

- Layout definition item

1. Create a new layout definition item called **Main** (use the **Content Editor**), make sure the path points to **/Main.aspx** and **Save**



Tip

You can **duplicate an existing definition item** to save time, e.g. right-click on the *Holiday Listing* layout definition item and select **Duplicate**, call the item **Main**
Remember to change the path field to Main.aspx

2. Call the layout definition **Main**, and point the Path field to the **/Main.aspx** you created earlier
3. **Save** your work and check the **Grid Designer** to make sure the binding is correct

- Sublayout definition item

1. Using the **Content Editor**, create a **sublayout definition item** by **duplicating** an existing sublayout under **/sitecore/Layout/Sublayouts/** and name it **Introduction**

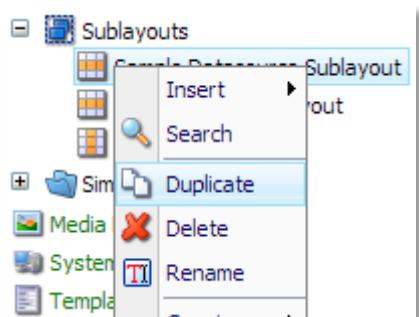


Figure 2-40 Duplicate an item

2. In the **Content** tab look for the **Ascx file** field in the Data field section, type the path to the corresponding user control you created, i.e. **/Introduction.ascx**
3. Create a sublayout definition item for **Holiday Overview** in the same way, and set the **Ascx file** path to the appropriate user control, i.e. **/Holiday Overview.ascx**
4. **Save** and confirm in the **Grid Designer** the binding

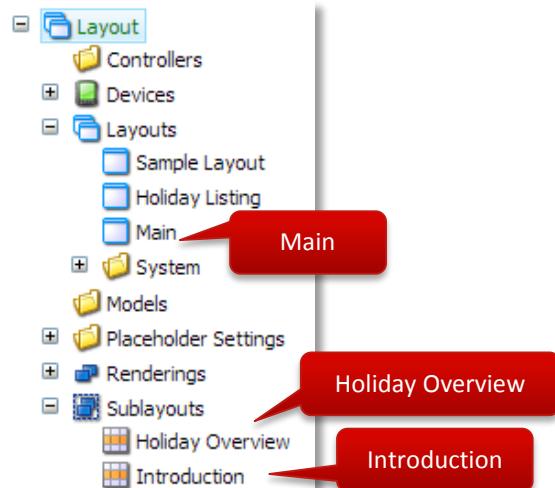


Figure 2-41 Resulting layout and sublayout definition items

Lab E. Assign the presentation to all content items

For the **holidays** items, change the **layout** to **Main** and assign **Introduction** to the **main** placeholder

1. Select the item
2. Click the **Presentation > Details** command
3. Click the **Edit** option next to the Default device
4. Select the **Main** option in the layout dropdown
5. **Select the Controls** option in the left-hand menu and click the **Add** option
6. Select the **Introduction** sublayout and assign it to the **main** placeholder and click select, **OK** and **OK** again

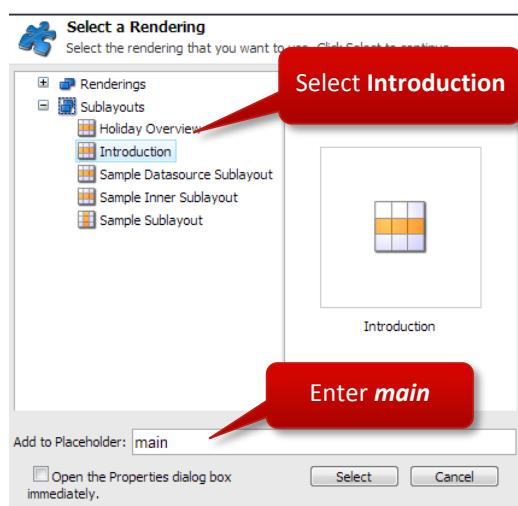


Figure 2-42 Inserting a sublayout and adding to a placeholder

7. **Preview** the **holidays** item and notice that the assembled page should look exactly like **HolidayListing.aspx**

For the children items for holidays i.e. **Discover the Cotswold**, **Explore Holland** and **Visit Thailand** you need to change the **layout** to **Main** as well as assign **Introduction** and **Holiday Overview** to the **main** placeholder

8. Select the child items of /holidays, **one at a time**
9. Click the **Presentation > Details** command
10. Click the **Edit** option next to the Default device
11. Select the **Main** option in the layout dropdown
12. **Select the Controls** option in the left-hand menu and click the **Add** option
13. Select the **Introduction** sublayout and assign it to the **main** placeholder and click **select**
14. Click **Add** again and select the **Holiday Overview** sublayout and assign it to the **main** placeholder and click **select, OK** and **OK** again
15. Click the **OK** button and **preview** the **child items** and notice that the assembled page should look exactly like **Holiday.aspx**

Lab F. Make minor presentation changes

Notice that the pages look the same as before, however, not only have we simplified the solution to use only **one layout**, we have also added new functionality. For instance, we can easily change the order of the components without touching the code, this only becomes relevant when multiple controls are inserted into the same placeholder

16. In the **Page Editor, Editing mode**
17. Select the **Advanced** tab in the ribbon and click **Details**



Figure 2-43 View presentation details through the Page Editor

18. Click the **Edit** option next to the Default device.

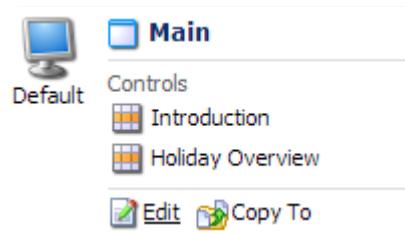


Figure 2-44 Edit presentation details

19. Select the **Controls** tab and choose the **Introduction** item

20. Click the **Move Down** option so that it will appear below the **Holiday Overview**

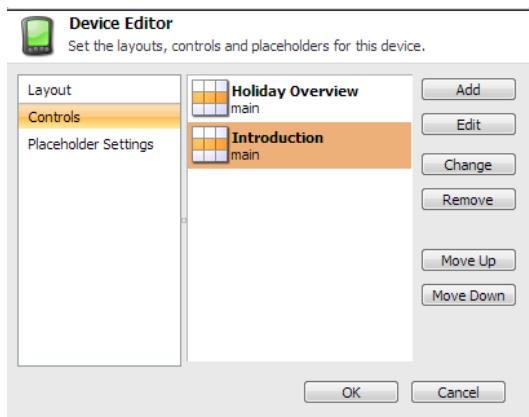


Figure 2-45 Final Result

21. Click the **OK** buttons in all windows and **save**
 22. Notice that the table has now moved above the introduction text

Review

Dynamic binding

| | |
|---|---|
| <p>Q Name the Sitecore control that allows for dynamic binding of components</p> <p>✓ <i>Placeholder</i></p> | <p>Q A component consists of a file on the file system and...</p> <p>✓ <i>A definition item in Sitecore</i></p> |
| <p>Q What property of a Sitecore placeholder must be specified?</p> <p>✓ <i>Key</i></p> | <p>Q How do you bind a component to a particular placeholder on an item?</p> <p>✓ <i>Presentation details – select component and type in placeholder</i></p> |
| <p>Q Name 4 different types of Sitecore components</p> <p>✓ <i>Sublayout, MVC rendering, XSLT rendering, web control</i></p> | <p>Q What is a good candidate for static binding?</p> <p>✓ <i>Header, footer – aspects of a page that are identical throughout the site.</i></p> |
| <p>Q What type of file is a sublayout?</p> <p>✓ <i>.ascx</i></p> | |

Extend

Presentation Component Reference

<http://sdn.sitecore.net/Reference/Sitecore%207/Presentation%20Component%20Reference.aspx>

Topic 2.5 Sitecore Rocks

Introduction

Objectives

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

- Describe some features in Sitecore Rocks
- Explain what Sitecore Rocks allows developers to do with .NET controls

Content

Sitecore Rocks

- Sitecore Rocks enhances Visual Studio with features for developers working with the Sitecore CMS



Important

Note: The majority of the Sitecore Rocks exercises in this course are based on Sitecore Rocks version **0.7.16.9**



Demo – Overview of Rocks

1. Connect Rocks to a Sitecore instance by right-clicking on your project and select **Sitecore > Connect** from the context menu
2. Select an existing connection, otherwise create a new one:
 - Data Provider: **Hard Rock Web Service**
 - Host Name: **Training** (your instance)
 - User Name: **sitecore\admin**
 - Password: **b**
 - Location: **C:\inetpub\wwwroot\Training\Website** (your website root)
3. Open Sitecore Explorer and show the items in the tree, as well as the Web Site folder etc.
4. Create a web form / user control and using rocks create the definition item in Sitecore
5. Show that you can browse to your site
6. Use command to browse the site and publish

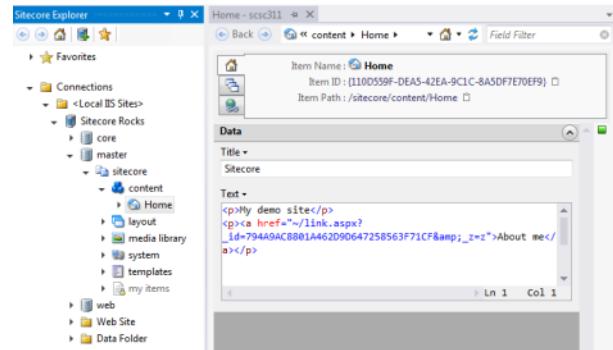


Figure 2-46 Sitecore Rocks installed in Visual Studio

Main Features

- **Sitecore Project Management**—connects a Visual Studio project to a Sitecore website and allows you to create Sitecore items from .aspx, .ascx and .xsl files with a few clicks
- **Sitecore Explorer**—allows developers to view the content tree for multiple websites and multiple databases, deleting, editing, or dragging and dropping multiple items simultaneously
- **Item Editor**—edits content of Sitecore items directly inside Visual Studio. The Item Editor supports multi-edit, so that multiple items can be updated in a single operation
- **Query Analyzer**—provides scripting CRUD
- **Search and Replace**—supports global search and replacement operations
- **Template Designer**—provides an editor for Templates
- **Media Library**—supports search and drag-and-drop for uploads
- **Expanded web.config Viewer**—shows the effective web.config after merging the included files
- **Publishing Operations**—includes publishing queue management
- **Validation Manager**—performs a health check of a Sitecore installation
- **Debug and Trace**—allows a view of the Profile and Trace files

- **Administration**—includes access to the recycle bin and archives, rebuilding the links database and rebuilding search indexes
- **Log Viewer**—manages Sitecore log files
- **Job Viewer**—manages background jobs
- **Index Viewer**—lets you query, view and drill down into Lucene indexes by document, field and term
- **Template Hierarchy Viewer**—indicates the base templates and derived templates for a Template
- **PowerShell**—exposes a Sitecore database as a PowerShell drive
- Install **NuGet** packages that includes Sitecore items and files

Apply – Topic 2.5 – 35 min



Create a bicycle data template and associated presentation details using Sitecore Rocks

In the following labs you will:

- Create a new Sitecore Rocks connection to your instance
- Create a Bicycle data template with a number of fields
- Create a number of content items based on the Bicycle data template and populate with data
- Bind presentation details (*main* layout and *Introduction* component) to each item
- Create a new sublayout called *Bicycle Overview*
- Bind it to each item based on the *Bicycle* data template

Lab A. Use Sitecore Rocks to link your project to your Sitecore instance

1. In **Visual Studio**, right-click on your project and select **Sitecore > Connect** from the context menu
2. If you can see your connection, **select** it and click the **OK** button and move on to **Lab B**, otherwise continue with the instructions below
3. If you don't see your connection click on the **New...** button on the bottom left

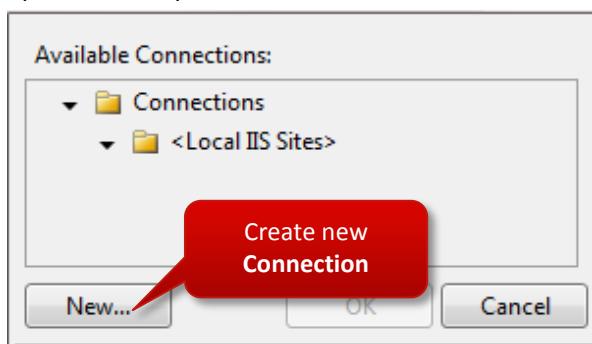


Figure 2-47 Creating a connection to a web site

4. In the Connection dialog box, select the following:
 - Data Provider: **Hard Rock Web Service**
 - Host Name: **Training** (your instance)
 - User Name: **sitecore\admin**
 - Password: **b**
 - Location: **C:\inetpub\wwwroot\Training\Website** (your website root)
5. Click the **Test** button. A warning that the **Hard Rock Web Server not yet installed** displays

| | |
|----------------|---|
| Data Provider: | Hard Rock Web Service |
| Host Name: | Training E.g. 'www.server.com' or 'localhost:81' |
| User Name: | sitecore\admin E.g. 'sitecore\user' |
| Password: | b |
| Location: | C:\inetpub\wwwroot\Training\Website |

Figure 2-48 Creating a connection to a web site

6. Click the **Yes** button to update (i.e. install) the server components, after a moment the Update Server Components dialog box appears
7. Click **Close** to continue, (this dialog is used both to show a new installation has taken place and to update existing components)
8. The **Yes, it works!** dialog box appears, click **OK**
9. Select your **connection** in the **Available Connections** dialog box, and then click the **OK** button

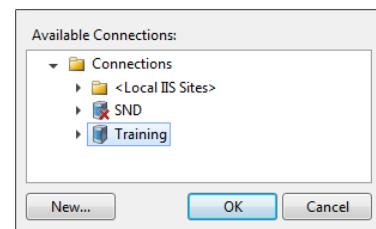


Figure 2-49 Sitecore Rocks connections

Lab B. Create a data template

1. Open **Sitecore Explorer** from the **SITECORE** menu
2. Using the **Sitecore Explorer** expand your connection: **master > sitecore > content** to see the Content Tree along with all the items you created in the previous Topics
3. Navigate to **/master/sitecore/templates/User Defined** folder to create a new data template
4. Right-click the **User Defined** item and then click the **New Template...** option
5. Enter **Bicycle** as the data template name



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, if you select an item in the tree the use the shortcut, you will be able to perform task on that item

6. Complete the data template following the schema below, add a field section entitled **Basic Information** with three fields: **Heading**, **Main Content**, and **Main Image**, and a field second section entitled **Bicycle Information** with two fields: **Type** and **Suitability**

Figure 2-50 Defining fields in Sitecore Rocks

7. **Save**
8. Click the **document** icon next to **Bicycle** at the top left
9. Select any tab and in the bottom text field type: **PeopleV2/32x32/bicycle.png** and click the **OK** button

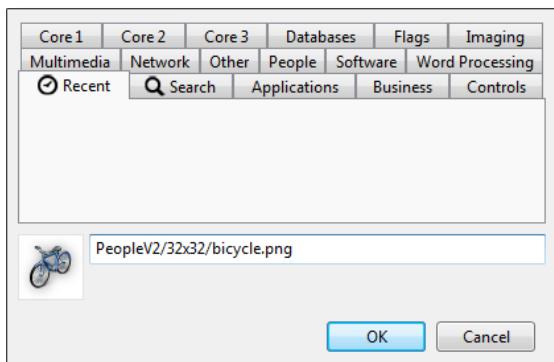


Figure 2-51 Selecting icon in Rocks

10. Save and close the Bicycle data template

Lab C. Create a content items based on this data template

1. Navigate to `/sitecore/content/Home/holidays/Explore Holland`
2. Right-click **Explore Holland** and select the **Add > New Item...** option from the context menu
3. In the **Add New Item** dialog box enter **Bicycle** in the filter text field to show the Bicycle data template you just created

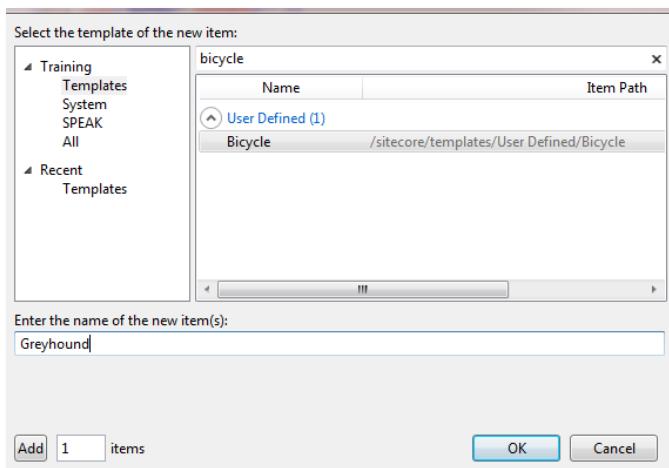


Figure 2-52 Add new item in Rocks

4. Select the **Bicycle data template** and call the item **Greyhound** then click **OK**
5. You should now see the two field sections with the fields that you created
6. You can now **populate** these **fields** with sample values i.e. to **add an image**, expand the **Images** folder in the **Media Library**, path is: `/sitecore/media library/Images`
7. **Drag and drop** the **image** you want **into** the **Main Image** field, you can also drag and drop an image from the file system which would upload it automatically

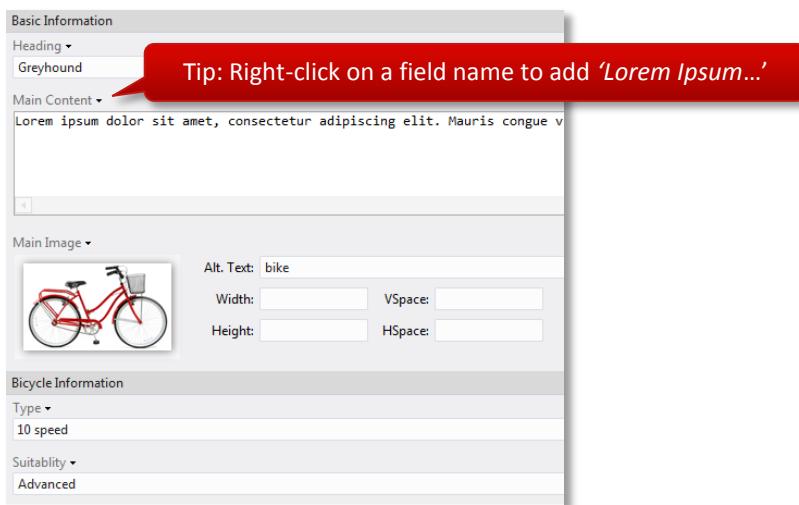


Figure 2-53 Item fields in Rocks

8. **Save** when you are done editing and **close** the Greyhound.item tab

Lab D. Re-use presentation

In the following lab you will assign existing presentation details to the Greyhound content item

1. Right-click on the **Greyhound** item and choose the **Tasks > Design Layout** option on the resulting context menu
2. In the window, click the **Browse...** button next to the **Layout** text box and choose the layout named **Main** and click **OK**

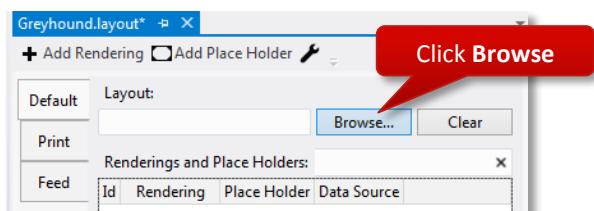


Figure 2-54 Select a layout in Rocks

3. Click the **Add Rendering** option in the top left corner

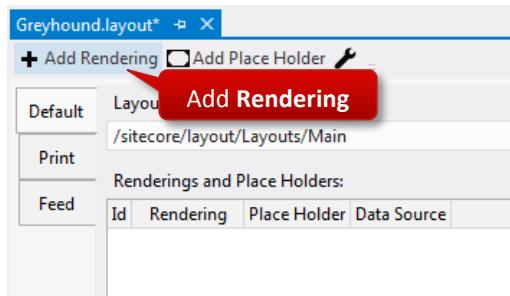


Figure 2-55 Add a new rendering

4. To find the **Introduction** sublayout, select **All** in the filter on the left and start typing **Introduction**, select the result and click **OK**

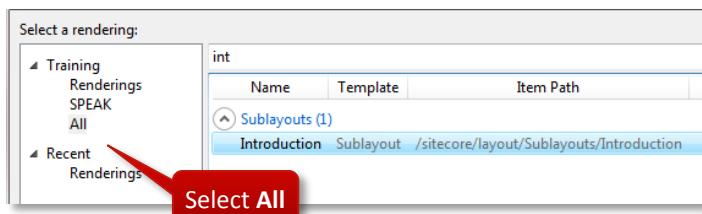


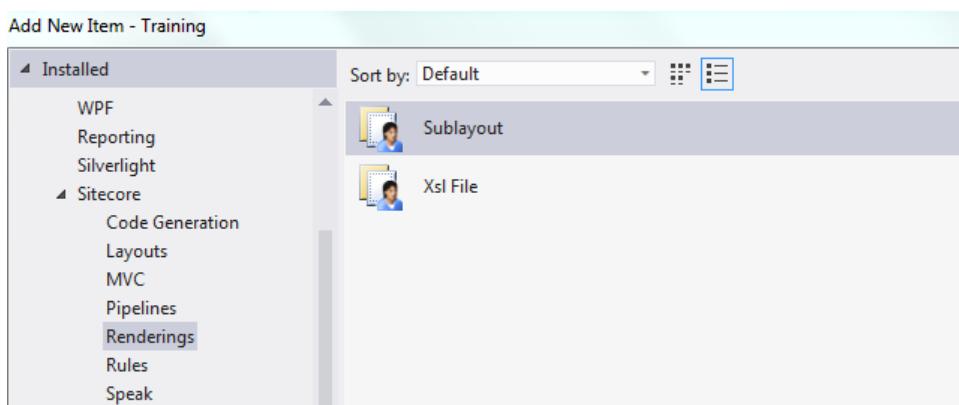
Figure 2-56 Filter the rendering list

5. Press the **F4** key / click **properties command** (the spanner) or **double click** on **Introduction** to get to the **components properties dialog box**
6. In the **PlaceholderKey** property and click the **ellipsis (...)** button next to it and select **main**, alternatively you can simply type **main** into the field and click **Close**
7. **Save** and close the **Greyhound.layout** tab
8. Right-click on the **Greyhound** item in the tree and choose the **Tools > Browse > Preview the page** option in the context menus
9. You should see the bicycle image, heading, and content

Lab E. Create a new component

There are two fields that are specific to the Bicycle page that we need to display. In this lab you will create a component to output the **Type** and **Suitability** fields

1. In the **Solution Explorer**, add a new component by right-clicking on your project, and choosing the **Add > New Item...**
2. Locate the Sitecore submenu and choose to add a **Sublayout**



2-57 Creating a new Sublayout through Rocks

3. Name the sublayout **BicycleInformation.ascx**
4. In the resulting dialog box, select **/master/sitecore/layout/Sublayouts** and click the **OK** button – this creates the definition item in Sitecore with the correct mapping to this file
5. **Copy** the **HTML** from the **student resource folder** (*WND Labs > Module 2 > Topic 2.5 > Lab E > HTML > bicycle-information.html*) and **paste** it into **BicycleInformation.ascx**
6. **Replace** all the **static content** in the sublayout **with Sitecore controls** to read the values from the items fields
7. Your **finished HTML** should look similar to this:



Code sample – Sitecore controls

```
<div class="indentedSection">
    <table class="bikes">
        <tr>
            <th>Type</th>
            <td><sc:Text Field="Type" runat="server" /></td>
        </tr>
        <tr>
            <th>Suitability</th>
            <td><sc:Text Field="Suitability" runat="server" /></td>
        </tr>
    </table>
</div>
```

8. **Deploy** your project
9. Assign the **Bicycle Information** sublayout to the Greyhound item, (right click on Greyhound > **Tasks > Design Layout > Add Rendering** and add the sublayout to the **main** placeholder). It should appear below the Introduction sublayout
10. **Save and close** all open **tabs** in Visual Studio
11. **Preview** the item by clicking on Greyhound item and use **Commandy** (CTRL + Shift+ Space), then type in preview the Sitecore Explorer
12. **Alternatively** you can **publish** your site and then browse to it by typing publish into commandy

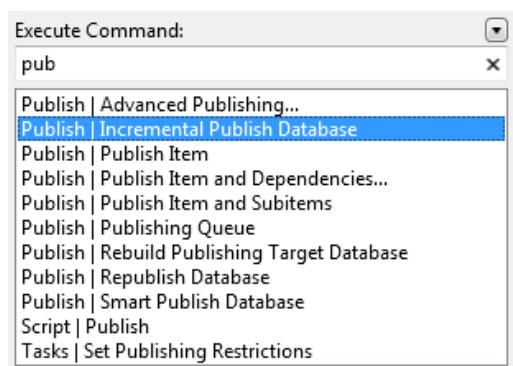
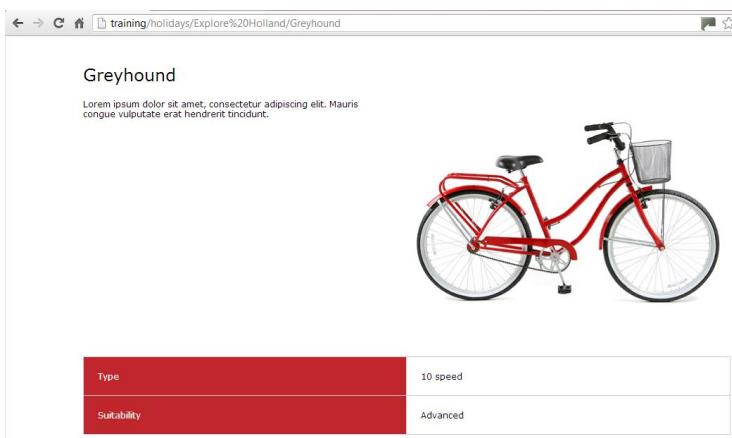


Figure 2-58 Publish through commandy

13. Open up a **new browser** and navigate to: <http://training/holidays/Explore%20Holland/Greyhound> to see your **deployed** and **published** work



2-59 The completed Greyhound page

Review

Sitecore Rocks

Q Describe some features in Sitecore Rocks

- ✓ Sitecore Project Management
- ✓ Sitecore Explorer
- ✓ Item Editor
- ✓ Template Designer
- ✓ Media Library
- ✓ Publishing Operations
- ✓ Sitecore Rocks allows developers to do create Sitecore items from the files on the file system

Extend

Sitecore Rocks cheat sheet:

<https://www.sitecore.net/~media/Community/Technical%20Blogs/West%20blog/2011/June2011/SitecoreRocksCheatSheet.ashx>

28 Days of Sitecore Rocks:

<http://www.sitecore.net/Community/Technical-Blogs/Trevor-Campbell/Archive.aspx#archive2013>

Topic 2.6 Module Summary

Content and Presentation

Defining data structure

Layouts and layout definition items

Outputting content

Components and component definition items

Static binding versus dynamic binding

Content

1 Create Data Templates

2 Create Content Items

Presentation

3 Create Layout

4 Create Components

Bind

5 Template Standard Values

Presentation Layout Details

Module 3

Items Items Items

Contents

- Data template inheritance
- Standard values
- Insert options
- Other item and template properties

Topic 3.1 Data template inheritance

Introduction

Objectives

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

- Describe the main reason for data template inheritance
- Name the data template that all data templates inherit from

Content

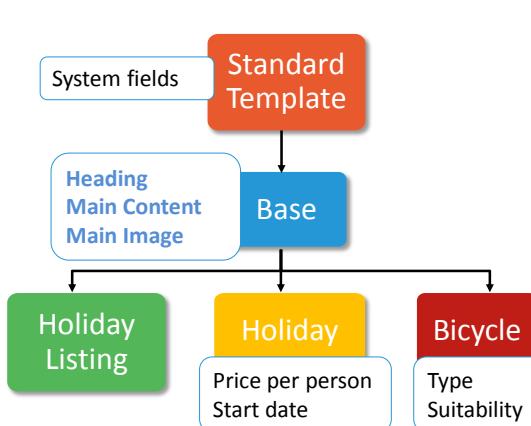
Challenges with field duplication

- There is **no one true source** for these fields
- Changes to field names have to be **applied on every single data template**



Solution – Data template inheritance

- Merges** the fields from all inherited data templates into a superset of fields
- All data templates inherit from the Standard Template, which contains fields that define system properties available to all items
- If you modify the **Base** template, it will affect all the items immediately



- A data template can inherit from any number of base templates
- A data template can be used multiple times in the inheritance chain without negative effect (e.g., if Base and Holiday both inherit from Standard Values, the fields will not appear twice)



Best Practice

- Although you can use a data template multiple times in the inheritance chain, you should avoid creating a circular template inheritance (A → B → A) at all costs
- 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 (i.e., if the content editor 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 (e.g., 20 levels)

Resulting data templates

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



- Watch out** for circular data template inheritance – a data template should **never** inherit directly or indirectly from itself
- Note:** Field names need to be unique, and field section with the same name will merge



Demo – Setting up and viewing data template inheritance

In this demo we will be:

- Creating a new data template with one new field
- Have our *Holiday* template inherit from it using rocks
- View data template inheritance using the Template Manager

- In Sitecore Rocks, right-click the **/sitecore/templates/User Defined/** item and create a new data template called **Meta Data**

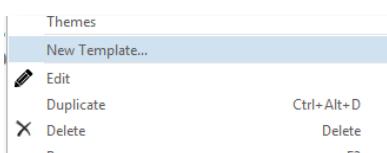


Figure 3-1 New Template option

- In the field section enter **Meta Data Information** with a field called **Keywords** with **Single-Line Text** as the field type

3. Change the **icon** and click the **save** option
4. We want the **Holidays** item to have this meta data field so make the **Holiday Listing** data template inherit from **Meta Data**, by double clicking on **Holiday Listing** and right-clicking in the **Editor** pane
5. Choose the **Set Base Template**
6. Notice that **Standard template** is listed as a base template, and that other templates can be selected by selecting the relevant checkbox
7. Filter by typing in **meta**, select it, click **OK** and **save**
8. **View Holidays item** in the content tree and notice the field has been added
9. You can also right-click on any item and see their chain of inheritance
10. Right-click on the **Holidays** item and click **Navigate > Template Hierarchy**
11. Select the **Super Templates** option to see the templates that are being inherited from, notice the **Meta Data** template
12. You can click on each data template in the hierarchy to see which fields it contributes to the ultimate superset of fields
13. You can perform the same tasks in the Template Manager using the **Inheritance** tab to see the inheritance chain and the **Content** tab to select base templates

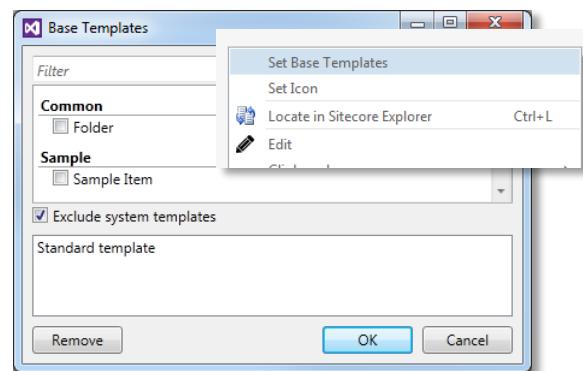


Figure 3-2 Set base templates in Sitecore Rocks

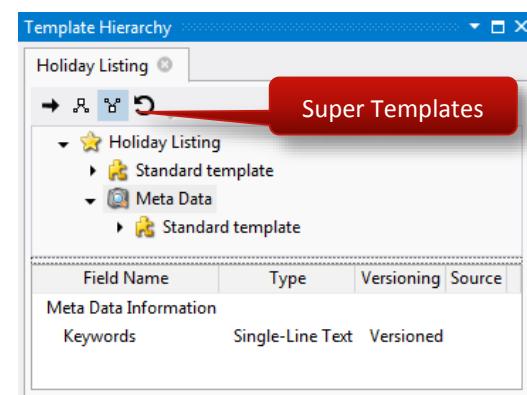


Figure 3-3 View super templates in Rocks

Set and view inheritance using rocks

Set inheritance

Double click template

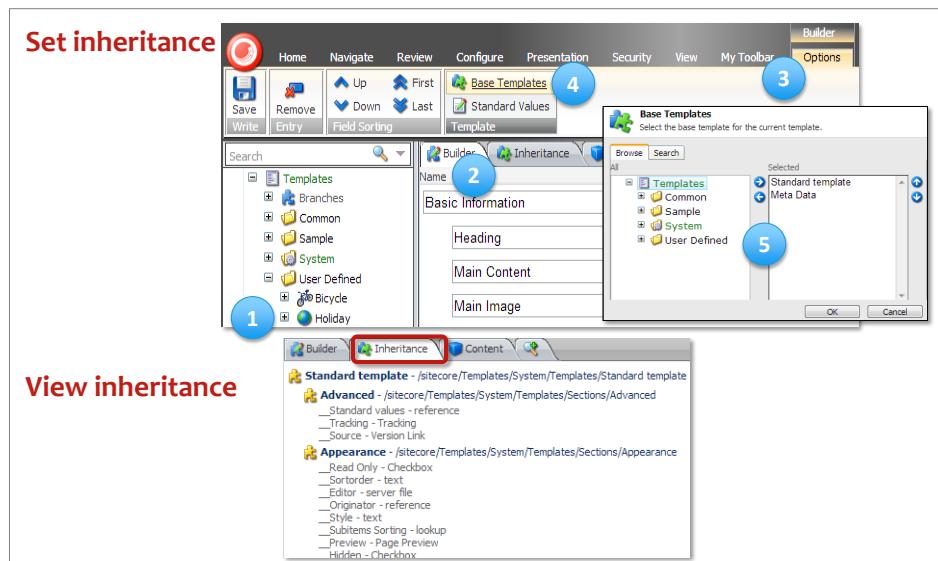
In the *Editor* pane right click and select **Set Base Template**

Choose the data template you want to inherit from

View inheritance

Right click on any item

Set and view inheritance using the template manager



View inheritance



Best Practice

Encapsulate all fields required for a specific feature in a data template, which other templates can then inherit from (e.g., Page Meta Data [Title, Description, Keywords] or Banner [Banner Image, Banner Text])



Tip

*Enabling **Standard** fields in the View group in the Content Editor shows 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*

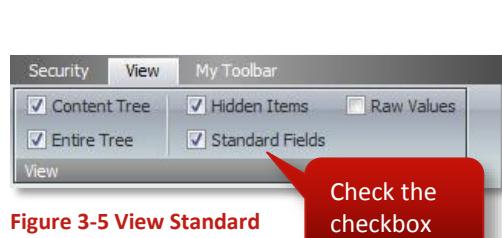


Figure 3-5 View Standard fields in Content Editor

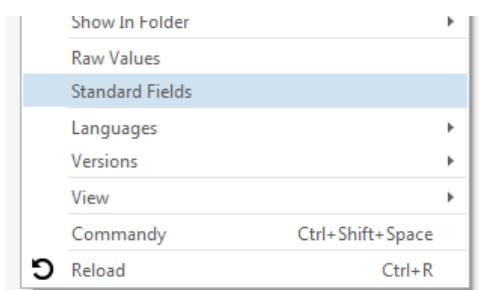


Figure 3-4 View Standard fields in Sitecore Rocks

Apply – Topic 3.1 – 15 min



Data template inheritance

In the following labs you will use Sitecore Rocks to:

- Create a Base template to define common fields
- Delete the duplicated fields from your existing data templates
- Make the three data templates inherit from the Base

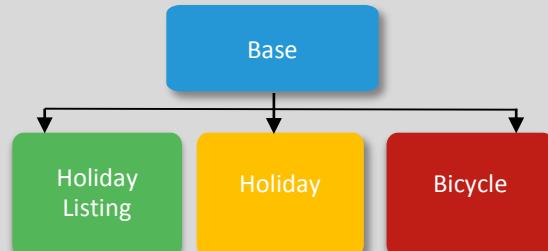


Figure 3-6 Data template inheritance structure

Lab A. Create a base template and delete duplicate fields

1. In Sitecore Rocks, create a new template called **Base** in `/sitecore/templates/User Defined`
2. By following the schema below, add one field section called **Basic Information** and **three fields** of various types and **save**

Figure 3-8 Create fields

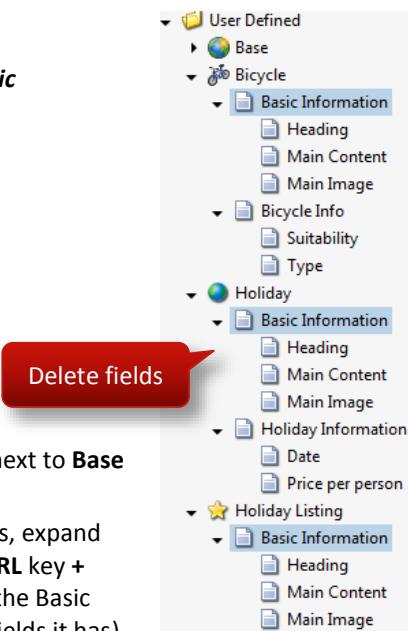


Figure 3-7 Multi select items in Rocks



Important

When deleting fields from a data template, keep in mind that you are effectively deleting those fields from the items that are based on this data template. Therefore, some thought has to go into your data template creation and inheritance structure

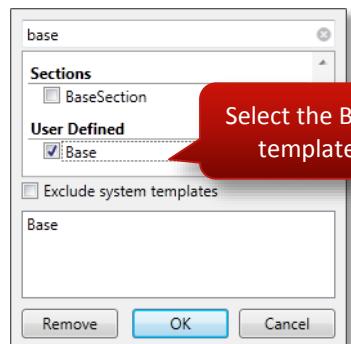
6. Right-click to **delete** and **confirm** that you want to delete these items

Lab B. Change Inheritance

1. Double click the **Bicycle** item. Right click in the **editing** pane and in the context menu select **Set Base Template** (See **Figure 3-2 Set base templates in Sitecore Rocks**)
2. Remove the **Standard Template** and select the **Base** option
3. Click **OK** and **save**

**Tip**

*Because **Base** Template already inherits from **Standard** Template, any data templates inheriting from Base Template will also inherit the **Standard** Template fields*

**Figure 3-9 Select a template**

4. Do the same for the **Holiday** and **Holiday Listing** template
5. **Save** all items and click **OK** in the confirmation dialog box
6. Navigate to the **Holidays item** under **master/Sitecore/content/Home/**, notice that the fields have been added but their values are now blank because you have **deleted the fields** from the data template that the item was using when you changed the inheritance. Leave them empty for now, as we will cover this in the next topic

Review**Data template inheritance**

- | | |
|--|---|
| <p>Q In what scenario would you use data template inheritance</p> <p>✓ When you have fields that will be reused in multiple data templates</p> | <p>Q By default what data template do all data templates eventually inherit from?</p> <p>✓ Standard Template</p> |
| <p>Q Why is it important to think of your data template creation and inheritance structure from the beginning?</p> <p>✓ You should not have to be deleting fields / field sections, values get lost</p> | <p>Q Do fields with the same names merge into one field?</p> <p>✓ No</p> |
| | <p>Q Do field sections with the same names merge into one field section?</p> <p>✓ Yes</p> |

Topic 3.2 Standard values

Introduction

Objectives

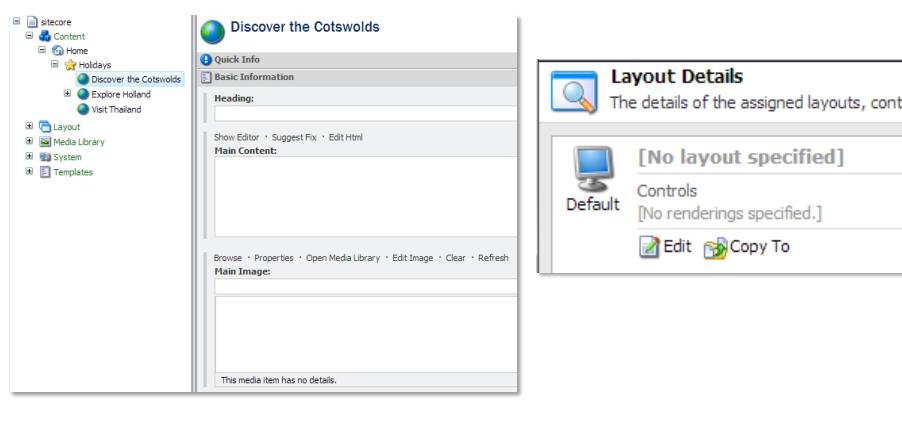
By the end of this topic you will be able to:

- Describe how to solve repetition with setting field values and presentation details on every item
- State what type of settings can be applied to standard values
- Understand the implication of resetting back to standard values

Content

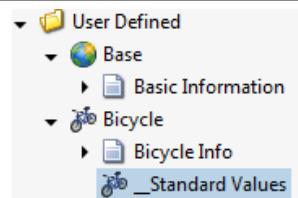
Repetition challenges

- For every new item - you have to specify the **item's name**, **populate all field values**, and **configure presentation details**



Solution - Standard values

- A template's **standard values** is an item
- **Created manually**
- **Regular child item** of the given template
- Made up of **all the fields** from all the **inherited templates**, including its own
- Always named **_Standard Values**



Standard values allow developers to:

Standard Value Settings:

- Specify **default field values**
- Provide **initial field values** using token replacement (e.g. \$date or \$name)
- Assign **default presentation details**
- When Sitecore retrieves an item, it will retrieve its **fields** and **field values**
- If any of those fields are null, it will check, and use, the value of the **same field** in the **Standard Values item**



Walkthrough – Create standard values, define default field values

We will now set up standard values for the base template that you created earlier and define some default field values

- In Rocks, right-click on `/master/sitecore/templates/User Defined/Base` and click **Create Standard Values**. A **_Standard Values** child item will be created
- Leave the **Heading** field blank for now and right-click on the **Main Content** field title
- Select the **Add 'Lorem Ipsum...'** option to add Lorem Ipsum text to the *Main Content* field
- Drag and drop an image into the **Main Image** field from the student **resource folder** (*WND Labs > Module 3 > Topic 3.2 > no-photo-available.jpg*)
- Save the _Standard Values item**
- Navigate to **Holidays** or **Discover the Cotswolds** item and notice that the previously empty fields are prefilled with default text from the standard values from the base template



Figure 3-10 Add Lorem Ipsum



Knowledge Check

Notice the different fields for the Holiday items vs. Discover the Cotswolds, why are some fields on the Discover the Cotswolds empty?

Part 2 of the Walkthrough Using Tokens

We will add the \$name token to the Base Template and create a new Holiday item

1. In the Base standard values, type **\$name** into the **Heading** field and **save**
2. **Navigate to an existing holiday item** (e.g., **holidays** or one of its children) and confirm that **\$name** now appears in the **Heading** field
3. **Add a new Holiday item** under **/Sitecore/content/Home/holidays** and call it **Cycle California**



Figure 3-11 Default values

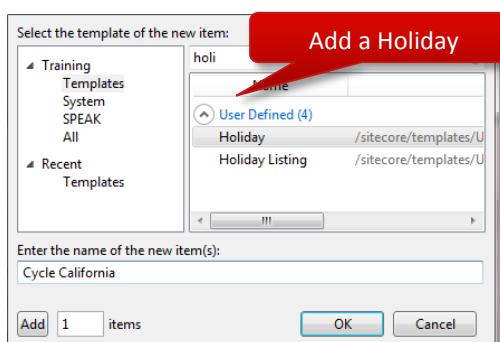


Figure 3-12 New item

4. Confirm that the **\$name** token in Heading has been **replaced** by the **item name**



Knowledge Check

Why does \$name appear in the *Main* Heading field for all the existing items and all the other fields have been replaced with the text you have entered?



Important

Note that replacement only happens on creation. Adding a token later on will **not** affect existing items based on the templates

Creating standard values



Default values defined

Default values defined in the Base Standard Values

Default values inherited

- Holiday data template is **inheriting** from the **Base** data template
- Default values set up on Base Standard Values are **inherited** by the Holiday data template

[standard value] -> fields using default values defined in _Standard Values and have not yet been overridden

- These values **can be overridden** on the Holiday Standard Values

Instance of an item

What happens when a new item is created?

- Cycle California is **derived** based on the Holiday data template
- We get the **default values** that the Holiday data template inherited from Base

Using tokens

- You can use **standard values** to perform **initialization of a field value**
- Sitecore offers several tokens which will be replaced by their current value at the moment an **item is created**:

| Token name | Description |
|--------------|------------------------------------|
| \$name | Name of the item |
| \$id | ID of the item |
| \$parentid | ID of the parent of the item |
| \$parentname | Name of the parent of the item |
| \$date | Current date (ISO format) |
| \$time | Current time (ISO format) |
| \$now | Current date and time (ISO format) |



Important

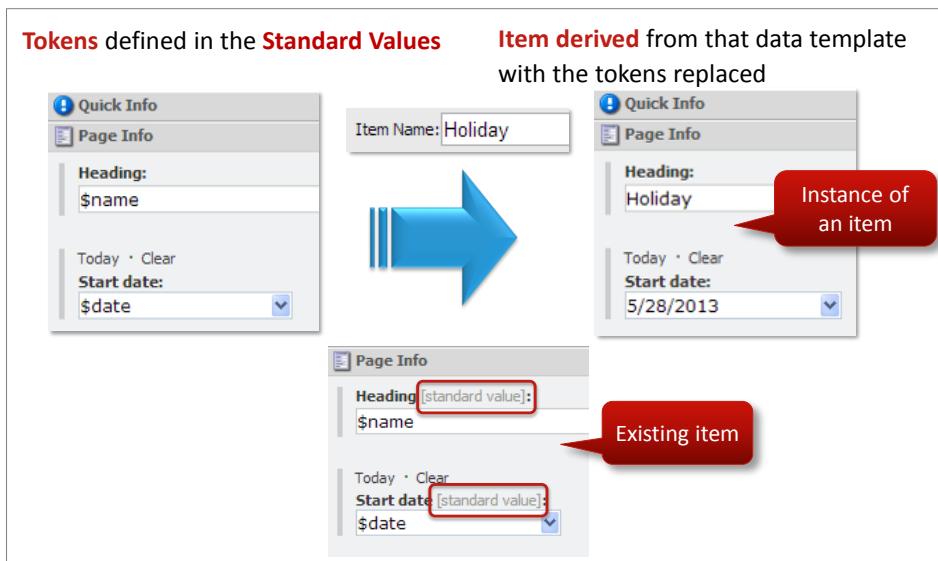
These tokens are only expanded at the moment of creating an item. This means that if you add a \$name token to a field in template's standard values, items that have already been created will literally have \$name in that field



Business Use Case

Tokens are useful for various reasons, for example the title of an article is required in several fields (navigation title, page title and a heading), when using tokens you don't need to type the name of the item in all those places

Tokens in action



Naming items

- Creating items without “-” is preferred because of **tokens (business names)**
- Tokens get replaced with the item name, if your item has “-” in the name, the token will then also have “-”

New item

Renamed item

Options:

- Create item with a space and rename each item in the tree manually after it's been created – that way the tree will have a “-” but the field name has a space
- A configuration change in the **web.config** is required



Note –URL duplication

By default you can add 30 'Holiday' items and Sitecore will just give you a warning saying that you have duplicated names

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



Walkthrough – Add presentation details to standard values

We mentioned that default values are not the only configuration being inherited. Presentation details also get inherited, so let's take a look at that.

In this walkthrough, we assign default presentation details to the Base standard values and create a new holiday

1. Preview the **Cycle California** page by right-clicking the item and selecting **Tools > Browse > Preview** option from the resulting drop-down menu, or use **command**
2. We did not assign any presentation details to the item and we see a **No Layout Found** page
3. Rather than assigning presentation details directly, we will do it on the Base standard values, right-click on the **Base standard values** item, and click and in the sub menu select **Tasks > Design Layout**, or use command
4. In the Default device select **Browse...** and choose **Main**
5. Click the **Add Rendering** option and select the **Introduction** sublayout option

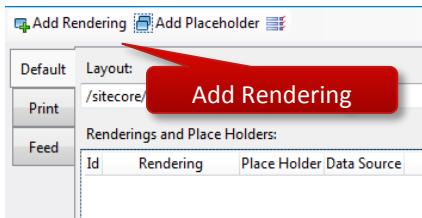


Figure 3-13 Add a new rendering

6. To specify the placeholder for the **Introduction** sublayout, click the **properties** command
7. Click into the **PlaceholderKey** property. Click the **ellipsis (...)** button, and select the **main** placeholder key

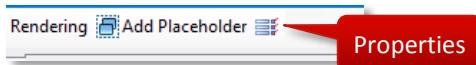


Figure 3-14 Component properties

8. Save the standard values item
9. Preview the **Cycle California** page again and you should now see the layout and sub-layouts that you added to the Base Template standard values
10. Notice that the Holiday Overview component has not been added to the Base Template standard values, because that rendering is specific to Holiday items

Cycle California

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris congue vulputate erat hendrerit tincidunt.



Figure 3-15 Final result



Knowledge Check

Given that the Holiday Overview component should only appear on items based on the Holiday data template, where should the Holiday Overview component be added?

-
11. Navigate to one of the earliest holidays you created (e.g., **Discover the Cotswolds**)
 12. Right-click on the item and click **Tasks > Design Layout**
 13. Notice that the layout does not match the layout defined on the Base Template standard values
 14. It is possible to **reset** both **presentation** details and **field values** to the **template's standard values** which we will cover later



Knowledge Check

Given that we created the Discover the Cotswolds item before we knew about standard values, why do the presentation details not match the standard values?

Presentation details inherited

Like any other standard values, presentation details are passed down the inheritance chain and can be overridden at item level

Default settings defined in the Base Standard Values

Cycle California **item derived** from the **Holiday** data template which is **inheriting from Base** resulting in **presentation details** getting **inherited** from **_Standard Value**

The screenshot illustrates how presentation details defined at the 'Base' standard value level are inherited by the 'Cycle California' item, even though it was created later.



Best Practice

It is best practice to assign any default presentation details to standard values. If a component needs to be added to all pages at any time in the future, adding it to the standard values will update all items based on that template. Additional components can be added per item



Walkthrough – Resetting to standard values

Authors with the appropriate access rights can reset an item's fields or properties to the values defined in standard values – including presentation details, you may want to do this when presentation on the item has been overridden and you need to reset it to what has been originally defined

1. In the **Page Editor**, add a number of components to the **Discover the Cotswolds** item

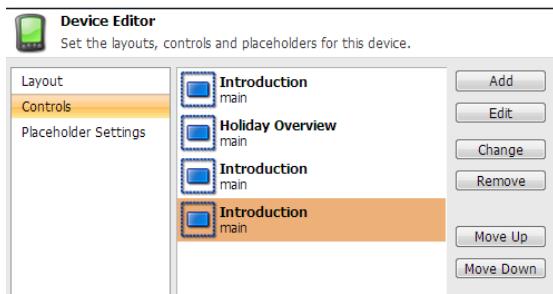
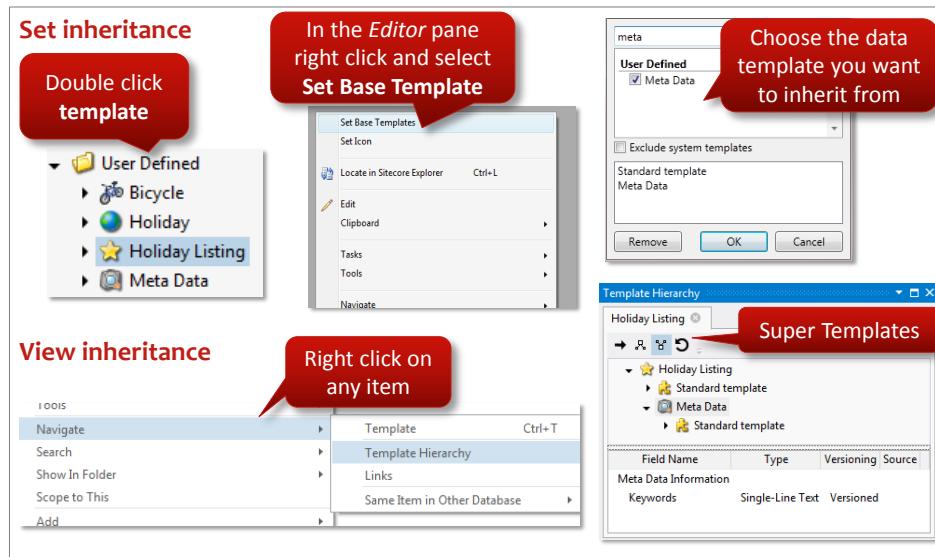


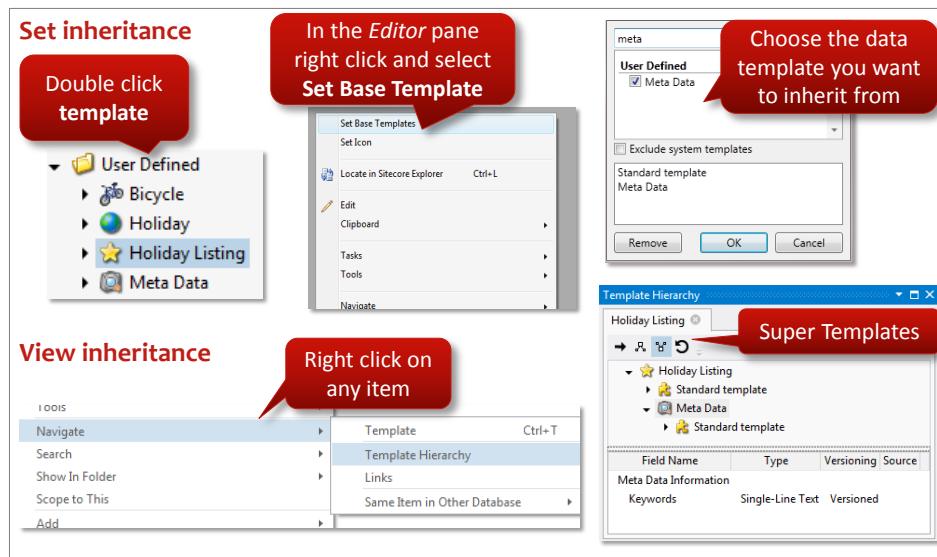
Figure 3-16 Multiple components on an item

2. Confirm that the item has a number of components and looks messy
3. Click on the **reset** command in the ribbon and click the **OK** button when prompted
4. In the Content Editor, you can view where the presentation details are actually stored for an item if you go to the **View** tab and click the **Standard Fields** option
5. Look for the **Layouts field section** and the **Renditions field** which stores presentation details for the item, we will talk more about this field later
6. You can reset all the fields, including the Renderings field

Reset to standard values (Page Editor)



Renderings field



Reset field values using Content Editor

1. Select the item you need
2. In the ribbon, select the **Versions** tab and the **Reset** command

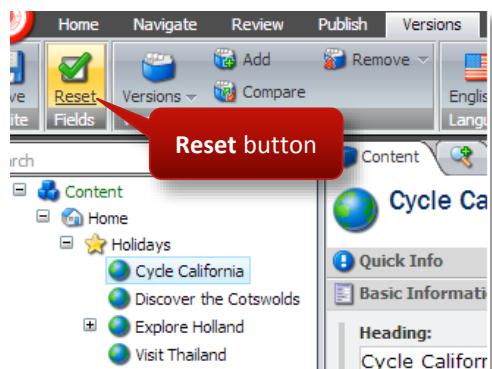


Figure 3-17 Reset fields command in the Content Editor

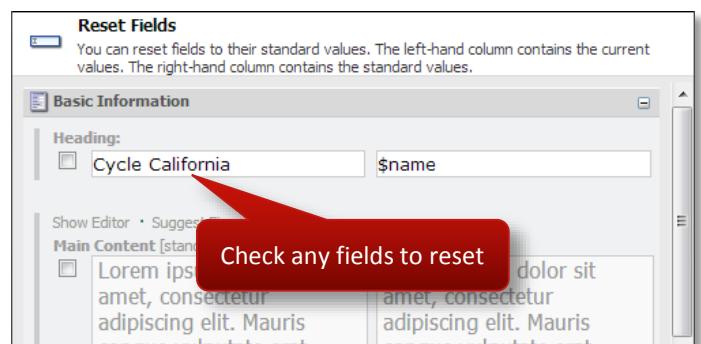


Figure 3-18 Select the fields you want reset

Reset presentation details using Content Editor

1. Select the item you need
2. In the ribbon, select the **Presentation** and **Reset** commands



Figure 3-19 Reset presentation details command in the Content Editor ribbon

3. Presentation details are stored in a *Renderings* field that is part of the Standard Template. To view that field in the Ribbon, select the **View tab > Standard Fields**
4. Look for the **Layout** field section and the **Renderings** field

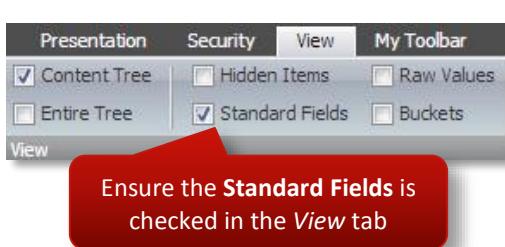


Figure 3-20 View the Standard Fields

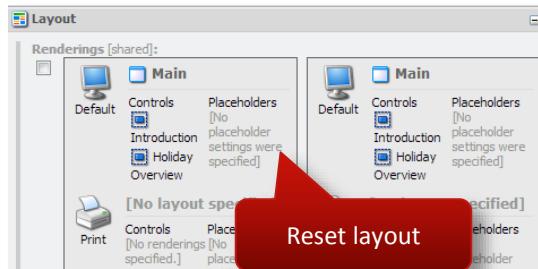


Figure 3-21 Reset the layout in the Renderings field

Reset presentation details using Rocks

1. Select an item in the tree
2. Right-click on the **editor** pane, and from the content menu select **Layouts > Reset to Standard Layout on Save**



Figure 3-22 Reset presentation details in Rocks

Reset field values using rocks

1. In the selected item, right-click on a field and select **Reset to Standard Value on Save**

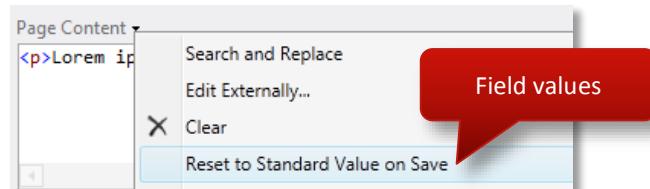


Figure 3-23 Reset field values in Rocks

Data template standard values

Some settings that should be assigned on the standard values of a data template are:

1. Default values

- Specify default field values using either tokens (e.g. \$name, \$now etc.) or “enter text here...”

2. Presentation details

- Assigning various presentation components (layouts, sublayouts etc.)

3. Insert Options *

- Specify which type of item a user can add under the current item

4. Workflows *

- Content approval process

Standard Values

1. Default values

2. Presentation details

3. Insert Options *

4. Workflows *

Apply – Topic 3.2 – 15 min



Create standard values

In the following labs you will:

- Reset presentation
- Create standard values for Holiday Listing, Holiday and the Bicycle data template
- Set default field values for all the fields
- Assign presentation details

Lab A. Reset presentation

- In Rocks, Reset presentation to the standard layout on **all** items in the tree – including items based on the Holiday Listing, Holiday and Bicycle data templates, refer to [Figure 3-22 Reset presentation details in Rocks](#)
- Preview several of your items (particularly **Holiday** and **Bicycle** items) and confirm that you cannot see the Bicycle Information or Holiday Overview sub-layouts after resetting

Lab B. Create standard values and enter default values

- In Rocks, right-click on `/master/sitecore/templates/User Defined/Bicycle` and click the **Create Standard Values** option
- Do the same for the **Holiday Listing** and **Holiday**



Knowledge Check

Why do the bicycle, holiday listing, and the holiday have prefilled fields?

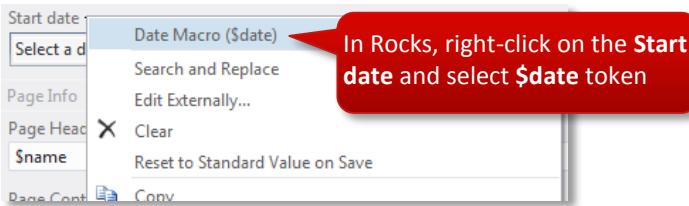


Figure 3-24 Adding date token in Sitecore Rocks

3. Enter some **default values** for the **Bicycle** and the **Holiday** data template



Tip

Remember to use **tokens** (e.g., \$name or \$date), which can be added to the Start date field as shown here

Lab C. Assign presentation on the standard values

1. Right click on the **Bicycle standard values** and in the sub menu select **Tasks > Design Layout**, or use **commandy – CTRL+Shift+Space**
2. Because presentation details are assigned to the Base data template from which the Bicycle data template inherits, you will be given an option to copy from standard values. Click **Do you want to copy from standard values?** The Main layout and Introduction sub-layout will appear

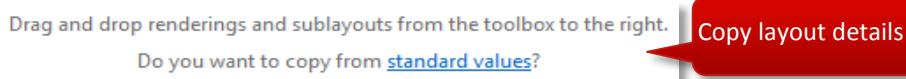


Figure 3-25 Copy layout details from the Base data template

3. In addition to the presentation details that were just copied in, add the **Bicycle Information** sublayout to the **main** placeholder

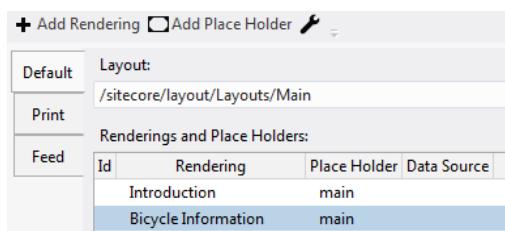


Figure 3-24 Resulting bicycle data template presentation details

4. Do the same for the **Holiday** data template, adding the **Holiday overview** sublayout to the **main** placeholder

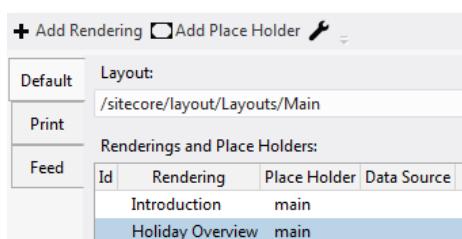


Figure 3-25 Resulting holiday data template presentation details

5. The Holiday Listing has no additional fields apart from what is inherited from the Base data template. So you do not need to add any additional components nor do you need to copy presentation details from the standard values of the Base

Lab D. Create new items

1. Using **Rocks** create a new item based on the holiday data template under **content/Home/holidays/** called **Uncover Copenhagen**
2. **Choose an image** that you have already uploaded
3. Create a new Bicycle item under that called **Principia**
4. Choose an image that you have already uploaded or upload the image below from the **student resource folder** (*WND Labs > Module 3 > Topic 3.2 > Lab D > Images > mountain-bike.png*)
5. **Preview both items** in the **Page Editor** and confirm that the correct presentation details are being displayed, you can also use commandy to preview

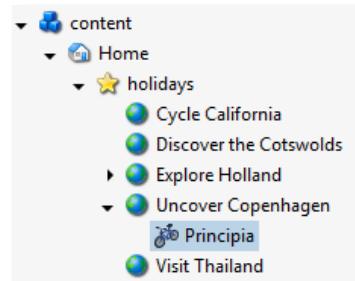


Figure 3-26 Content tree

Uncover Copenhagen

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking and adventures](#). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi scelerisque dui at nunc scelerisque eu molestie dolor porttitor. Donec aliquam mi eget sem consequat tempor.

Vestibulum ultricies viverra viverra. Proin laoreet velit in erat sollicitudin iaculis. Ut et vehicula leo. Curabitur non dolor elit. Morbi ac ante volutpat ante sollicitudin posuere. Sed ut nibh ut felis hendrerit vulputate at vel tortor. Suspendisse dui nibh, rutrum nec tempus vestibulum, tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut at mollis eros. Suspendisse potenti. Duis ornare ornare tincidunt. Donec pellentesque aliquet urna in facilisis. In hac habitasse platea dictumst. Proin tempor ultricies velit, sit amet aliquam nunc aliquet eget.

Sed non pulvinar orci. In est nunc, adipiscing pharetra ornare consectetur, iaculis non mauris. Suspendisse suscipit ligula non dolor interdum posuere luctus enim accumsan. Morbi consectetur turpis quis turpis ultricies. Cras tempor sodales euismod. Donec ultricies mauris quis nulla bibendum vitae placet.



Principia

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking and adventures](#). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi scelerisque dui at nunc scelerisque eu molestie dolor porttitor. Donec aliquam mi eget sem consequat tempor.

Vestibulum ultricies viverra viverra. Proin laoreet velit in erat sollicitudin iaculis. Ut et vehicula leo. Curabitur non dolor elit. Morbi ac ante volutpat ante sollicitudin posuere. Sed ut nibh ut felis hendrerit vulputate at vel tortor. Suspendisse dui nibh, rutrum nec tempus vestibulum, tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut at mollis eros. Suspendisse potenti. Duis ornare ornare tincidunt. Donec pellentesque aliquet urna in facilisis. In hac habitasse platea dictumst. Proin tempor ultricies velit, sit amet aliquam nunc aliquet eget.



| | |
|------------------|------------------|
| Date | 20th April |
| Price per person | £2000 per person |

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

3-27 The finished pages



Best Practice

Always try to create standard values and set up tokens and default values when you create the template – before creating any items. Remember, that tokens are only replaced when you create items, and they are not replaced on existing items

Review

Standard Values

Q Describe standard values (Name, location & purpose):

- ✓ Standard Value item, child item, contains all the fields from its own data template as well as inherited ones

Q Name three tokens:

- ✓ \$name, \$date, \$id, \$parentid, \$parentname, \$time, \$now

Q How do tokens work?

- ✓ Tokens get replaced only upon creation of the item, later on they are just treated as regular text

Q What type of settings can be applied to standard values?

- ✓ Default values, presentation details, insert options and workflows

Q When would you reset back to standard values?

- ✓ When you have overridden fields on an instance of an item

Topic 3.3 Insert options

Introduction

Objectives

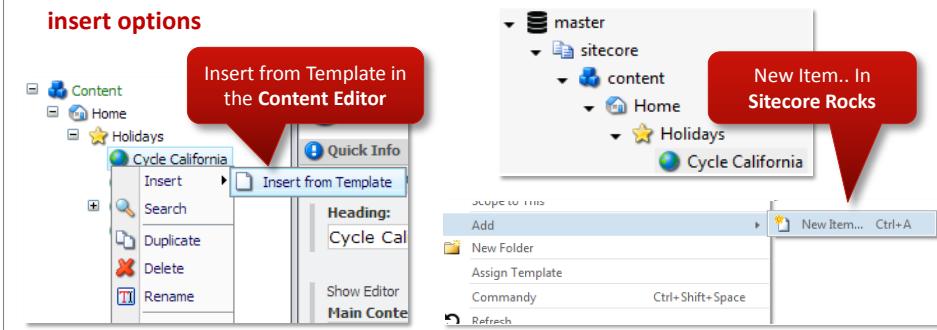
By the end of this topic you will be able to:

- Describe the functionality of Insert options
- State where you should assign them

Content

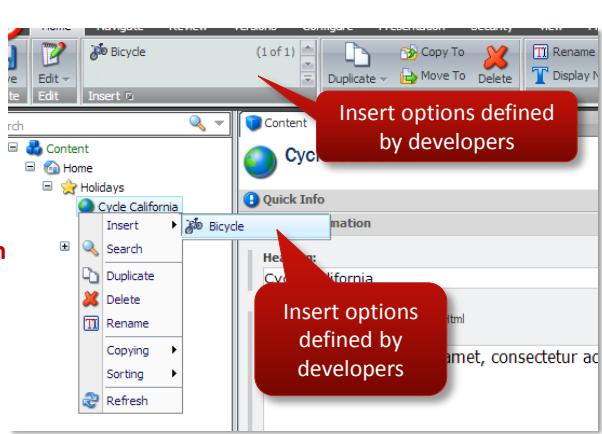
Adding new items - developers

- **Creating items** based on **any templates** we choose, **anywhere** in the content tree
- Reserved for **developers** and **administrators**
- Standard authors have no access to **Sitecore Rocks** or the **Insert from template...** option in the Content Editor
- **Authors** require a list of allowed template types to be defined – these are called **insert options**



What are insert options?

- A list of items that can be used to **create child items under a particular item**
- Authors would see a **restricted list of items** that they can insert
- **Define insert options on** a data template **standard values**
- Can be **overridden at item level**
- Reduces the possibility of authors creating items of a type that is unexpected, e.g. items based on a **system template**



Best Practices

Do not give your users administrative rights to avoid setting up insert options. An admin user will be able to view, and potentially delete, items in the site tree, as well as layouts, templates and system items



Walkthrough – Set Insert Options

In this walkthrough we will:

- Assign the Bicycle data template as an insert option for the Holiday data template using the Template Manager
- View how it can be done in Sitecore Rocks



Figure 3-3-28 Holiday data template insert options

- Open the **Template Manager**, show that if you want to add a new holiday item under `/sitecore/content/Home/holidays` from the **Home** tab, you have to use the **Insert from Template** option
- Navigate to the template using either the **Quick Info** section or in the tree, the path is: `/sitecore/templates/User Defined/Holiday/_Standard Values`
- Assign** the **Bicycle** data template to be the Insert option by going to **Configure** in the Ribbon and clicking the **Assign** option
- Go back to the holidays item and add a new holiday using the insert option
- In **Rocks**, you can right-click on the data template and select **Tasks > Set Insert Options**
- Filter your options

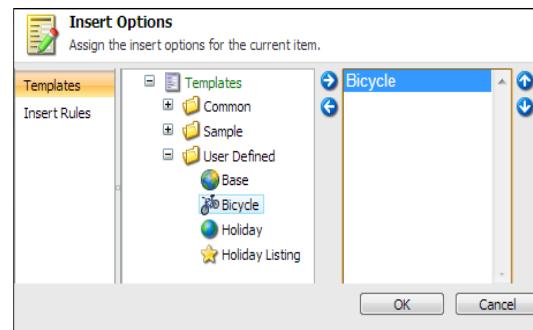


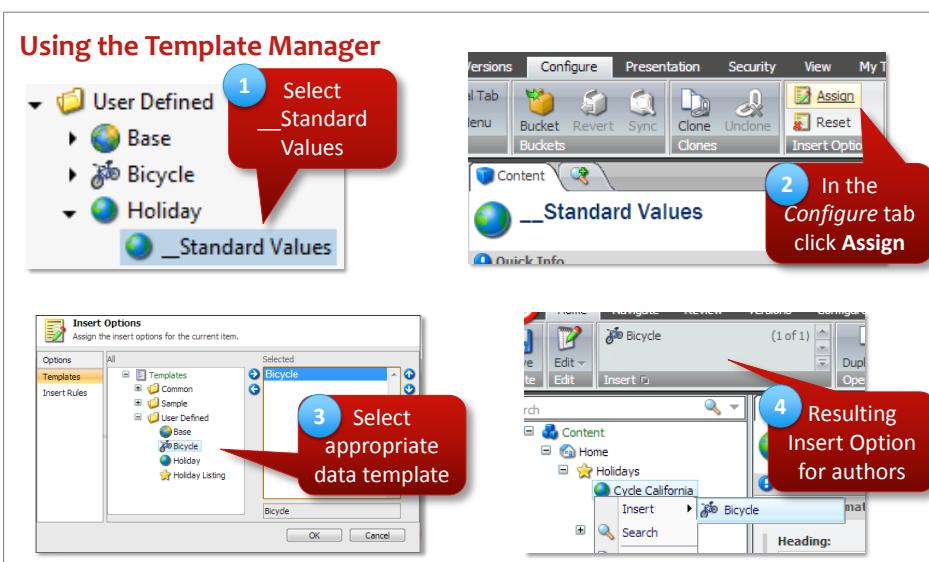
Figure 3-29 Set Insert Option in Template Manager



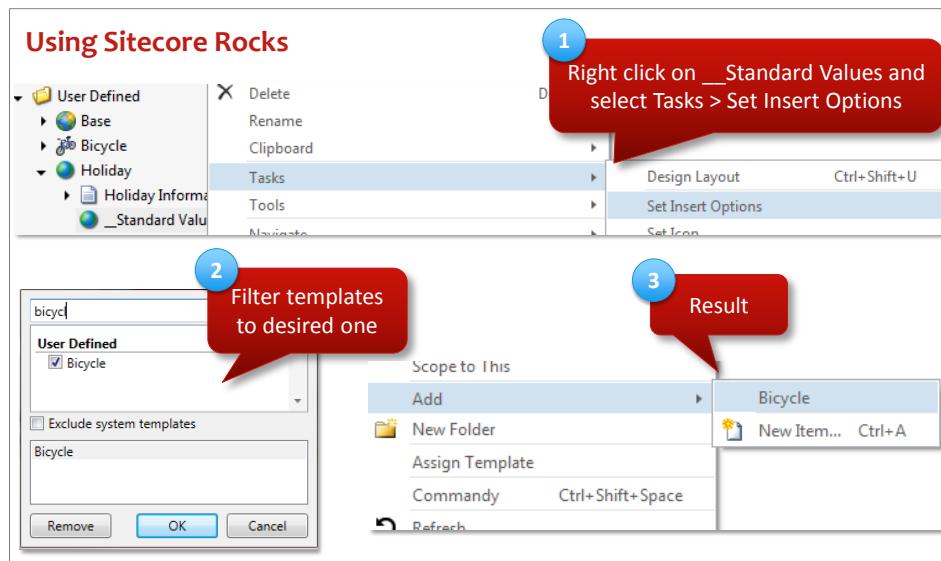
Best Practice

Wherever possible, it is recommended that you set insert options on a template's Standard Value. However, there are instances that you might want to assign insert options directly on an item (e.g., if you have a set of folders with varying insert options and don't want to create a specific template for every single folder type)

Assign Insert Options in Template Manager



Assign Insert options in Sitecore Rocks



Apply – Topic 3.3 – 5 min



Set Insert Options

In the following labs will:

- Set insert options using Sitecore Rocks for the Holiday Listing data templates
- Create a new item using insert options



Figure 3-30 Holiday Listing data template insert options

Lab A. Set Insert Options

1. To assign Holiday to be the Insert Option for the Holiday Listing, navigate to **/sitecore/templates/User Defined/Holiday Listing/__Standard Values** using **Sitecore Rocks** and right-click on the item, select **Tasks > Set Insert Options**, or use **command**

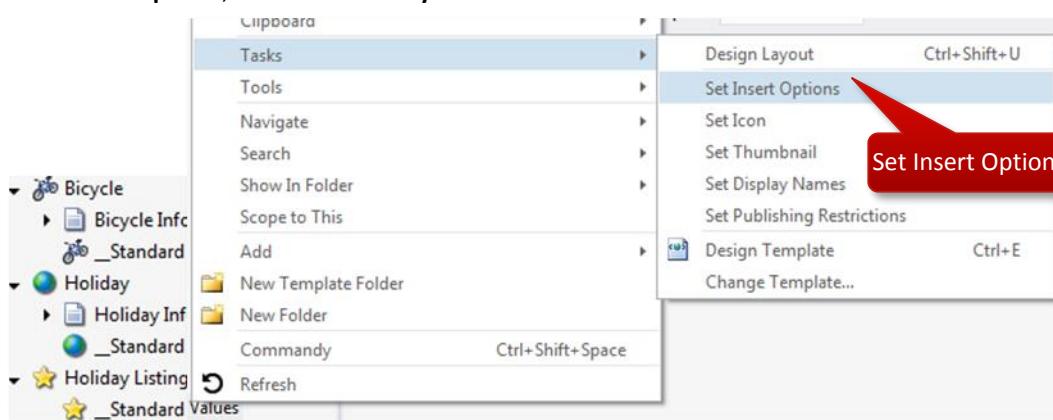


Figure 3-31 Set Insert Option in Sitecore Rocks

2. From the **Insert Options** dialog, filter the list by typing in **holiday** and selecting the **Holiday** data template then click **OK**

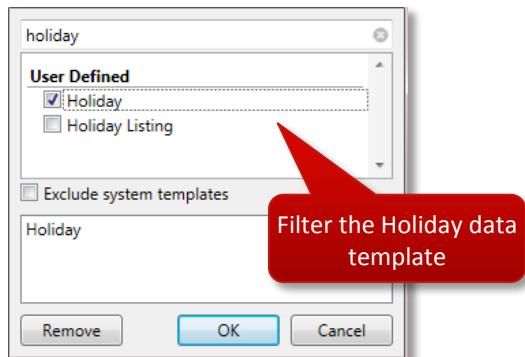


Figure 3-32 Select the Holiday insert option



Knowledge Check

Why would you not have insert options for the Base template?

Lab B. Create items

Create some items using the insert options you have specified

1. Create a new item under Holidays called **Helsinki in 5 days**
2. Create a bike page under that called **Hilkama**, (choose images that you have already uploaded)
3. Publish Sitecore and navigate to those two pages to view your content <http://training/holidays/Helsinki in 5 days> and <http://training/holidays/Helsinki in 5 days/Hilkama>

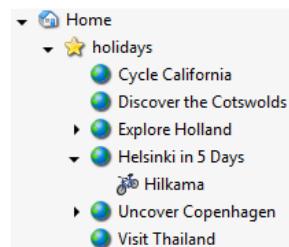
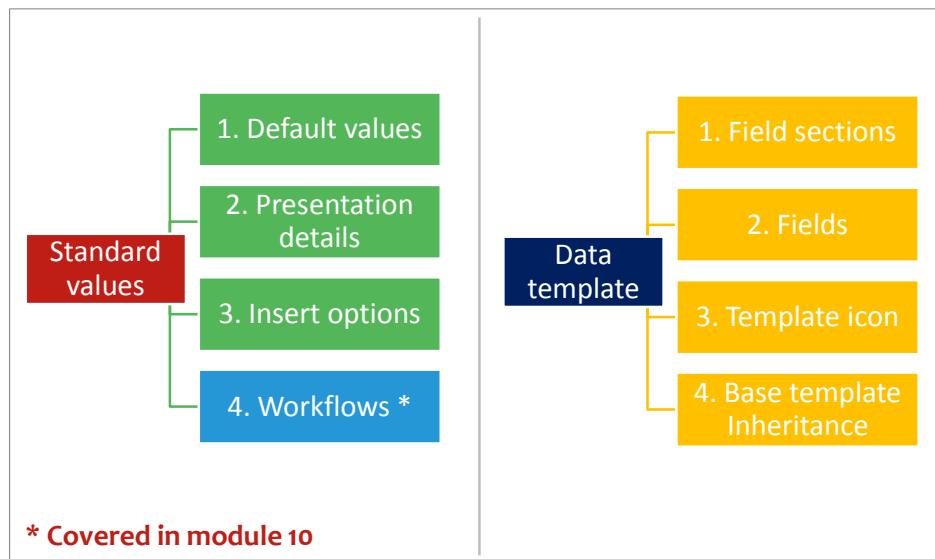


Figure 3-33 Sitecore tree structure

Data template vs. standard values



Review

Insert Options

- | | |
|--|---|
| Q Describe the functionality of Insert Options | Q State where you should assign them? |
| ✓ Insert Options define what type of items can be created under a particular item type | ✓ Template Standard Values as a best practice |

Extend

- **Branch Templates** – allows authors to insert a number of predefined set of items at once
- **Command Templates** – allows an editor to only insert certain items based on logic you setup

Topic 3.4 Other item and template properties

Introduction

Objectives

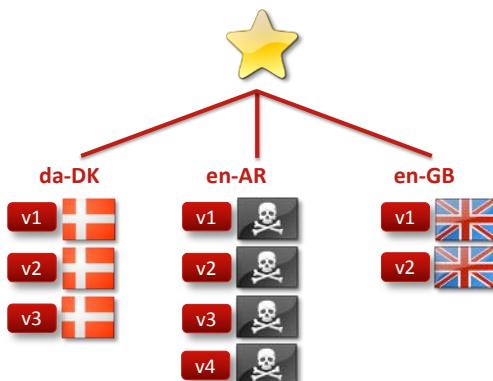
By the end of this topic you will be able to:

- 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

- Items can have any number of **language versions** without any extra work
- Each language version can have any amount of **numbered versions**
- Switch language in browser with `?sc_lang=da-DK`



Important

For content to exist you need to have a language version and at least one numbered version of the content



Business Use Case

You may have a website whose 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 developers involvement and content can start being populated



Demo – Languages and numbered versions

In this demo you will see:

- That you can have content in different languages
- That you can have a number of numbered versions
- How field versioning works

Language and numbered versions

1. Using **Sitecore Rocks**, right-click on 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 **Cycle California** item and switch between the languages, notice there is no content showing for the Danish language, because there is no numbered version under it



Figure 3-34 View language versions

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



Business Use Case

What if travel agents needed a unique ID for every holidayitem? 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 **Holiday 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)



Figure 3-35 New field, shared

2. Enter a default value for the **Holiday ID** field
3. Navigate back to **Cycle California** and switch between the **language** and **number** versions, and notice that the field value is the same
4. A month has passed and you need a new numbered version of the content. Create **V#2** in **Danish**
5. **Switch** between **DK v#1** and **DK v#2** (same field value)



Business Use Case

What if you have a field with a name of a country that you want to keep the same for all numbered versions in a particular language? For example, the country itself doesn't change, but every language may spell a country name differently. This is where unversioned fields come in

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



Figure 3-36 New field, unversioned

7. Enter a default value for the **Country** field
8. Navigate back to **Cycle California** and switch between the **language** and **number** versions, notice the English version is different from the Danish one, which is empty
9. Fill in the **Danish** field with **USA**
10. Switch to **English v#1** (different value)

In conclusion, 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

- **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

- You can have items in Sitecore with no language and no versions; this is because they could have shared fields



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 – Field source

In this demo you will see that, for certain types of fields, you can control what your authors can select. For example, when they browse an image, they don't get the whole media library view, but only a specific folder



Business Use Case

For instance, if you have a bicycle news listing with an image field, you would not want to give your authors a chance to add images to do with furniture, so you would restrict that image fields source to the bicycle news folder in the media library

Field source is also used to populate field content. For example, if you have a folder with a number of content items (such as a list of countries stored somewhere in the tree) you can have a drop list field that has a field source pointing to that folder, resulting in populating the drop list with the names of those items



Tip

Various field types accept various field sources, which will be covered more at a later stage

Field source

1. In the **Content Editor**, change the Image in **Cycle California**, notice the browsing location
2. Navigate to the **Cycle California** data template / then to Base
3. Change the **field source** to a folder in **media library**
4. Go back to **Cycle California** and change an image, or check the standard values and note the change

Field source

Restrict an authors browsing location for certain fields, eg:

The screenshot shows two instances of the Sitecore Media Browser. On the left, labeled "Image field before field source", the browser displays a tree view of the media library with several folders like "Media Library", "Files", "Images", and "System". A red callout bubble points to the "Images" folder with the text "Fill in the Source for the image pointing to the desired location". Below the browser, a content editor interface shows a "Main Content" section with a "Main Image" field set to "Image". The value dropdown contains a GUID: "(15451229-7534-44EF-815D-D93D6170BFCB)". On the right, labeled "Image field after field source", the browser shows a restricted view where only the "Images" folder is visible, and its contents ("banner" and "capture-footer") are displayed.

Use GUID / item path / etc.

Note: An inherited field source cannot be overridden



Important

An inherited field source **cannot** be overridden. If you specify the image field source in a base data template and you are inheriting from it and you wish to override it, you cannot



Demo –Helping authors

There are a number of things that you can do to help the authoring experience, we will demo some of them

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 Holiday data template notice the order of the field sections **Basic Information** and **Holiday Information**

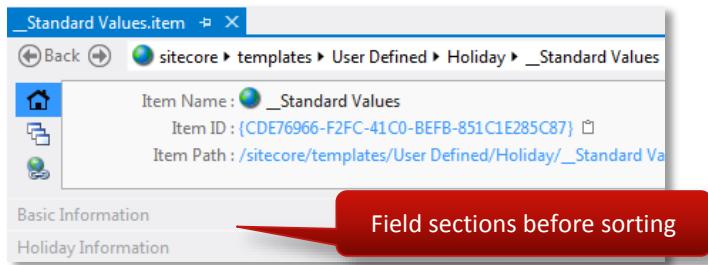


Figure 3-37 Field sections view

2. To change the sorting order we need to go to the definition of those field sections. Navigate to `/sitecore/templates/User Defined/Holiday/Holiday 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 Holiday Information)
6. Open up the `_Standard Values` item for the *Holiday* data template and view that your field sections have now changed order
7. Lastly, right-click again in the **editor** pane and uncheck the **Standard Fields** checkbox

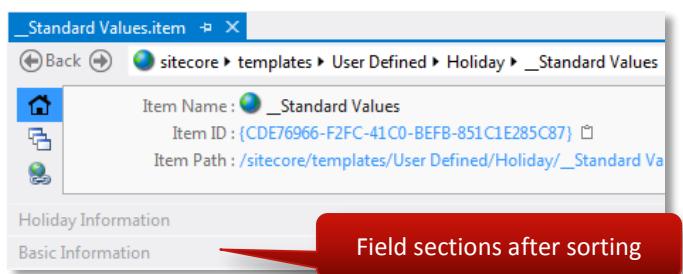


Figure 3-38 Field sections view



Demo – Helping authors

You can also change the names of fields displayed with the use of Title fields

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/Bicycle /Bicycle Information/Type** and right-click to view the standard fields
2. Find the **Help** field section and the **_Short description** field
3. Enter **hybrid / mountain / racing etc.** and the **_Long description** with **this is the long description**
4. **Preview** a bicycle item or create a new bicycle item. You should see fields amended

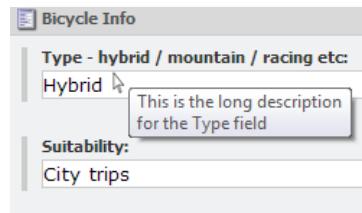


Figure 3-39 Author help fields



Tip

Using the Contend Editor, a quicker way to get to the help fields is to select the field definition items in the tree, but instead of viewing standard fields, you can click on the Configure tab and the help command, this opens up the Help field section

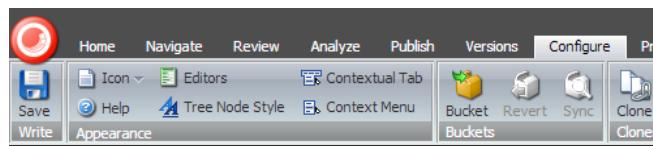


Figure 3-40 Helper field shortcut

Title field

If you want to completely change the field name that is displayed to authors, (for example, change Holiday ID [which developers will still use and see] to Booking reference number [which is what authors see] you can do that with the **Title** field

1. Navigatge to **/sitecore/templates/User Defined/Holiday/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

Sorts fields / field sections for display in the content area

Show standard fields

To view the Sortorder field ensure the standard fields is ticked

Help text

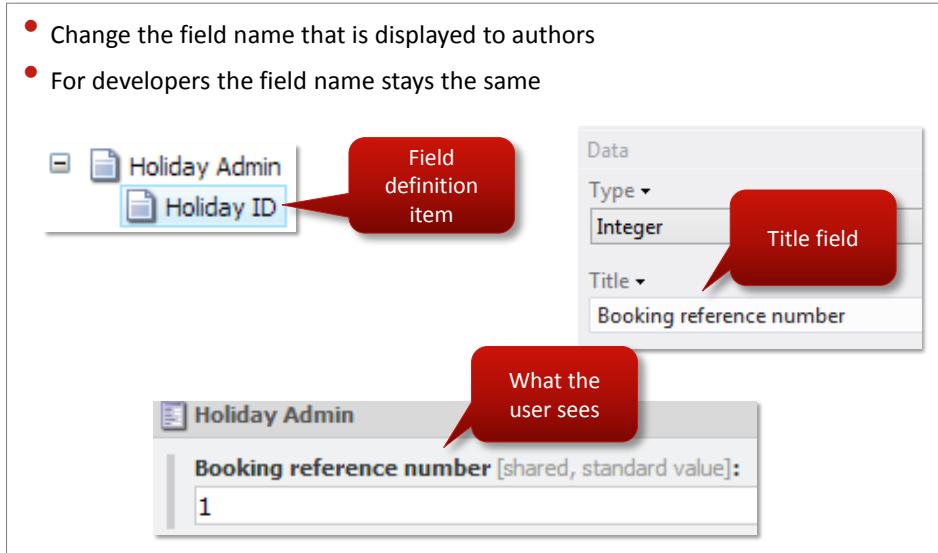
Add helper text to field names for authors

The helper fields are in the standard fields

- You can sort fields and field sections as well as merge field sections using sorting

Title field

- Change the field name that is displayed to authors
- For developers the field name stays the same



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 its Image B. This is done by creating new numbered version of that language in the standard values

Apply – Topic 3.4 – 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
- Changing the field source for an image field
- Altering the sort order for the Bicycle template (OPTIONAL)
- Setting helper text for your bicycle information item (OPTIONAL)
- Changing the title field of a particular field definition item (OPTIONAL)

Lab A. Add a new language and some numbered versions

In this lab you will:

- Add new language to your website
- Add new numbered versions of content
- Create 2 new fields
- Change field versioning

1. Using **Sitecore Rocks**, right-click the **master database** and select **tools** and then select the **Manage** option in the submenu
2. Right-click in the content pane and select the **Add New Language...** option
3. Select the **Danish** language

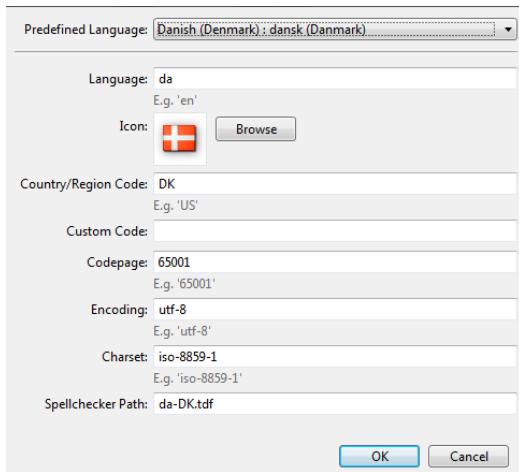


Figure 3-43 Adding Danish language



Figure 3-41 Languages in Sitecore Tocks

4. Navigate to the **Cycle California** item and switch between the **English** and **Danish** language
5. 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
6. The fields you get in a new version you are blank because the field values have not been defined on the standard values for that particular language, **enter** some **values** for the fields

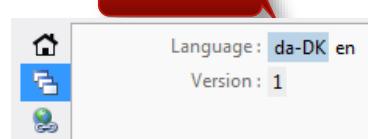


Figure 3-42 Switching languages

7. In the **Holiday 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)

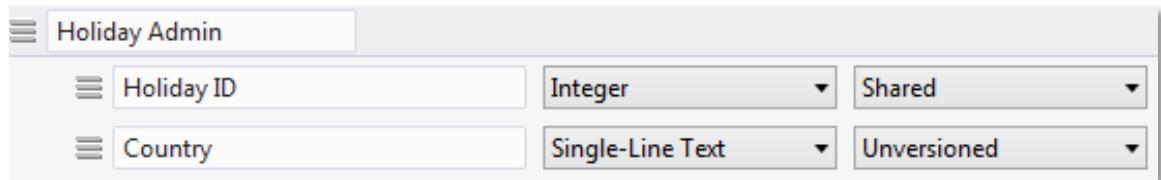


Figure 3-44 New fields, shared and unversioned

8. Enter a default values for the **Holiday ID** and the **Country** field in the Standard Fields
9. Navigate back to **Cycle California** 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
10. Fill in the **Danish** field with **USA**
11. Switch to **English v#1** (different value)
12. Create **V#2 in Danish** by right-clicking in the **Editor** pane, select the **Versions > Add Version** option

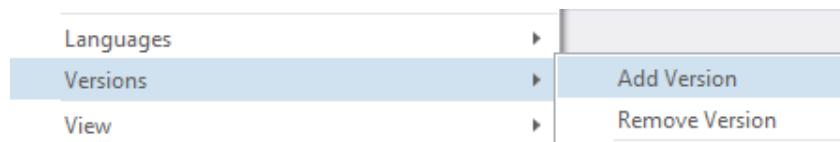


Figure 3-45 Add new numbered version

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

In conclusion 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

Lab B. Field source

In this lab you will specify a specific location of where your authors can select images from

1. In the **Content Editor**, navigate to **Cycle California** at **/sitecore/content/Home/Holidays/Cycle California**
2. Change the image, note the browsing location



Figure 3-46 Media library browsing without field source

3. Close that and navigate to the **Cycle California** data template – use the **Quick Info** section
4. Remember **we are inheriting** that image field from the Base data template, so navigate to **Base**
5. Change the **field source** to be the **location of a folder in media library**

**Tip**

You can use the item path in the tree or the GUID for field source. There are different field sources depending on different field types. We will cover this later in the course

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

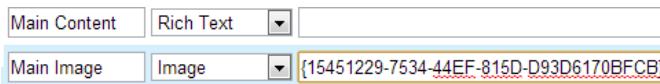


Figure 3-47 Main Image field source

7. Go back to **Cycle California** and browse to an image or check the standard values and note the change of location

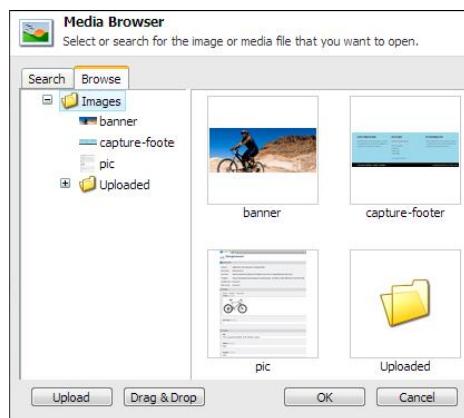


Figure 3-48 Media library browsing with field source

Lab C. Changing the sort order (OPTIONAL)

In this lab you will change the sort order of the Bicycle fields

1. View the **__Standard Values** item for the **Bicycle** data template
2. Note the order of the fields in the **Bicycle Information section** – Type and Suitability, we are going to move **Suitability** above **Type**
3. Double-click the **Suitability** item at **/sitecore/templates/User Defined/Bicycle/Bicycle Information/Suitability**
4. Right-click inside the **editing** pane to view the **Standard Fields**
5. Change the **__Sortorder** field to **20**
6. Double-click on the **Type** item at **/Sitecore/templates/User Defined/Bicycle/Bicycle Information/Type**
7. Change the **_Sortorder** field to **200**
8. Right-click again to deselect the **Standard Fields** checkbox
9. Open up the **__Standard Values** item for the Bicycle data template, and view that your fields have now changed order

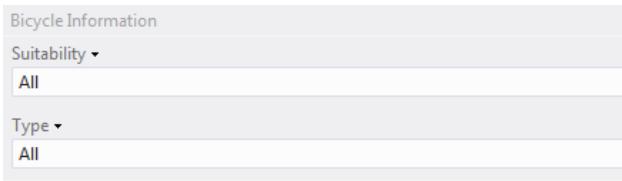


Figure 3-49 Bicycle data template standard values



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 (OPTIONAL)

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/Bicycle/Bicycle Information/Type> and in the ribbon select **View > Standard fields** [Figure 3-5 View Standard fields in Content Editor](#)
2. Find the **Help** field section and the **_Short description** field. Enter *hybrid / mountain / racing etc.*
3. **Save**
4. Double-click the **Suitability** field definition item at </sitecore/templates/User Defined/Bicycle/Bicycle Information/Suitability>
5. Find the **Help** field section and the **_Short description** field, enter *city trips / country lanes / rough terrain etc.*
6. **Save and un-check** the view of **Standard Fields**
7. Look at an existing bicycle item and you will see the help text appear next to the fields

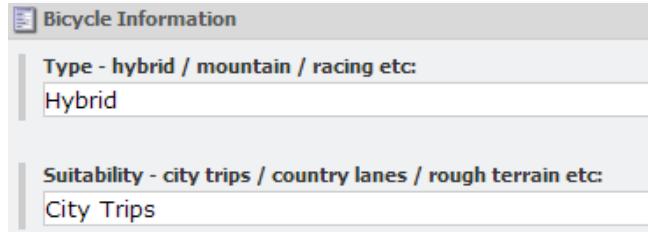


Figure 3-50 Helper fields

Lab E. Title field (OPTIONAL)

In this lab you will change the way the field name is displayed to authors, however your field definition stays 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/Holiday/Holiday Admin/Holiday ID> definition item
2. In the **Title** field (Data field section), enter: **Booking reference number**
3. **Save**
4. **Preview** any item that has that field in the Content Editor

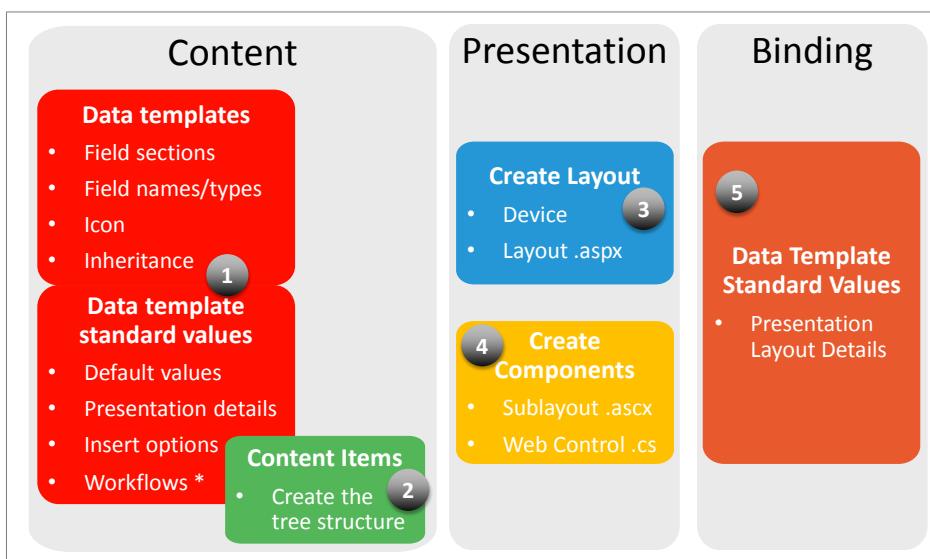
Review

Other item and template properties

- | | |
|---|---|
| <p>Q Items exist in a ?? and ??? version ✓ Language and numbered version</p> <p>Q What does unversioned for field versioning mean? ✓ Single value per language</p> <p>Q What does shared for field versioning mean? ✓ Single value for all numbered versions in all languages</p> <p>Q What is the default field versioning? ✓ Versioned – different values for all languages and numbers</p> | <p>Q What is the field called where you set up field sorting? ✓ Sortorder</p> <p>Q Where is this field found? ✓ Standard fields</p> <p>Q On what type of item do you do field sorting? ✓ On the field definition item</p> <p>Q Name 2 fields that you can use to help a multi-lingual audience ✓ Long Description and Short Description fields - also part of the standard fields</p> |
|---|---|

Summary of Module 1, 2 and 3

Review the steps



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 & 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
- Compare item objects
- Retrieve all children of a particular item

Content

Sitecore.Kernel

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

What do you think the context database is if you are in *Preview* mode when viewing a page?

Sitecore Context

- The Sitecore context, exposed by the static `Sitecore.Context` class, contains information about the **Sitecore installation** and the current **HTTP request**

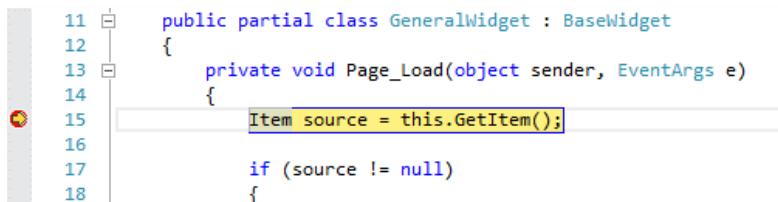


- `Sitecore.Context` assembled when a request is made – some data (e.g., language) retrieved from cookie
- In a typical request, Sitecore determines properties like the **context database**, **context language**, **context item**, and that item's **presentation details**

Debugging

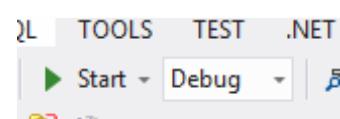
Sitecore is a regular .NET application so you can use the usual Visual Studio debugging techniques:

- Attach to w3wp (tip: if you have more than one instance listed, select all to save you time)
- Step through your code
- F5 not recommended



Demo – Basic API

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



4-1 Select debug configuration

The GetItem() method

Retrieving items

- Items represented by `Sitecore.Data.Items.Item`
- Items can be retrieved using the `GetItem()` method:
`Sitecore.Context.Database.GetItem("/sitecore/content/Home");`
- This method accepts a Sitecore ID or path to an item
- Whenever possible, IDs or paths should not be hardcoded in your solution
- Keep IDs or paths in one place in your solution – e.g. a `References.cs` file

Retrieving the context site's start item

- To get the context site's start item (specified on the `<site>` note in the `web.config`), you can use the following method:

```
string startPath = Sitecore.Context.Site.StartPath;  
Item home = Sitecore.Context.Database.GetItem(startPath);
```

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

1. In **Visual Studio**, open the **code behind** for any existing sublayout (e.g., **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 (suggest 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 (e.g., to ignore any security applied on those items)
5. **Build and deploy** the solution
6. Add a **break point** and attach to **w3wp**
7. In a browser, navigate to a page that uses the sublayout you modified. Step through to see what each property or method returns

Apply – Topic 4.1 – 30 min



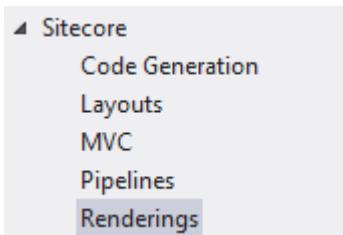
Render item children

In the following lab, you will:

- Build a sub-navigation 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

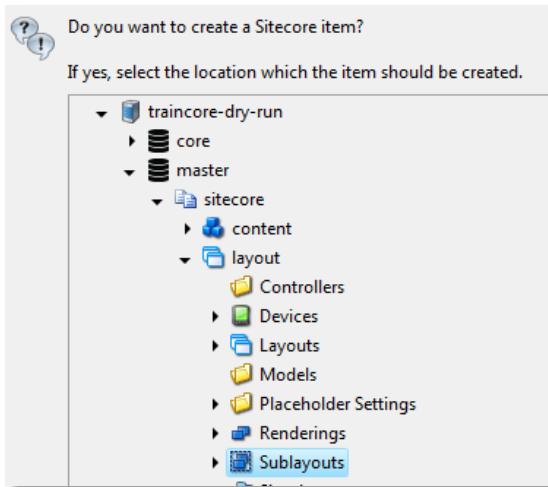
Lab A. Build a Navigation Sublayout

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



4-2 Renderings menu in 'Add New Item'

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



4-3 Choose location for sublayout definition item

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:



4-4 Sub-navigation finished HTML

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

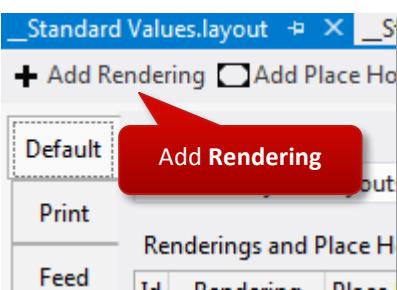


Figure 4-5 Add new rendering

9. Add the **Sub-navigation** sub-layout to the **main** placeholder. Ensure that it is listed below the Introduction sublayout
10. **Save** your work and **preview** the /holidays item, you should see the Sub-navigation sub-layout at the bottom of the page, (if it is not, reset presentation details to standard value)
11. In **Visual Studio**, navigate to the Sub-navigation sub-layout's **code behind**
12. In the **Page_Load** method, **get the context item's children** (holiday 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. **Replace the dummy HTML** if you are **using this sample code**

15. 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 (Note the colon after the #. This automatically escapes any HTML.) to output properties of the object (e.g., the parent ID of the item)

`<%#: Item.ParentID %>`

16. Save and deploy

17. Using a browser, navigate to <http://training/holidays>, and confirm that the Sub-navigation component now appears and that it is listing the child items by **item name**:

Cycling holidays

Our holidays

[Cycle London](#)
[Discover Helsinki in 5 Days](#)
[Discover Copenhagen](#)

Would you like to win a holiday to the Welsh mountains for you and your family? 3 days of [mountain biking and adventures](#). Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi scelerisque dui at nunc scelerisque eu molestie dolor porttitor. Donec aliquam mi eget sem consequat tempor.

Vestibulum ultricies viverra viverra. Proin laoreet velit in erat sollicitudin iaculis. Ut et vehicula leo. Curabitur non dolor elit. Morbi ante volutpat ante sollicitudin posuere. Sed ut nibh ut felis hendrerit vulputate at vel tortor. Suspendisse dui nibh, rutrum nec tempus vestibulum, tristique eget mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut at mollis eros. Suspendisse potenti.

Duis ornare ornare lacinia. Donec pellentesque aliquam urna in facilisis. In hac habitasse platea dictumst. Proin tempor ultricies velit, sit amet aliquam nunc aliquet eget.

Sed non pulvinar orci. In est nunc, adipiscing pharetra ornare consectetur, iaculis non mauris. Suspendisse suscipit ligula non dolor interdum posuere luctus enim accumsan. Morbi consectetur turpis quis turpis ultricies imperdiet. Sed eu pellentesque elit. Ut eget purus libero. Cras tempor sodales euismod. Donec ultricies mauris quis nulla bibendum vitae placerat sapien imperdiet.



[Visit Thailand](#)

[Explore Holland](#)

[Uncover Copenhagen](#)

4-6 The finished page



Knowledge Check

What items would you expect to see if you added the subnavigation sublayout to one of the *Holiday* page items?

Review

Basic API concepts

- | | |
|---|---|
| <p>Q In which .dll can you find the majority of the API?</p> <p>✓ <code>Sitecore.Kernel</code></p> <p>Q When Sitecore makes a request, what is the name of the static class that is assembled?</p> <p>✓ <code>Sitecore.Context</code></p> <p>Q Name four properties that you might get from <code>Sitecore.Context</code>:</p> <p>✓ Context user, language, database, and item</p> | <p>Q When you are looking at a page in Page Editor mode, what is the context database?</p> <p>✓ Master</p> |
|---|---|

- | | |
|--|--|
| <p>Q Name the method used to retrieve items?</p> <p>✓ <code>.GetItem()</code></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> | <p>Q How should you compare two items in code?</p> <p>✓ Using their IDs</p> |
|--|--|

Extend

Content API Cookbook

http://sdn.sitecore.net/upload/sitecore6/64/content_api_cookbook_sc64_and_later-a4.pdf

Data Definition API Cookbook

<http://sdn.sitecore.net/Reference/Sitecore%207/Data%20Definition%20API%20Cookbook.aspx>

Topic 4.2 Item links

Introduction

Objectives

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

- State what the ItemResolver does
- State 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
- Be able to customize LinkManager

Content

The ItemResolver

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

<http://mysite/....>

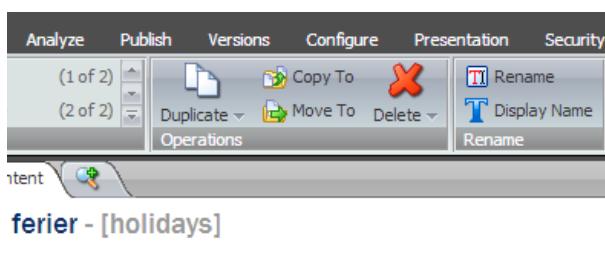
Holidays
 Holidays/
 Holidays.aspx
 Our%20Holidays.aspx
 en-us/Holidays.aspx
 Holidays.aspx?sc_lang=se-SE
 ?sc_itemid={F02D3ACC-45B1-45A1-9BD2-C19B9D81099E}
 ~/link.aspx?_id=F02D3ACC-45B1-45A1-9BD2-C19B9D81099E&_z=z

- **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**'



Demo – Display Name Property

1. Using the **Sitecore Desktop**, open the **Content Editor** and locate the `/sitecore/content/Home/holidays` item
2. Change to another language (e.g., Danish)
3. Select the **Home** tab > **Display Name** command, and change the display name to **ferier**



4-7 Danish display name for the holidays item

4. The content tree will refresh and display the Danish word for **holidays**, in the tree as well as in the content pane. **Note** that the original item name is written in square brackets next to the display name
5. **Change the language back**, notice that the item display name changes again



Knowledge Check

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

Linkmanager for an Item URL

Rendering an Item's URL

- When you are outputting an item's link e.g. in a navigation list, the reverse of `ItemResolver` is required i.e., `LinkManager`
- `LinkManager.GetItemUrl()` which produces an SEO-friendly, human-readable URL
- Do **NOT** use `LinkManager.GetDynamicUrl()` to render links in the website, as it uses GUIDS – for use behind the scenes only

URL Options

- You can send a `UrlOptions` object, with options for customizing the URL
- A useful `UrlOptions` is `options.UseDisplayName` set to `true` so that the URL output will vary **depending on the context language**

<http://training/ferier> or <http://training/holidays>



Demo – Display Name

- Using **Visual Studio**, open a sublayout's code file (e.g., `Introduction.ascx.cs`)
- In the `Page_Load` method, use `Sitecore.Links.LinkManager.GetItemUrl()` to **retrieve the URL of the context item as a string**



Code sample – Retrieving Item URL

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

- Save and deploy**
- Add a **break** point and **attach** to `w3wp`
- Navigate to the <http://training/holidays> page and note the value of the property you created
- In **Visual Studio**, **before** retrieving the context item's URL, create a `UrlOptions` object and set the `UseDisplayName` property to `true`
- 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;
```

- Deploy** the solution and **attach** to `w3wp` again
- In your browser, change the **context language** to the language you previously set the alternative display name in. Use `?sc_lang=` query string the end of the URL in the browser's address bar. Press the **Enter** key on the keyboard
- In **Visual Studio**, step through your code until you get to the `URL` property you created. The `URL` should now read `/ferier`

LinkManager.GetItemUrl() Method

```
Sitecore.Links.LinkManager.GetItemUrl(item)

Item item = Sitecore.Context.Database.GetItem(new
ID("{565A8A17-032F-44AC-B506-B65F27A31241}"));

string url = LinkManager.GetItemUrl(item);

Sitecore.Links.UrlOptions
UrlOptions options = new UrlOptions();
options.UseDisplayName = true;

string url = LinkManager.GetItemUrl(item, options);
```

Customizing the LinkManager

You can customize the LinkManager in the web.config by:

1. Changing the default options
 2. When invoking the methods without a UrlOptions object
 3. Overriding the type completely
- Note that IIS configurations may be necessary

```
<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>
```

Web config
example

Controlling Your URLs

Decide how you want to render links

- Individually or globally

URL resolution handled automatically

- Regardless of how a URL comes in, Sitecore knows which item to map it to

Multiple URLs for a single page can be confusing for:

- Google Analytics
- JavaScript third party services
- Web Log Analyzers













Tip – Media Item URLs

Notice that we are instantiating a `MediaItem` object 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 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, 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 deploy** the solution and browse to the <http://training/holidays> page, confirm that you can now use the sub-navigation list to navigate to holiday sub-pages



4-8 Clickable links

Review

Item links

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

Topic 4.3 Creating, deleting and modifying 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
- Recycle or delete an item
- Make a modified item live on the web database

Content

How to edit an item

There are 5 steps involved in editing an item

1. Retrieve the specific item you want to edit from the master database.
2. Make sure the item is being edited in the context of a user that has the appropriate permissions.
3. Put the item into an editing state.
4. Set the item's field values to the new values (e.g., text values from a form post).
5. Publish your changes if you want them to appear in the web database.



Step 1 – Retrieve the item from the database

- You can edit items in any database, but if you edit items in the web database, changes will be overwritten when you publish from the master
- Perform modifications in the master database and then publish the changes



Sitecore user context

Sitecore uses and extends standard .NET membership

- Wraps standard Microsoft SQL server provider implementations in **custom providers** to offer **extra functionality**

All Sitecore requests happen in the context of a user

- Including **anonymous requests** (extranet\anonymous, the unauthenticated user)

When code is executed, it is bound to the context user

- The user's permissions affect if the code can be run or pages accessed

Extranet\anonymous has no permissions to create/modify content

- So how to store user-created content, like comments or testimonials?
- **Force code to run regardless of context user's permissions**

Step 2 – Security

- Creating, modifying, or deleting an item needs to happen in the context of a user that has the appropriate permissions – if you are not logged in, you are browsing a website as extranet\Anonymous user
- Either **disable** security checks (not recommended) or employ the **UserSwitcher**



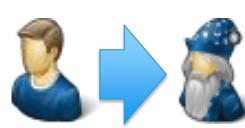
1. Disable

 `using (new SecurityDisabler(user)) { }`
method to disable all security checks – **disables all security**



2. Switch user

`using (new UserSwitcher(user)) { }` method
and perform your actions using an extranet user with the appropriate permissions **method**





Code sample – UserSwitcher

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

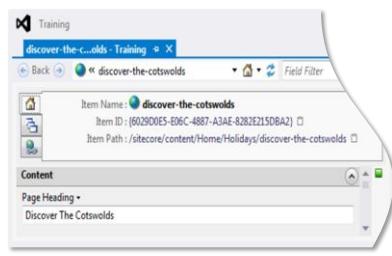
Step 3 – Editing context

Modifying an item means either

1. Changing its properties (like item.Name) or
 2. The contents of its fields (like the page title)
- All changes to an item must take place in an editing context

For each item that you want to edit, do the following:

```
item.Editing.BeginEdit();
// All modifications take place here
item.Editing.EndEdit();
• Calling .EndEdit() saves changes
```



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

The contents of fields

Retrieve field as object or string

The first method will return a **value** or an **empty string** (even if it does not exist):

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

The second method will return a **Field** object, which can be converted to more **specific types** (e.g. **LinkField**)

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

You can get or set the field's **raw value** as a string by using **.Value**:

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



Demo – Outputting Raw Values

1. Using **Visual Studio**, open an existing sublayout's **code behind** (e.g., **Introduction.ascx.cs**)
2. 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;
```

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

Step 4 – Setting Field Values

- Use **.Value** to set the field's new value – using **.Value** is only appropriate for simple field types like Single-Line Text:

```
Sitecore.Context.Item.Fields["Profile Text"].Value = "I enjoy  
long walks on the beach.";
```

Do not use .Value for complex field types – they should be retrieved as objects (Topic 4.4)

- Complex fields include General Link, Image, Multilist, Droplink, etc.



Knowledge Check

Why might it be difficult to set the value of an image or link field in this way?



Demo – Editing an item

1. Using **Visual Studio**, open an existing sublayout code behind (e.g. **Introduction.ascx.cs**)
2. Create a button with an **OnClick event**
3. In the **OnClick** event, retrieve the **Heading** field as a **Field** object and **change** its value by setting the **.Value** property



Code sample – Editing an item

```
using (new SecurityDisabler())
{
    Item item =
Sitecore.Configuration.Factory.GetDatabase("master").GetItem(Sitecore.Context.Item.ID);

    item.Editing.BeginEdit();

    Field heading = item.Fields["Heading"];
heading.Value = "Our fantastic holidays";
item.Editing.EndEdit();

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

4. Save and deploy the solution
5. In a browser, navigate to a page that uses the sublayout you edited (e.g. /holidays), and refresh, notice that the heading of the page changes to what you specified in code

Before you create an item

Before creating an item, ask the following questions:

Where do you want to create the item?

- Get the item's proposed parent from the master database e.g. if you are creating a blog post comment item, the parent might be the blog article, or a comments folder item under the news article



What should my new item be called?

- Use Sitecore.Data.Items.ItemUtil.ProduceValidItemName() on your proposed item; if it's invalid, Sitecore will try to modify it



What template should the new item have?

- You need to know the new item's template's ID

TEMPLATE ID



Best Practices

Do not hard code references to templates. Use a centralized class (e.g., 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?

How to create an item

- Get the item's parent from the master database
- Make sure the item is being created in the context of a user that has the appropriate permissions
- Use the `.Add()` method to create the item, supplying the item's name and template

```
Item parent = Sitecore.Context.Item;  
string name = ItemUtil.ProposeValidItemName("Comment-5");  
TemplateID templateID = new TemplateID(new ID("{565A8A17-  
032F-44AC-B506-B65F27A31241}"));  
Item newItem = parent.Add(name, templateID);
```



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);
```

Deleting Items

An item can either be deleted or recycled using methods on the item object:

Recycle an Item

```
item.Recycle();
```

- if you recycle an item, it goes into the recycling bin provided the `RecycleBinActive` setting has been set to true in web.config:

```
<setting name="RecycleBinActive" value="true" />
```



Deleting an item

```
item.Delete();
```

- Alternatively, you can delete an item – note that by default calling `.Recycle` will call `.Delete` if `RecycleBinActive` has been set to false



Demo – Creating Items

1. In Visual Studio, locate an existing sublayout (e.g., `Introduction.ascx`)
2. Create a button with an `OnClick` event
3. In the `OnClick` event, create a new item based on the `Holiday` template beneath the `/sitecore/content/Home/holidays` item:



Code sample – Creating items

```
using (new SecurityDisabler())
{
    Database master =
Sitecore.Configuration.Factory.GetDatabase("master");
    master.GetItem("/sitecore/content/home");

    Item item = Sitecore.Context.Database.GetItem(new
ID("{565A8A17-032F-44AC-B506-B65F27A31241}"));

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

    string name = ItemUtil.ProposeValidItemName("Cycle the
Cotswolds");

    item.Add(name, templateID);
}
```

4. **Save and deploy** the solution
5. Using a browser, navigate to the new item you just created

Where are My Changes?

When an item is created, deleted, or edited using the API, those changes must be published

- Publish **programmatically** (best practices covered in Module 10)
- Wait for a **scheduled publishing task**
- Items (e.g., new comments) may go into an **approval process** and have to be **reviewed and approved** by a moderator (Module 10 – Workflow)

Apply – Topic 4.3 – 10 min



Create and edit items

A common example of when you might want to create new Sitecore items programmatically is a **commenting system** (e.g., allowing anonymous visitors to comment on particular holidays 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 Holiday template standard values

Wire the comments form up to create new items based on the Comment data template beneath the context holiday, populated with form data

Lab A. Create comment data template

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

| Field Name | Field Type |
|----------------|------------------|
| Comment | |
| Comment Author | Single-line text |
| Comment Text | Rich Text |

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



Knowledge check:

In this instance, why might you not need to assign the *Comment* data template as an insert option on the Holiday data template standard values?



Knowledge Check

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

Lab B. Create a comments form sublayout

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

Lab C. Create comment items from form inputs

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

1. Open **Comments Form.aspx.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.aspx.cs`). Do not replace the entire .cs file with the contents of this sample; it is **only** the **OnClick method**
 - For the remaining steps in **Lab C** and **Lab D**, comments are available in `CommentsForm.aspx.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()`). Since the sublayout appears on a Holiday item, the context item will be a holiday



Knowledge Check

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

3. Having retrieved your intended comments **parent** (the context holiday 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 holiday item you retrieved from the master database to **insert a new comment item as a child**, this method will return an `Item` object
5. Set the **template parameter** to the **Comments template** you created in **Lab A**



Tip

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

6. Set the **name parameter** to the **current date/time in ISO format**. Use the `Sitecore.DateUtil.IsoNow()` method
7. **Save and deploy** the solution
8. **Browse** to a holiday page and **submit** the form
9. Confirm that child items are created under the context **Holiday** item, note that the 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 item URL**

```
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 and deploy** the solution
 5. **Browse** to a **holiday** page, fill in and submit the form, and confirm that the child items created under the context holiday now contain your form content

Review

Creating, modifying and deleting items

| | |
|--|---|
| Q Sitecore security is an extension of... | Q How can you force a piece of code to run even though the currently logged in user does not have the appropriate permissions? |
| ✓ Standard .NET membership | ✓ Use the SecurityDisabler or UserSwitcher to run in the context of a user that does have permission |
| Q All code is executed in the context of... | |
| ✓ The current user | |
| Q What affects whether or not a piece of code will run? | |
| ✓ That user's permissions | |
| Q What permissions does extranet\anonymous lack by default? | |
| ✓ Permission to create or edit items | |

- | | |
|--|---|
| <p>Q When creating or editing an item, the code must be executed with the appropriate security rights. Name two ways that this can be done.</p> <p>✓ UserSwitcher or SecurityDisabler</p> <p>Q How do you put an item into an editing state?</p> <p>✓ <code>Item.BeginEdit()</code></p> <p>Q Why should you create / edit items in the master rather than the web database?</p> <p>✓ Web overwritten by publish</p> | <p>Q What property do you use to update?</p> <p>✓ UserSwitcher or SecurityDisabler</p> <p>Q Why should you not output a field's value straight to the screen?</p> <p>✓ Not editable in Page Editor and complex fields contain custom XML</p> <p>Q Which field types are suitable for editing using the <code>.Value</code> property?</p> <p>✓ Simple text fields – e.g. Single-Line Text</p> |
|--|---|

Extend

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

Topic 4.4 Working with complex fields

Introduction

Objectives

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

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

Content

Raw values

Ultimately, editing an item will change its raw value

- All item field values are stored as **strings** – they are referred to as **raw values**

For complex fields (e.g. image, link), the raw value is custom XML

- In theory, you could build the custom XML as a string
- In practice, easier to **retrieve the specific field type object** you require and use its **properties and methods** to modify the field (complex fields will be covered later)

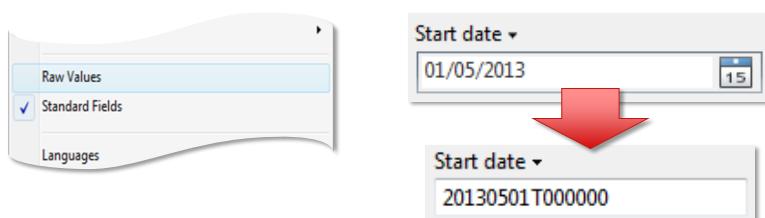
Do not output raw values directly to the browser

- They will not be editable in Page Editor and will be in a format that neither browser nor visitor will understand

Viewing raw values

You can view the raw values of an item's fields:

- In **Sitecore Rocks**, right-click in the content pane and clicking '**Raw Values**'
- In the **Content Editor**, tick the **Raw Values checkbox** in the View tab





Demo – Raw values

1. Using **Sitecore Rocks**, enable **raw values** by right-clicking in the content pane and selecting the **Raw Values** option
2. Navigate to an item based on the **Holiday** data template
3. Note the contents of the **Start date** (ISO date and time) and **Main Image** (custom XML) fields

Field types and their corresponding objects

Field type determines...

- Sitecore raw value
- Sitecore API field class
- Sitecore control

How do you cope with complex raw values?

- Some raw values are **too complex** to set as a `.Value` string – use **custom XML**
- Fields can be retrieved as **objects**
- `Sitecore.Data.Fields` namespace
- Use field objects to **get or set** properties

Field object

Base class

- The base field object class for all fields – **Field**
- **Simple fields** (e.g. Single-Line Text) are set using `.Value`

Raw value

```
<p>Our <strong>favourite</strong> holidays!</p>
```

Sample code

```
Field headingField = item.Fields["Heading"];
headingField.Value = "Our holidays";
```



Field object cont.

Namespace

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

Appropriate field controls

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

Supported fields

- Single-Line Text
- Number, Integer
- Multi-Line Text
- Password
- Dropdown, Grouped DropDownList

ImageField object

Raw value

```
<image mediaid="{743C2FAD-0E01-426E-8972-2CCFEBCD20F9}"  
mediapath="/BaseCore/Images/Holiday Image"  
src="~/media/743C2FAD0E01426E89722CCFEBCD20F9.ashx" />
```

Sample code

```
MediaItem mediaItem = new MediaItem(item);  
ImageField imageField = item.Fields["Banner"];  
imageField.MediaItem = mediaItem;  
imageField.Alt = "Holiday snap";
```



ImageField object cont.

Namespace

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

Appropriate field controls

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

Supported fields

- Image



Best Practice

Do not manually construct an `` tag from an `ImageField` object and render it to the page as the values will not be editable in Page Editor

LinkField object

Raw value

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

Sample code

```
LinkField linkField = item.Fields["Search Engine"];  
linkField.TargetID = item.ID;  
linkField.Text = "Google";  
linkField.LinkType = "internal";  
linkField.Url = LinkManager.GetItemUrl(item);
```



LinkField object cont.

Namespace

Sitecore.Data.Fields.`LinkField`

Appropriate field controls

```
<sc:Link Field="Referrer" runat="server" />
```

Supported fields

- General Link



Best Practice

Do not manually construct an `` tag from a `LinkField` object and render it to the page. The values will not be editable in Page Editor

Rendered vs. raw value

For complex field types, the raw value is custom XML – it is not fit for browser interpretation

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

Rendered field content

Rendering is outputting a field's content onto a page as the user should view it – e.g., the contents of an image field becomes an **HTML image tag**:

```

```

FieldRenderer.Render()

When rendering to the browser, always use the Render() method:

```
FieldRenderer.Render(Sitecore.Context.Item, "Main 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

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! Hard-coded 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) |



Best Practice

ALWAYS GO THROUGH THE FIELDRENDERER METHOD when 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** (e.g., the **Class** on a link or the **Width** on an **image**)
- **Links in Rich Text fields** to be transformed into **SEO-friendly URLs**



Demo – Rendering Field Values

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 (e.g. max width and max height **[mw=200&mh=100]**)
4. Show that these parameters are represented as properties on an image control

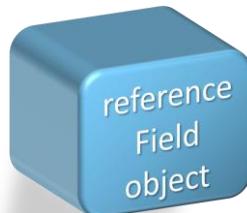
ReferenceField object

Raw value

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

Sample code

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



ReferenceField object cont.

Namespace

Sitecore.Data.Fields.ReferenceField

Appropriate field controls

None; this is not a rendered field type

Supported fields

- Droplink
- Droptree
- Grouped droplink

MultilistField object

Raw value

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

Sample code

```
MultilistField listField = item.Fields["Related Bikes"];  
  
// Retrieve items  
IEnumerable<Item> i = listField.GetItems();  
// Add or remove items  
i.Add(bicycleItem);  
i.Remove(bicycleItem);
```



MultilistField object cont

Namespace

Sitecore.Data.Fields.MultilistField

Appropriate field controls

None; this is not a rendered field type

Supported fields

- Checklist
- Multilist
- Treelist
- Treelist-Ex

Other field object types and controls

Examples of other commonly used field objects and controls

| Sitecore.Data.Fields | Controls |
|----------------------|---|
| CheckboxField | |
| DateField | <sc:Date Field="Date" runat="server" /> |
| FileField | |
| GroupedDropdownField | |
| NameValueListField | |

Generic field renderer control

```
<sc:FieldRenderer FieldName="Heading" runat="server" />
```

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 Holiday template

Lab A. Populate a comment link field

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

| Field Name | Field Type |
|------------------------|--------------|
| Comment | |
| Comment Author Website | General Link |

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

| Object property | Object value |
|-----------------|---|
| .URL | The contents of the link input (e.g. AuthorWebsite.Text) |
| .Text | The same contents of the link input (http://www.google.com/) |
| .Target | "_blank" |

| | |
|-----------|------------|
| .LinkType | "external" |
|-----------|------------|

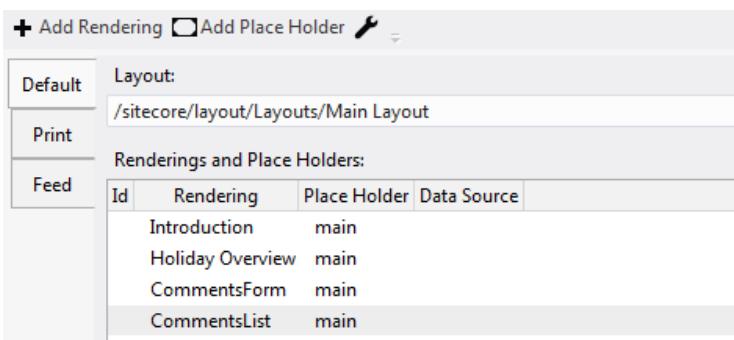
5. **Save and publish** Sitecore and **deploy** the solution
6. Browse to a **holiday page**, fill in and submit the form
7. Switch to **Sitecore Rocks** and locate the newly created comment – confirm that the **link field** has been populated with the form content

Lab B. Render list of comments

In the following lab, you will:

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

1. Create a new sublayout called **Comments List**. Paste **sample HTML for the comments list** in the **student resources folder** (*WND Labs > Module 4 > Topic 4.4 > Lab B > HTML > campaign-page-comments.html*) Remember to **append** this sample HTML to the content already in the **Comments List** sublayout. Do not overwrite the directives and include it at the top of the file
2. Add the new sublayout to the **standard values** of the **Holiday data template**, below the Comments Form sublayout:



4-9 Design layout interface

3. **Save and publish** Sitecore and **deploy** the solution
4. **Browse** to a holiday page to confirm that the sample HTML appears
5. **Replace** the dummy HTML with a repeater (do not delete the directives and include it at the top of the file). A sample repeater is available in (*WND Labs > Module 4 > Topic 4.4 > Lab B > Code > CommentsList.ascx*)
6. Open the CommentList.ascx **code behind**
7. In the Page_Load method, retrieve the context item's **children** and **filter it** so that only items with the Comment data template are included
8. **Bind** this list to the repeater



Tip

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

9. In the repeater, use **Sitecore controls** (e.g. <sc:Text />) to output the following field values:
 - a. Comment Author (**Text**)
 - b. Comment Text (**Text**)
 - c. Comment Author Website (**Link**)
 - d. Date created (**Date** – __Created field – use `Sitecore.FieldIDs.Created` to retrieve field and considering experimenting with the Format attribute)
10. 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 (note the colon after the #):
`DataSource="<%#: Item.ID %>"`*

11. Save and **deploy** the solution, then use a browser to preview a holiday 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/>

4-10 An example of a single comment



Knowledge Check

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

Review

Working with complex fields

| | |
|---|--|
| Q Why should you never output raw value to screen? | Q What method should you use to render the contents of text, date, image and link fields to the screen (and why)? |
| ✓ Not editable in Page Editor, complex field types will not make sense – e.g., image | ✓ <code>FieldRenderer.Render()</code> – Page Editor support, transforms custom XML, transforms Rich Text links |
| Q Name the best suited field object for the following field types: single-line text, treelist, droplink, general link: | Q Why can you not render a multilist field to the page using <code>.Render()</code>? |
| ✓ Field, MultilistField, ReferenceField, LinkField | ✓ It contains IDs, not readable content |

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

It's important to realize that not all items are actually pages. Many items are just pieces of content. You can choose which item or items a component displays



Business use case

- What if you have a quotes component that is used on a number of different pages that displays the same quote? Without datasources you would have to define those quote fields on every single item that you wish 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 in the bottom right of the page and the top left? Without datasources you would have to define several fields with similar names so that you can target each field individually, as well as several similar components (e.g., Quote Title1, Quote Title2, Quote Author1, Quote Author2 resulting in field duplication). With datasources you have one location in the tree with your quote items. Your quote component accepts a datasource of one or more of those items and displays them, resulting in no duplication

1. Using TrainCore, in **Page Editor** design mode change the data source of a widget component to another item under /global/reusable content using the floating toolbar command
2. Add another widget to the *widgetcontainer* placeholder
3. Change its datasource



Knowledge Check

What kind of content can be “global Content?”

4. Set the datasource to one of the widget items
5. Notice that we can reuse this component and this content over and over again, filling it with different datasources

Demo recap

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

Home Page.aspx

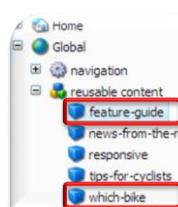
The screenshot shows a web page titled "Home Page.aspx". On the left, there's a "General Widget" containing a red bicycle image and the text "WHICH BIKE?". Below it is a link "Let us help you pick a bike". On the right, a modal window titled "Select the Associated Content" is open, showing a tree view of Sitecore content items under "sitecore / Content / sitecore-cycling-holidays / Global". A specific item, "feature-guide", is selected and highlighted with a red box. The modal also contains a preview of the "SITECORE FEATURE GUIDE" page, which features a cyclist and the text "Find out more about where particular Sitecore features have been demonstrated on this site. Read the feature guide".

Benefit?

- Reusable content and /components and prepares for DMS implementation

Reusable content

- Content stored outside of the Home node in a global section



Two Sitecore widgets are shown in configuration mode. The left widget has a heading "Widget Heading: Not SURE Which bike?" and displays a red bicycle image with the caption "Dimensions: 460 x 345 Alternate Text: 'greyhound' (Default)". The right widget has a heading "Widget Heading: Sitecore Feature guide" and displays a cyclist in a forest with the caption "Dimensions: 460 x 345 Alternate Text: 'mountain bike' (Default Alternate Text: 'Female cyclist in a forest')". Both widgets have "Widget Content" sections with descriptive text.

- Items have no presentation assigned
- Items have the same fields with different content

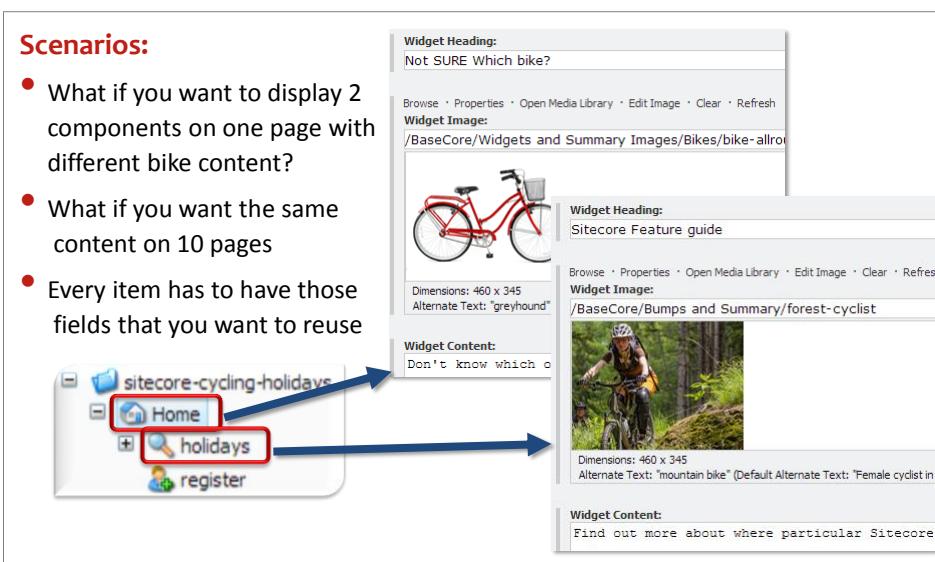
Why have a datasource?

A screenshot of a website with multiple pages sharing a common "datasource". The top page shows a cyclist and the URL "http://mysite.com". The middle page shows a search interface with the URL "http://mysite.com/holidays". The bottom page shows a "Component Heading Image Content" section with the URL "http://mysite.com/copenhagen". All three pages reference a single "feature-guide" item in the Sitecore navigation menu, which is highlighted with a red box. Arrows point from the "datasource" label to each of the three pages.

What happens if you don't use a datasource?

Scenarios:

- What if you want to display 2 components on one page with different bike content?
- What if you want the same content on 10 pages
- Every item has to have those fields that you want to reuse



In summary

Without datasources:

- You are limited to how many components you can have per page and you would have to **define fields with similar names per item** (e.g. Widget Heading 1, Widget Heading 2 etc.) resulting in **field duplication** as well as **component duplication**
- You are forced to **repeat content** all over the tree - if you have the same bit of content (e.g. latest news or best bike this year) that should be displayed on multiple pages you would have to **define those fields and repeat** the same **content** for **every item** resulting in **content duplication**



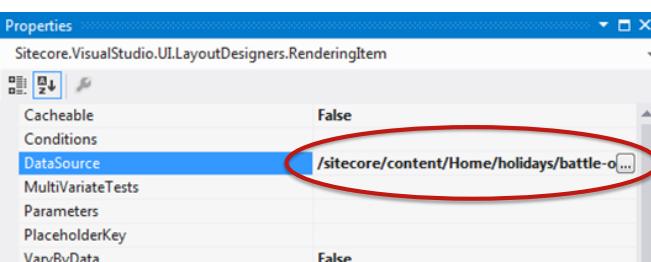
With datasources:

- Content is stored **once** somewhere in the tree
- Component retrieves those fields
- **Content and components are reused and DMS ready**



Configuring a datasource

- When you place a component in a placeholder, you can assign a **datasource**
- A **datasource** is a reference to one or more items in the content tree
- By defining a **datasource**, you are giving the component an option to **take data from an item that is not the context item** – e.g. you could create a reusable ‘featured bike’ component that draws data from a particular bike item, and place that component on other pages to promote crosslinking





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

Using the API for web control data sources

- You explicitly tell a component to use the datasource item it has been provided
- Each type of component has different methods of accessing the datasource

Web Controls

- All web controls inherit from `Sitecore.Web.UI.WebControl`
- This class has an inbuilt method that retrieves the datasource item

```
this.GetItem()
```

If the datasource is not set this method will return the context item instead



TIP – XSLTs

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

Using the API for sublayout data sources

- 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 data source

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 (an Item object) in the code-behind:

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

```
WidgetHeading.Item = myItem;
```

Or

- Set `DataSource` property on the control itself (item path or GUID as a string):

```
<sc:Text Field="Widget Heading"
DataSource="/sitecore/content/my-item" runat="server" />
```



Demo – Datasource API

Let's look at how you do that in code:

1. On the bank site, change the datasource of the **Holiday Overview** component, make no changes in the content
 2. Click the **Save** button, and then switch to Visual Studio
 3. In the codebehind, file for the component code your two fields: **start date** and **price per person** to accept the datasource. Otherwise, if no datasource is supplied, then fall back to the context item
 4. A sublayout is a Sitecore construct—it's a wrapper around ASP.NET controls (ASCX). You need to access the properties of the Sitecore sublayout by looking up the control hierarchy as ASP.NET doesn't know what a sublayout is
 5. Using `.parent` ASP.NET will look to its parent control, find something with a type of sublayout, THEN we can access the properties (like datasource) after you cast it as a `Sublayout`
 6. Use the `Sitecore.Web.UI.WebControls.Sublayout` object
- You can then check if the component has a datasource. If it does, then display it, otherwise return the context item



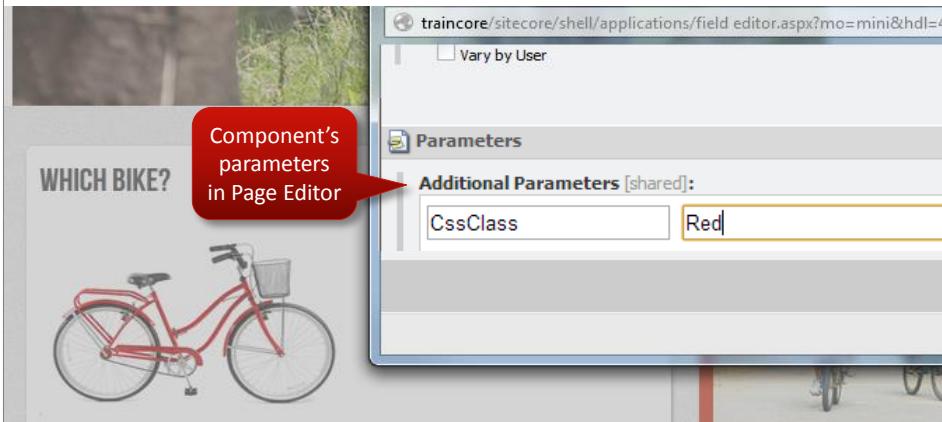
Tip

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

7. Build and deploy your solution
8. Switch to the **Page Editor**
9. Verify that content is appearing from another item
10. Change the datasource again

Component parameters

- Parameters allow properties to be set **per instance** of a component
- For example add two 'General Widget' components to a page, the 'CssClass' parameter could be set to 'Red' on one instance, and 'ornate' on another



Uses for parameters

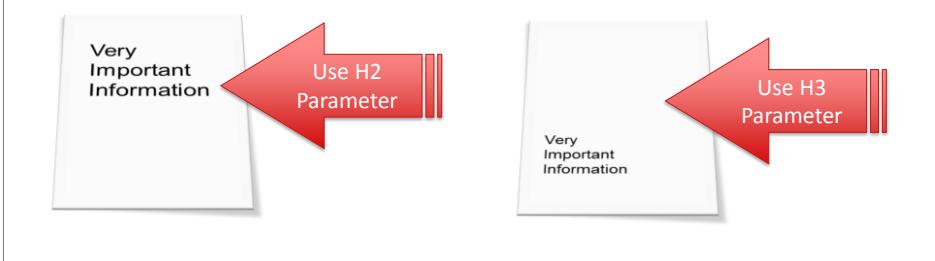
Per-instance styling

- For example specifying a class that a widget should use



Per-instance formatting

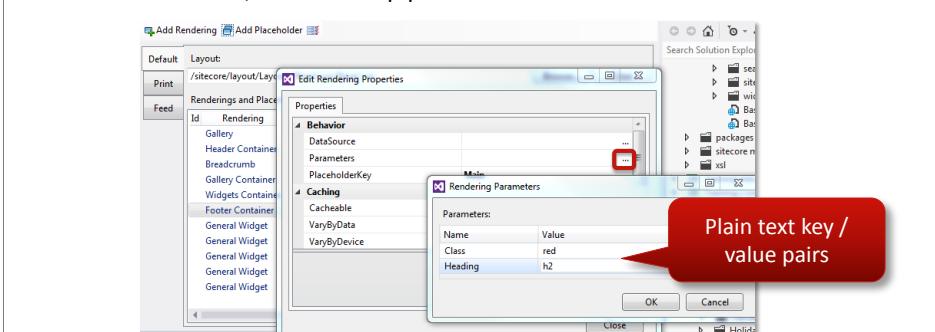
- For example for SEO purposes the same data has greater or lower importance on a page and so needs an h2 or an h3 heading depending on where it is located



Specify component parameters

When you add a component to a page, the properties window has a Parameters [...] field

- The key / value pairs are specified as plain text
- The value should take the same format as a Sitecore raw value – e.g. to reference a number of items, use a list of pipe-delimited GUIDs



Retrieve Parameters Using the API

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 as Sublayout;
```

```
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

1. Change the color for the *Holiday Overview* table from grey to red
2. The color red already exists in our CSS, we need to tell our sublayout to pick up any parameters that are set on the component
3. In Page Editor, go to the component properties and add a **CssClass Red** into the *Holiday Overview* component
4. In *Holiday Overview.ascx.cs*, get the component parameters as a string (use `((Sublayout)Parent).Parameters`).
5. Convert the string to a `NameValueCollection` object using the `Sitecore.Web.WebUtil.ParseUrlParameters()` method
6. Extract the value of the `CssClass` key that you specified earlier and set it to a public property
7. Output that public property in *Holiday Overview.ascx* in the following location:



Code sample

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

8. Click the **Save** button and deploy the solution
9. Switch to the **Page Editor** and verify the red table

Presentation details on standard values or item?

Presentation details on standard values

- For presentation details to appear on all items based on a particular template, define them on the data template's standard values
- Applies to **Datasources** too, when all items based on a template must have a particular component with a particular **Datasource**, this information would be defined on the data template's standard values

Presentation details on directly on an item

Set the presentation details on the item if...

- Component is to appear on **one page** with a **particular datasource** – define component **and** datasource on the particular item
- Component to appear on **every page** with a **variable datasource** – define component on standard values and set datasource on the particular item

Presentation changes to a data template's standard values will override the items presentation unless...

- Presentation changes have been overridden at item level since per-item presentation detail changes are stored as **layout deltas** (next topic)

Apply – Topic 5.1 – 40 min



Build a featured bike component

In these labs you will build a reusable Featured Bike component that accepts a bike data source and can be styled using parameters

Lab A. Create Component and Get the DataSource Item

1. Create a sublayout called **Featured Bike**
2. There is sample HTML available in the **student resource folder** (WND Labs > Module 5 > Topic 5.1 > Lab A > HTML > featured-bike.html)
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 and deploy** the solution
7. Using **Sitecore Rocks**, add the **Featured Bike** component to the **main** placeholder on the **holidays** item, (in this case not in the standard values – you might not want the featured bike component to appear every time you create a holiday)

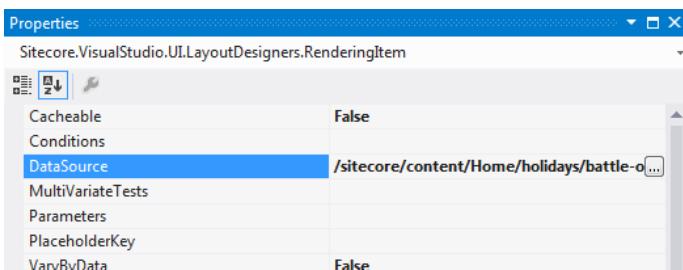


Figure 5-1 Set datasource in Sitecore Rocks

8. Select the **sublayout** and open the **Properties** window / press the **F4** key / double click the sublayout
9. Set the **DataSource** property to any bicycle item
10. **Save**
11. Preview the **holidays** item
12. **Confirm** that the **component** is **outputting data** from the **context item**, not the bike item; we have to explicitly tell the controls to output data from the data source
13. In the **Page_Load** method of the featured **Featured Bike.ascx.cs** file, get the component's datasource using the following code:



Code sample – Datasource

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

14. In the **Featured Bike.ascx.cs** file, set the **.DataSource** property of each control to the **datasource item**

15. Click the **Save** button to publish the solution
16. **Preview** the page and confirm that the **content** is now **coming** from the **component's data source** item rather than the context item

Lab B. Retrieve the *CssClass* parameter value

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

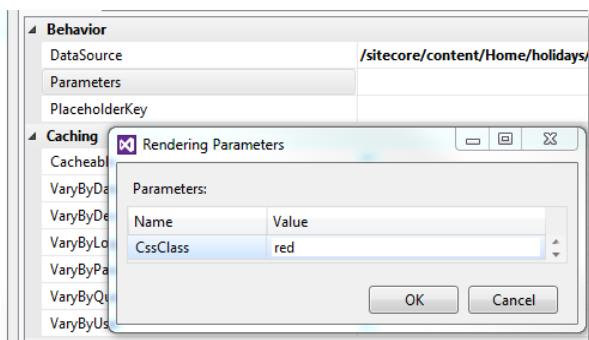


Figure 5-2 Add a property to a component in Sitecore Rocks

5. In **FeaturedBike.ascx.cs**, get the component **parameters** as a **string** (use `((Sublayout)Parent).Parameters`)
6. **Convert** the **string** to a **NameValuePairCollection** 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 Bike.ascx** in the following location:



Code sample – Output public property

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

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

| | |
|------------------|---------------------|
| Date | 05/07/2013 10:52:00 |
| Price per person | £1000 |

Figure 5-3 Resulting table

Review

Datasources

Q What is a datasource?

✓ Takes information from another item that is not the context item and displays it in a component

Q What property do you have to use on the sublayout object to get the datasource?

✓ `.datasource`

Q How do you force Sitecore controls to target an item in code behind?

✓ By setting the `.Item` property

Q How do you force Sitecore controls to target an item in the markup?

✓ Setting the `DataSource` property on the control itself

Q What do parameters allow you to do?

✓ Allows properties to be set per instance of a component

Q Which property do you use to retrieve parameters?

✓ `.Parameters` property

Q What utility converts parameter lists to `NameValueCollection`s?

✓ `ParseURLParameters`

Q Why would you set the datasource on a component on the item itself?

✓ When your component and its datasource appear on one page

Q Why would you set the component and the datasource on the Standard Values?

✓ When you want all items to have that component with that particular datasource

Q Why would you set the component on the standard values and override the datasource on the item?

✓ When your component has to appear on every page but has a different datasource each time

Topic 5.2 Layout Deltas

Introduction

Objectives

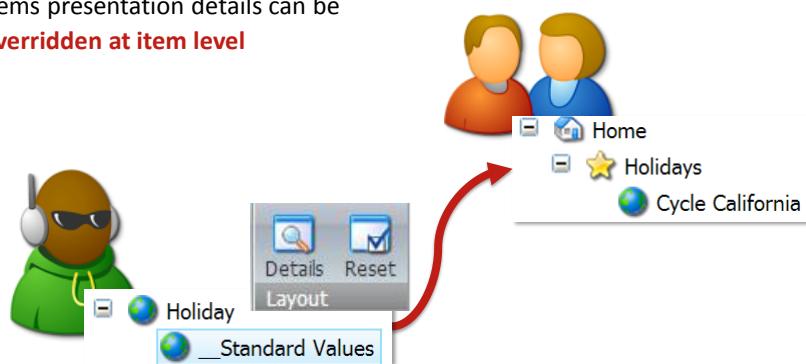
By the end of this topic you will be able to:

- Describe how Sitecore stores presentation details
- Define the purpose of a Layout Delta
- State how the LayoutResolver works with presentation details

Content

What are layout deltas?

- If a developer makes presentation changes on the standard values the item gets updated and not overridden
- Items presentation details can be **overridden at item level**



How and where are layout deltas stored?

- **Presentation** details are **stored** in the **Renderings** field (standard fields)

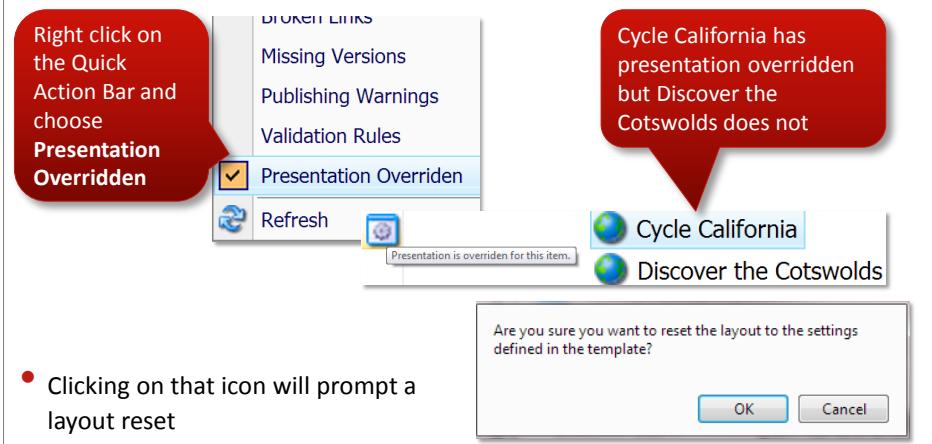
- The **RAW value** of that field is **custom XML**

```
<r xmlns:xsd="http://www.w3.org/2001/XMLSchema" xsd="http://www.w3.org/2001/XMLSchema">
  <d id="{FE5D7FDF-89C0-4D99-9AA3-B5FBD009C9F3}" l="{63DE1F09-91D3-4495-B25A-6F0C4652BDA0}">
    <r id="18303E84-8352-4348-BC98-C9CF82F05C9F" ph="Main" uid="{BF15750A-E70E-4963-96DA-299D29178564}" />
      <r id="564D7ABA-D3AA-495B-B7CB-1ACF7E99F817" ph="Main" uid="{0DE1B61D-A4BB-40E0-ADE8-0149ADACDB7E}" />
        <r uid="660225EC-F59E-4EA4-8CFA-E882C5D96F86" id="48CB254F-7E8B-4009-9BFC-982F7B40E757" ph="PageContainer" />
          <r id="27A14165-42AF-4147-8BB6-461FD05E5281" ph="Placeholder" />
```

From an author perspective

How can you see if presentation is overridden at item level?

- Use the Quick Action Bar



- Clicking on that icon will prompt a layout reset



Walkthrough – Review raw values of renderings field

1. In **Sitecore Rocks** double-click on a **holiday** item that has no presentation overridden
2. Right-click in the **editor** pane and select **Raw Values** and **Standard Fields**
3. Look at for the **Layout** field section with the **Renderings** field – note the XML
4. Because no presentation details have been specified on the item itself, its presentation details are determined by the standard values of its data template. As such, the contents of the **Rendering** field displays the presentation details inherited from the standard values:

```
<r>
<d id="{FE5D7FDF-89C0-4D99-9AA3-B5FBD009C9F3}" l="{CBA12FE1-ADA9-4BAD-9BCD-DEA43CB16DD7}">
  <r id="{7A2FDD0A-D210-4D16-895B-ADBD83284FAF}" ph="main" uid="{2FFBCA73-B497-4946-8E55-9618107BA0DE}" />
  <r id="{D8437E8A-7420-4041-8077-3FEA9C384169}" ph="main" uid="{62F4C382-34A2-4C4B-8F53-896CCFE9226D}" />
</d>
<d id="{46D2F427-4CE5-4E1F-BA10-EF3636F43534}" />
<d id="{73966209-F1B6-43CA-853A-F1DB1C9A654B}" />
</r>
```

5. Add another **Holiday Overview** component to that holiday item (> right-click the **component**, **Tasks>Design Layout** and **save**)
6. Right-click on the **editor** pane of the **holiday** item and click **reload** in the context menu
7. Look at the **Rendering** field again. Now you only see what you have changed. This is called a *Layout Delta*. In this example, a new rendering is added to the **main** placeholder in a particular device

```
<r xmlns:p="p" xmlns:s="s" p:p="1">
  <d id="{FE5D7FDF-89C0-4D99-9AA3-B5FBD009C9F3}">
    <r uid="{E443101D-E33D-4DC4-AE80-F2CF65F66DDD}" s:id="{D8437E8A-7420-4041-8077-3FEA9C384169}" s:ph="main" />
  </d>
</r>
```

8. If you switch to the Content Editor and right-click in the **Quick Action** bar, select **presentation overridden** it shows you which items have Layout Deltas
9. You can click on the icon to **reset** the presentation back to standard value

Processing an item's presentation details

- When an item is rendered, the **LayoutResolver** pipeline first looks at the **standard values** of an item's data template

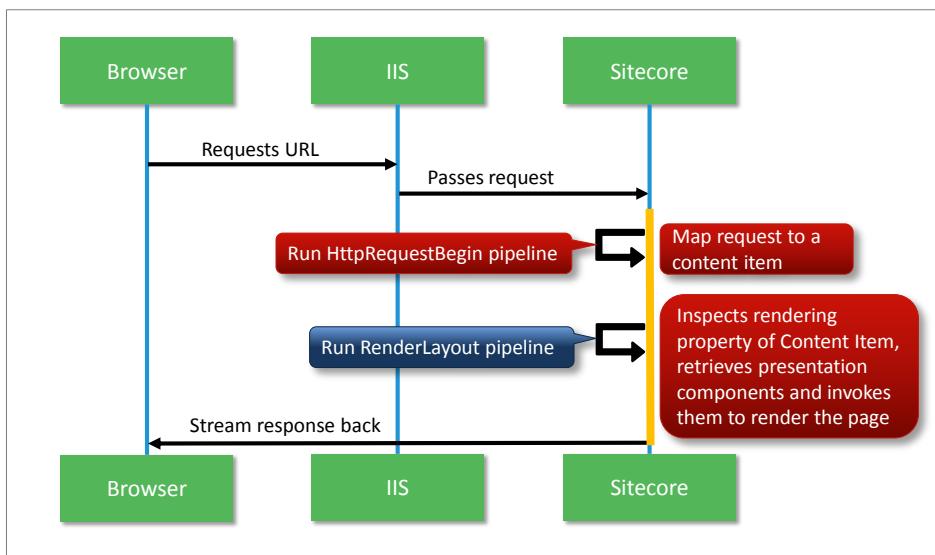


- Next if the **item has presentation** details applied to it, the **LayoutResolver** will **incorporate** these **changes** along with the standard values



- The end result is an **item** that **takes** its **presentation** details **from** the **standard values** of its template **as well as any item-specific** changes
- **Note:** that this includes deletions – if you delete a component on an item that it is inheriting from standard values, the layout delta will store those instructions

The RenderLayout in the Request



Review

Layout deltas

Q In what field are presentation details stored?

✓ Renderings field

Q In what format are presentation details stored?

✓ Custom XML

Q How does the LayoutResolver pipeline work?

✓ It uses the standard values presentation details first and then incorporates any item specific presentation details

Module 6

Real world solutions

Contents:

- Introduction to Sitecore Cycling Holidays
- Familiar concepts
- Dealing with larger sites

Topic 6.1 Introduction to Sitecore Cycling Holidays

Introduction

Objectives

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

- Use the Page Editor to design and edit pages in the context of Traincore

Content

Sitecore Cycling Holidays

- A fictional **cycling holiday site** where users can search for holidays and get tips and advice about cycling
- SCH is built according to **front-end** and **back-end best practices**
- It tackles common **architectural problems** faced by **multi-language** and **multi-site** implementations
- Built to support **Page Editor** and the **personalization and analytics** aspects of the CEP



Demo: Change the appearance of a holiday page

As an author, you are empowered to change the appearance of a page without developer intervention:

1. Navigate to the **Battle of the Hills** holiday page and switch to **Page Editor** mode
Move the **Gallery Container** component beneath the **Page Container** component



Figure 6-1 Moving a component

2. Add a new widget to the widget container at the bottom of the page
3. Change the widget to **third width** and **hide** the image



Figure 6-2 Inserting a widget

Review

Introduction to Sitecore Cycling Holidays

Q Identify some components on the easyJet homepage

- ✓ Banner, booking form, promotional widgets, latest travel deals

Q Which components would qualify for static inclusion with <sc:Sublayout />?

- ✓ Navigation, footer

Q What might some of the 'structural' components be (e.g. columns, containers)?

- ✓ Two column, one column



Topic 6.2 Familiar concepts

Introduction

Objectives

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

- Describe the topology and data structure of Sitecore Cycling Holiday site
- Understand the difference between global content items and items representing pages
- Describe the levels of presentation detail nesting used
- Create and configure a new device definition item

Content



Walkthrough – Solution setup

1. In the file system, look at the **Traincore** project folder. As we saw in Module 2, the project folder sits outside the Website root
 - a) The core of the application lives in the **Training** folder
 - b) There are two additional projects that are specific to this implementation – **Training.Utilities** (abstracted functionality) and **Training.Controls** (any Sitecore web controls)
2. The **Sitecore.Utilities** project contains abstracted functionality that can be used across projects. Think back to Module 1 and the suggestion that you have a base implementation
3. All reference .dlls are stored in a **Libraries** folder If you reference them straight from the website, you are relying on the .dll always being in the same Website folder, and you have no control over what version is being referenced on different developer machines
4. Note the number of files in **/App_Config/Include**, pay particular attention to the **sites** and cache-clearing configuration files
5. Open the **sites** configuration file, **note** that the content is patched in
6. In a browser, navigate to the <http://traincore/sitecore/admin>ShowConfig.aspx> page
7. Search for the **<sites>** node and note that all the changes specified in the **Include** folder are applied

Solution setup

Project structure

- Abstract functionality in separate projects – base implementation in **Sitecore.Utilities**
- Outside of the web root
- Everything in **subfolders**

Configuration

- Patching in configuration from **/App_Config/Include**

Libraries

- Referenced .dlls in a **/Libraries/** folder

| Name | Last modified | Type |
|--------------------|------------------|-------------|
| Libraries | 19/05/2013 20:58 | File folder |
| Sitecore.Utilities | 12/04/2013 09:02 | File folder |
| Training | 03/06/2013 11:23 | File folder |
| Training.Advanced | 12/04/2013 09:02 | File folder |
| Training.Controls | 03/06/2013 11:23 | File folder |
| Training.Utilities | 12/04/2013 09:02 | File folder |

| Name |
|--|
| layouts |
| BaseCore |
| columns |
| containers |
| content |
| print |
| search |
| site |
| widgets |
| basecore-book-holiday.ascx |
| basecore-book-holiday.ascx.cs |
| basecore-book-holiday.ascx.designer.cs |
| basecore-news-listing.ascx |
| basecore-news-listing.ascx.cs |
| basecore-news-listing.ascx.designer.cs |

| Name |
|-----------------------------------|
| BaseCore.CascadeDatasource.config |
| BaseCore.HtmlCacheClearer.config |
| BaseCore.LayoutInheritance.config |
| BaseCore.Sites.config |

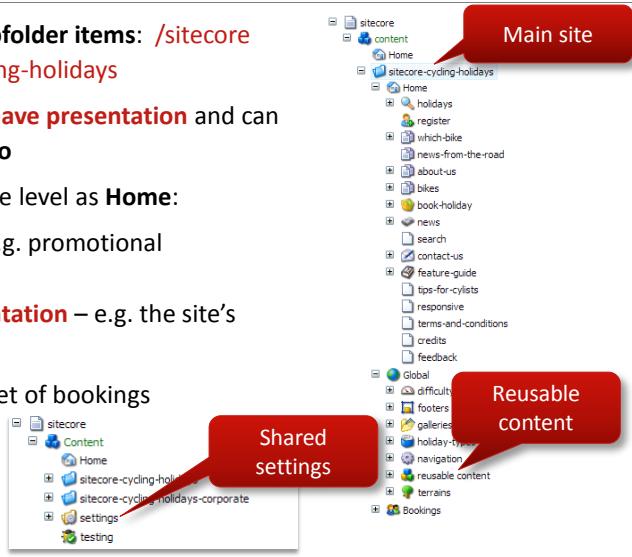


Demo – Site topology

1. Log into the **Sitecore Desktop** as **admin** (password: **b**)
2. In the Sitecore Desktop, open the **Content Editor** and expand the **/sitcore/content** item
3. Note the presence of two folder items – **sitcore-cycling-holidays** and **sitcore-cycling-holidays-corporate**, we will cover how a multi-site implementation is set up in a later topic
4. Expand each folder item – both have **Home**, **Global**, and **Bookings** child items. The children of the **Home** item represent the site structure; the children of the **Global** item are small units of reusable content without presentation that can be used across the site. When we picked a data source for our widget, we picked from the children of the **Reusable content** item
5. Note the **Settings** item contains content that is identical across sites (like **CSS Classes**)
6. Change one of the **CSS classes**, change the **language**; the value is the same across all languages because the field is shared
7. Look at the **Bookings** item, it is a searchable bucket with high volumes of data, this interface is covered in more detail in Module 9

Site topology

- **Sites defined under subfolder items:** `/sitecore /content /sitcore-cycling-holidays`
- **All items under Home have presentation** and can therefore be **browsed to**
- **Global item on the same level as Home:**
 - Reusable content** – e.g. promotional widgets, galleries
 - Items without presentation** – e.g. the site's **navigation structure**
- **Bookings item** – a bucket of bookings
- **Settings item** – outside the site folder, settings that apply to all



Demo – Templates and Inheritance

1. In the Sitecore Desktop, open the **Template Manager**. All templates live in subfolders; the majority are in **/BaseCore**
 2. Click on **/BaseCore/Pages/Holiday**. It is inheriting from three base templates. Excluding the **Standard Template**; the base templates are stored under **/BaseCore/Base**. Only pages that require an image and content need Base Page Content. The homepage, for example, might not need it. Splitting base templates up allows you to pick and choose what to include, and prevents field redundancy
- All templates have identifying icons, as do some field sections

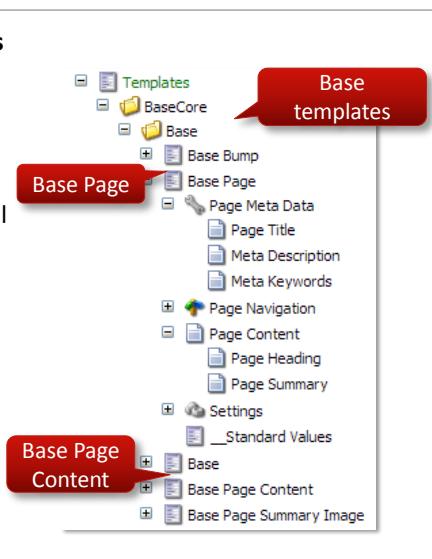


Tip

SCH defines main and footer navigation in a separate folder under /Global rather than generating it from site structure. This is a much cheaper 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

Templates and inheritance

- Data templates live under **subfolder items**
- Recognizable names for authors
- Use of **icons** – even for individual fields or field sections
- Uses a number of **base templates** – e.g. all items with presentation inherit from **Base Page**
- Prevent **field redundancy** by using multiple base templates – e.g. **Base Page Content**
- **Shared fields** used for **settings items**



Best 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



Draw It

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

- Canvas component – could swap out an entire page in the future
- Head and Body components (with placeholder) – could put in an entirely different *Head* section for a particular site
- Containers (with placeholders)
- Columns – one column, two column, three column Functional components



Demo – Presentation configuration

1. Using Visual Studio, open `/layouts/BaseCore/site/bascore-canvas.ascx`
2. Note statically bound `basecore-head.ascx` and `basecore-body.ascx`
3. Note the **main** placeholder in `basecore-body.ascx` – everything around this placeholder is scaffolding
4. Open `/layouts/BaseCore/containers`, this represent slices of a page; all other widgets are nested in their placeholders



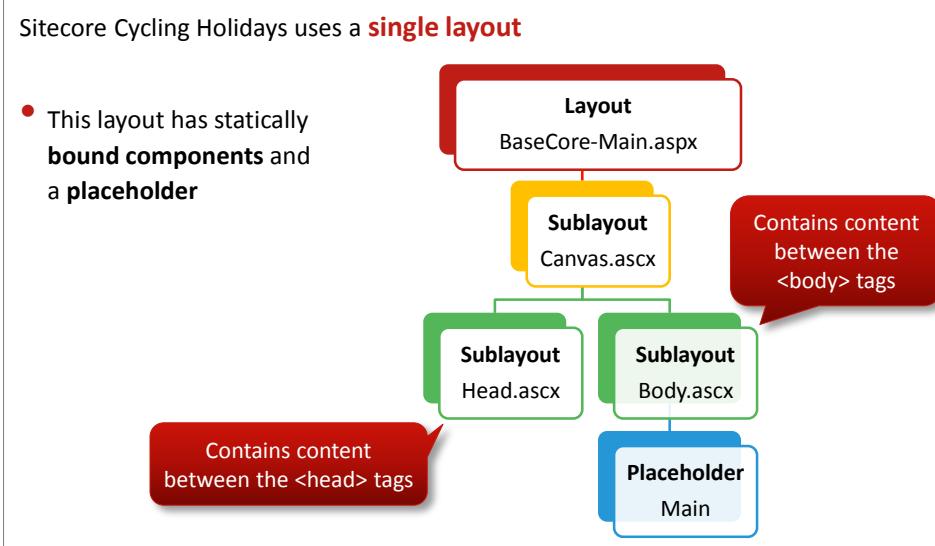
Important

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

5. Open `/layouts/BaseCore/columns`, this third level of nesting, has very little functionality on its own
6. Open `/layouts/BaseCore/content` and `/layouts/BaseCore/widgets`. These components actually perform a function (e.g., outputting content or listing children)
7. Open the **General Widget** sublayout, note that it accepts a data source and uses parameters

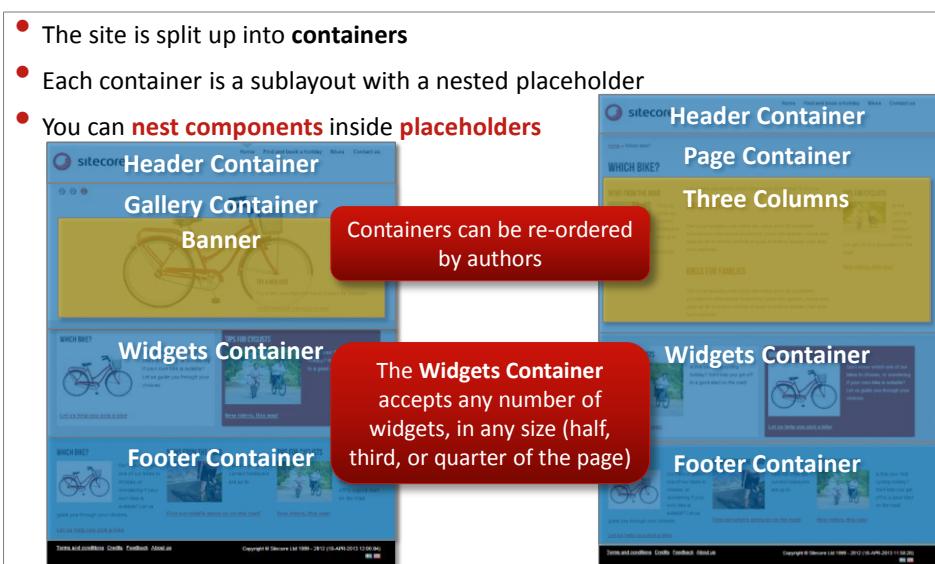
8. Open the **Content Editor**
9. Select the **Home** item under `/sitecore-cycling-holidays/` and click the **Presentation > Details** option
10. Note that presentation detail definition items have been assigned icons to make it easier to differentiate them from one another
11. Open a **General Widget** and note the datasource and parameter selections. The **General Widget** is the most reusable sublayout in the site

Presentation configuration



Placeholder nesting

- The site is split up into **containers**
- Each container is a sublayout with a nested placeholder
- You can **nest components** inside **placeholders**



Flexible, reusable components

- In the context of SCH, a **widget** is a reusable component that can be used in a number of placeholders – the HTML is written in such a way that it will adapt automatically
- Style, heading type, and width** determined by parameters
- Accepts a **data source** from a global store of items that do not represent pages



Global, reusable content

- Outside the main body of the site under **Global**
- Not associated with any presentation – therefore **cannot be browsed to**
- Used as data source for **General Widget** component



Demo – Create a mobile site

- Using Sitecore Rocks, create a new device called **Mobile**
- View the presentation details of any item and note that **Mobile** now appears as a device option in the Device Manager
- Using Visual Studio, create a new layout called **Mobile**. Note that you are prompted to create a layout definition item in Sitecore
- Use Sitecore controls to output the **Heading** and **Main Content** fields
- Add a placeholder control to the bottom of the page and name its key **widgets**
- Save and deploy** the solution
- Assign the new layout to the **Home** item on the **Mobile** device. Add a number of widgets to the **widgets** placeholder. You do not have to use a mobile-specific set of sublayouts, just style them differently
- Open the item in Preview mode. Use the **Experience** tab to change the device between **Default** and **Mobile**. Note the **sc_device** in the query string

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 6.2 – 60 min

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

- Sitecore Cycling Holidays are 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**

The screenshot shows a completed news article page. At the top, there's a navigation bar with links for Home, Find and book a holiday, Bikes, and Contact us. Below that is a header with the Sitecore logo and a sub-header 'Hello from us'. The main content area contains a paragraph of placeholder text. To the right, there's a sidebar titled 'RELATED ARTICLES' with two items: 'New bikes released' and 'Review of bikes'. At the bottom, there's a footer with a copyright notice and a link to Sitecore 2013.

Figure 6-3 The completed page



Create a news article data template and standard values

In the following labs you will:

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

Lab A. Create a News Article data template

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



Knowledge Check

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

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



Tip

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

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

| Field Name | Field Type |
|------------------|------------|
| Article | |
| Related Articles | Multilist |

Figure 6-4 News Article field in Rocks



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** item
5. Set the template icon to the **News Add** icon (search for or type **Network/16x16/news_add.png**)



Knowledge Check

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

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

1. Create the **News Article** template **standard values**
2. On the template's standard values, add a token to the **Page Title**, **Page Heading**, and **Navigation Title** fields that will result in the item's name inserted into those fields on creation
3. Navigate to the **News Listing template's standard values** and set **News Article** as an **insert option**

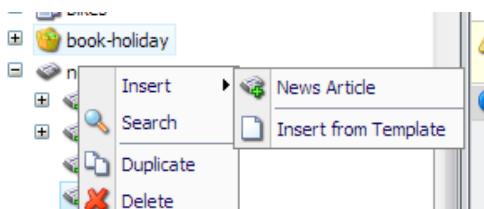


Figure 6-4 News Article as an insert option

4. Create several new items based on the **News Article** template under /sitecore/content/sitecore-cycling-holidays/home/news
5. Populate each item with sample content and then **save** your work. As you create news articles, select two or more **articles** in the **Related Articles multilist**
6. The message in **Figure 6-5 An item in workflow** 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 (e.g. to make sure that no news articles that have not been approved by the news team appear on the site)

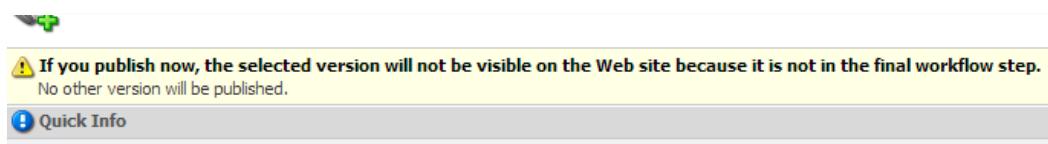


Figure 6-5 An item in workflow

7. To approve the news article, click the **Home** tab, then click the arrow next to the **Edit** button > Click the **Submit** option as detailed below

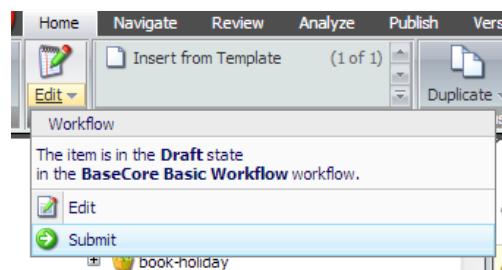


Figure 6-6 Submitting an item through a workflow

- We will cover workflow in more detail in Module 10
- 8. **Smart publish** the database, then **browse** to the **news listing** 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 do you **not** see the ‘The layout for the requested document was not found’?



Create Component and Bind to 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 C. Create a Related Articles Sublayout

1. Using **Visual Studio**, right-click on the **/layouts/BaseCore/content/** folder and select the **Add > New Item...** option, insert a new sublayout called **Related Articles**
2. Use the Sitecore section to **insert** a Sublayout, name your sublayout **Related Articles**

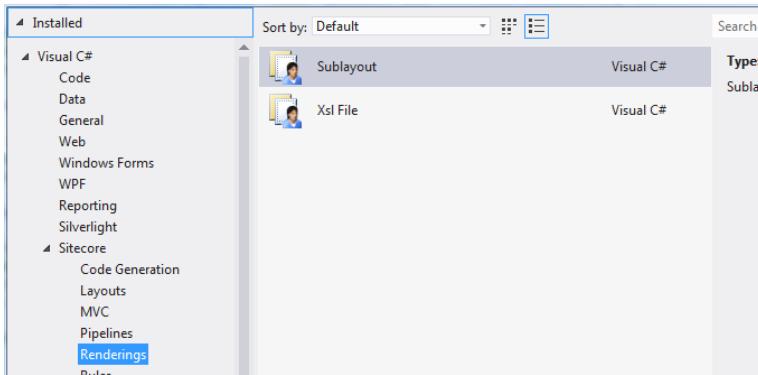


Figure 6-7 Add a new sublayout

3. When prompted by Sitecore Rocks, add the corresponding **sublayout definition** item to **/sitecore/content/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 on the project, select the **Sitecore>Connect...** option and pick the Sitecore instance you want to use

4. A sample repeater has been provided in the **student resources folder** (*WND Labs > Module 6 > Topic 6.2 > Lab B > Code > Related Articles.aspx*) – **Copy** and **paste** this code into your **.aspx** and **save** Do not overwrite the entire file, the sample only contains the code for the repeater
5. Navigate to the **News Article** template’s **standard values** and open the **Design Layout** pane
6. **Copy** presentation details from standard values when prompted

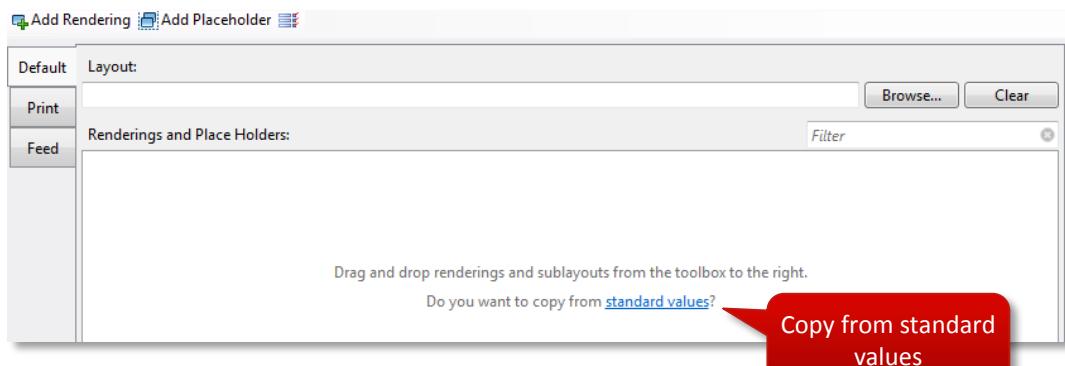


Figure 6-8 Copying from Standard Values

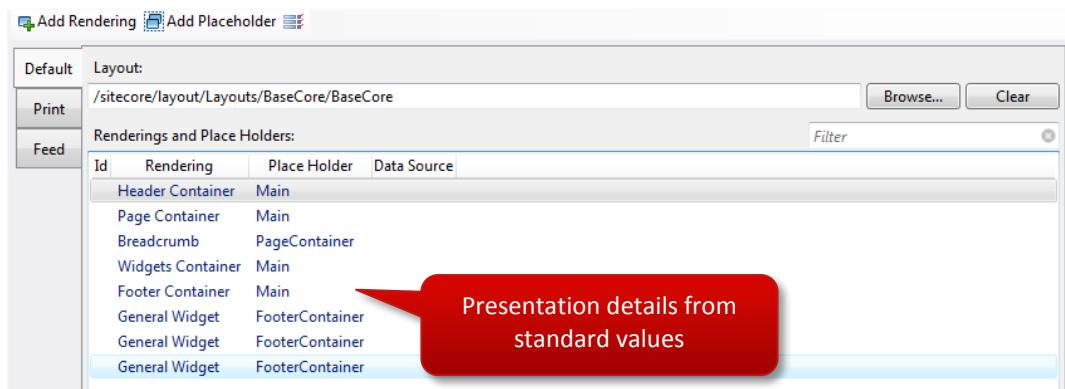


Figure 6-9 Presentation Details

7. Add the following components: an **existing component**, and the **one you created earlier**. Add them in the order specified – the **TwoRightColumn** placeholder is specified on the **Two Column Content** sublayout

| Rendering name | Placeholder |
|--------------------|----------------|
| Two Column Content | PageContainer |
| Related Articles | TwoRightColumn |

8. **Save and deploy** the solution



Knowledge Check

When you use a Sitecore control and specify a field – e.g.

`<sc:Text field="Page Heading" runat="server" />`, what item does the control default to as its data source?

Lab D. 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 would you not 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>
```

- Because the control is in a repeater, we want to tell it to retrieve the **Page Heading** from each repeater item, not 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 %>"
```

6. 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 which method to use)



Code sample

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

9. **Save and deploy** the solution
10. **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** multilist

[Home](#) » [News](#) » Hello from us

HELLO FROM US

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. [Donec non enim](#) in turpis pulvinar facilisis. Ut felis.

HEADER LEVEL 2

1. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

RELATED ARTICLES

- [New bikes released](#)
- [Review of bikes](#)

6-10 Finished news article page

Review

Familiar concepts

- | | |
|--|--|
| <p>Q What might you do with methods that could be used across multiple Sitecore projects?</p> <p>✓ Abstract code to a generic utilities project</p> <p>Q Why can't you preview items under Global?</p> <p>✓ They do not have presentation details</p> <p>Q Why is the Settings item outside any individual site folder structure?</p> <p>✓ Values shared across all sites</p> | <p>Q What kind of content might you display using a General Widget?</p> <p>✓ Promotions, announcements – anything that could be re-used across the site</p> <p>Q Name 2 devices you may want to target with different presentation details</p> <p>✓ Tablets, mobiles</p> |
|--|--|

Topic 6.3 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

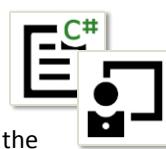
SCH supports multiple languages

- SCH has been configured to use English (US), Swedish and French

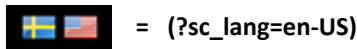


Things to consider when building a multi-language site:

- Components must be able to support **different content lengths**
- **All content must be translated** – including small text like ‘back to top’ or button text (consider **domain dictionaries**)
- Make fields **shared** if their value should be **shared across all languages** (e.g. in SCH, settings are the same regardless of the language)
- Build feature (list of links) that allows users to **change the site language**:



- Build feature (list of links) that allows users to **change the site language**:



Language settings

For all sites

```
<setting name="DefaultLanguage" value="en-US" />
```

Per individual site

```
language="en-US" property on <site>
```

Sitecore client language

- Change language of **Sitecore interfaces** by setting:

```
<setting name="ClientLanguage" value="en" />
```

Restrict access to languages per user or role

Everything is an item – apply **security restrictions** as you would with any item

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)

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

Custom code must take languages into account

- e.g. Spider creating a Google sitemap

Language fallback

- Consider how any **fallback solution** will affect performance

Translation vs Localization

- Are you writing content for a **locale**, or translating **global content**?

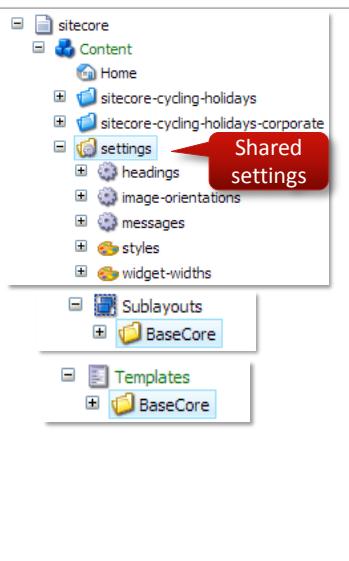


Demo – Multi-site

- Using the Sitecore Desktop, open the **Content Editor** and navigate to **/sitecore/Content/**
- Re-visit the two sites – **sitecore-cycling-holidays** and **sitecore-cycling-holidays-corporate**. Each has a **Home** item and each can be previewed
- Preview both **Home** items in turn – same layout and capabilities, different content and presentation details applied on the items
 - 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 (e.g., a mobile-specific site, taking advantage of devices)
- If you wanted to copy content between sites (e.g., terms and conditions) you could **clone** items (as an Author) by clicking the arrow next to **Duplicate > Clone** and selecting a location
- While the item is a clone, it is ghosted, any changes to the original will update the clone, unless a field has been explicitly overridden
- Each site has a node in the web.config, in the Traincore example, they have been patched in using **include files**
- Show **/sitecore/admin>ShowConfig.config** for final, transformed configuration
- Note that the site **host name**, **language**, **start path** and **start item** differs between sites
- Note that that sites share an **application pool** – **license implications** if they do not, but **performance implications** if they do
- Multi-site implementations should not be abused In many cases they are not appropriate

Multi-site content tree

- Multiple **site root items** in content tree
- **Shared content** outside individual site structure – e.g. settings folder
- Templates and presentation organized into **subfolders** – in file system and in Sitecore tree, based on which site they belong
- In single-site installations, **mobile or print** presentation would live in a subfolder



Multi-site configuration

Company's main site and a smaller corporate site

- Each site has its own <site /> node in the web.config with a unique **start path** and **host name**
- The sites share **templates** and **presentation** – code base is identical, content varies

```

<sitecore>
  <sites>
    <site patch:after="[@name='modules_website']" name="sitecore-cycling-holidays-corporate"
      hostName="traincore-corporate" virtualFolder="/" physicalFolder="/" rootPath="/sitecore/content/sitecore-
      cycling-holidays-corporate/" startItem="/home" language="en-US" database="web" domain="extranet"
      allowDebug="true" cacheHtml="true" htmlCacheSize="10MB" registryCacheSize="0" viewStateCacheSize="0"
      xslCacheSize="5MB" filteredItemsCacheSize="2MB" enablePreview="true" enableWebEdit="true"
      enableDebugger="true" disableClientData="false" dictionaryDomain="{8F2E665E-1048-4D86-A8D7-D7A831DA3CDC}" />
    <site patch:before="[@name='modules_website']" name="sitecore-cycling-holidays" hostName="traincore"
      virtualFolder="/" physicalFolder="/" rootPath="/sitecore/content/sitecore-cycling-holidays/" startItem="/
      home" language="en-US" database="web" domain="extranet" allowDebug="true" cacheHtml="true"
      htmlCacheSize="10MB" registryCacheSize="0" viewStateCacheSize="0" xslCacheSize="5MB"
      filteredItemsCacheSize="2MB" enablePreview="true" enableWebEdit="true" enableDebugger="true"
      disableClientData="false" dictionaryDomain="{8F2E665E-1048-4D86-A8D7-D7A831DA3CDC}" />
  </sites>

```

Warning!

Multi-site quickly becomes complex

- URL resolving/generation affected
- Single set of presentation details and templates vs site-specific requirements
- Shared content means complex security considerations
- Affects ability to customize pages
- Page Editor initializes in context of the site user logged into
- Source fields need to contain a query
- Sites share the same application pool (licensing implications)





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 has licensing implications



Best 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.
- SCH 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

Review

Dealing with larger sites

| | |
|---|--|
| <p>Q What might you do with methods that can be re-used across projects?</p> <p>✓ Abstract code to a generic utilities project</p> | <p>Q What kind of content might you display using a General Widget?</p> <p>✓ Promotions, announcements – anything that could be re-used across the site</p> |
| <p>Q Why can't you preview items under Global?</p> <p>✓ They do not have presentation details</p> | <p>Q Name 2 devices you may want to target with different presentation details</p> <p>✓ Tablets, mobiles</p> |
| <p>Q Why is the Settings item outside any individual site folder structure?</p> <p>✓ Values shared across all sites</p> | <p>Q Why should you approach multi-site implementations with caution?</p> <p>✓ Single application pool, code base quickly becomes messy</p> |

Extend

Dictionary Domains

- <http://www.sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog/Posts/2012/11/Sitecore-ASPNET-CMS-6-6-Features-Dictionary-Domains.aspx>

Re-using and Sharing Data

- http://sdn.sitecore.net/upload/sitecore7/70/reusing_and_sharing_data_sc70-a4.pdf

Topic 6.4 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

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):

| | |
|--|--|
| Absolute query | |
| query:/sitecore/content/Home//* | |
| Relative query | |
| query:./ [@@templatename='Bike'] | |
| Filter by template | |
| query:/sitecore/content/Home//*[@templatename='Bike'] | |
| Filter by content | |
| query:/sitecore/content/home/*[@#Page Title# = 'News'] | |
| Return at index | |
| query:/sitecore/content/home/*[3] | |

Sitecore query

When you use Sitecore query, you are interrogating an XML representation of the Sitecore tree

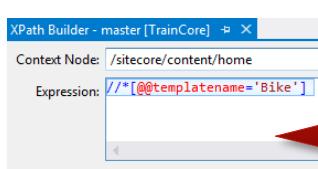
Sitecore query is very similar to XPath

Tools

Sitecore Rocks – XPath Builder

Use the prefix “query:”
(not required in Xpath Builder)

```
<item name="Home" key="home" id="11109550F-DEA5-412A-B11C-5A1DF7E70E7F" tid="76036F5E-CBCE-46D4-  
version parentid="10E91A24-41A8-4D01-9E80-07443B7C4501">  
+<version language="en" version="1">  
+<fields>  
+<field tuid="10E91A24-41A8-4D01-9E80-07443B7C4501" key="__created_by" type="text">  
+<content>sitecore\admin</content>  
+</field>  
+<field tuid="C8F93A4F-BFD4-4E8F-9C61-152558584661" key="__valid_from" type="datetime">  
+<content></content>  
+</field>  
+<field tuid="BADD0FC9-53E0-4D0C-BCC0-2D784C282F6A" key="__updated_by" type="text">  
+<content>sitecore\admin</content>  
+</field>  
+<field tuid="18DC037E-A112-427B-BBD4-4143751E123F" key="__revision" type="text">  
+<content>1b0dcded-b1c8-4bd1-9353-d318e50989eb</content>  
+</field>  
+<field tuid="4C346442-X559-4EFD-8982-44AEDF467D21" key="__valid_to" type="datetime">  
+<content></content>  
+</field>  
+<field tuid="178977384-3C97-45DA-A847-81B00500E250" key="title" type="text">  
+<content>Sitecore</content>  
+</field>  
+<field tuid="A60ACD61-A62B-4182-8329-C9579922CE74" key="text" type="Rich Text">  
+<content>p>Welcome to Sitecore</content>  
+</field>  
+<field tuid="25BED78C-4957-4165-998A-CAB52F67497" key="__created" type="datetime">  
+<content>20080407T135900</content>
```



In Rocks, right click on
any item in the tree
Tools > Xpath Builder

Query syntax

| | |
|---|---|
| Specific path | <code>query:/sitecore/content/home/</code> |
| Gets all immediate items under <i>home</i> | |
| Note: Use if there is only one location. e.g., /settings and when settings is shared by all sites | |
| Cannot be used in a multi-site | |
| * wildcard | <code>query:/sitecore/content/*/news</code> |
| Gets items under content/[anything] called news | |
| / children | <code>query:/sitecore/content/home/*</code> |
| Gets all children of <i>home</i> | |
| / followed by GUID, item name, or * | |
| // descendants | <code>query:/sitecore/content/home//*[@@templatename='News']</code> |
| Gets all items that have <i>home</i> as ancestor and have <i>News</i> as the template name | |
| // followed by GUID, item name, or * | |
| Note: Can be expensive if the tree is large | |
| @@ XML attribute | <code>query:/sitecore/content/home[@@templatename='News']</code> |
| Gets all items under <i>/home</i> that use the <i>News</i> template | |
| e.g. @@templatename or @@templateid | |
| Note: Case sensitive | |
| @ field | <code>query:/sitecore/content/home/*[@#Page Title# = 'News']</code> |
| Gets any items under <i>/home</i> that have a field called <i>Page Title</i> and field value of <i>News</i> | |
| Note: Use # to escape spaces and hyphens in field names | |
| . context item | |
| In the case of a source field, it's the item you are currently viewing in the content tree | |
| [1] index | <code>query:/sitecore/content/home/*[3]</code> |
| Returns the item under <i>home</i> at index 3 | |
| Note: It's not a zero-based index | |



Demo – Axes

1. Open the **XPath builder** and evaluate the following query:

| | |
|----------------------------------|--|
| ancestor-or-self::* | |
| | <code>query:/sitecore/content/Home/ancestor-or-self::*</code> |
| ancestor-or-self::[@@ ..] | |
| | <code>query:/sitecore/content/Home/ancestor-or-self::*[@@templatename='Main section']</code> |

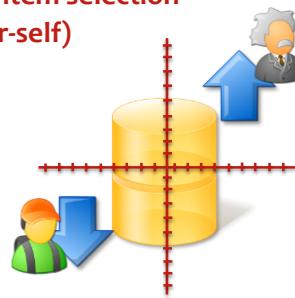
2. Using Sitecore Rocks, navigate to the **Holiday** Page template and look at the **Terrain** field, **note** the presence of an ancestor-or-self query that returns all values relevant to the context site

Axes

**Axes allow you to determine the direction of item selection
(e.g., parent / ancestor-or-self / descendant-or-self)**

Query without axes

- `query:/sitecore/content/home/*`
returns all items under home
- `query:/sitecore/content/home/parent::*`
returns the parent of home – i.e., ‘content’



Query with axes

- `query:./ancestor-or-self::*[@@templatename='News Article']`
returns all ancestors or self of the context item with the template *News Article*
- For more examples of axes, see documentation:
<http://sdn.sitecore.net/Reference/Using%20Sitecore%20Query/Sitecore%20Query%20Syntax/Axes.aspx>

Uses of Sitecore query

Source field

- Any field that accepts a source – filter by name, template content, location

In the API (sparingly!)

e.g. `Sitecore.Context.Database.SelectItems("query:/sitecore/content/*");`
e.g. `Sitecore.Context.Item.Axes.GetDescendants();`

For any complex querying, use Sitecore search! (Module 9)

- Queries an index
- LINQ API
- Sitecore query more likely to lead to performance issues

Limitations and warnings

- Queries are useful in multi-site environment when you need to get items without using a path or GUID
- Queries can be **expensive** – recommend against using in high traffic environment
- **Don't query the entire content tree** 
- Note where you have used complex queries - may affect **performance** 
- 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.MaxItems
    Specifies the max number of items in a query result set.
    If the number is 0, all items are returned. This may affect system performance, if a
    large query result is returned.
    This also controls the number of items in Lookup, Multilist and Valuelookup fields.
    Default value: 100
-->
<setting name="Query.MaxItems" value="100" />
```

Apply – Topic 6.4 – 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 displays all items based on the **News Article** template that exist under the same news listing as the item you are editing. (Hint: you are looking for any item whose ancestor-or-self:: has a type of **News Listing**)

Review

Sitecore query

- | | |
|---|---|
| <p>Q What will . return?</p> <p>✓ The context item – i.e. the one you are currently viewing</p> <p>Q Name two axes available in Sitecore query</p> <p>✓ ancestor-or-self:::, parent::</p> | <p>Q Why should you not query the entire content tree?</p> <p>✓ Expensive – particularly if you are iterating through all descendants</p> <p>Q Unless you are doing a very simple query on a very limited area of your Sitecore tree, what should you use instead?</p> <p>✓ Sitecore search</p> |
|---|---|

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/databdefinitioncookbook-a4.pdf> (Search: “Treelist Parameters”)

Module 7

Configuring the Page Editor

Contents:

- Building for the Page Editor
- Datasource restrictions
- Parameters and parameter templates
- Placeholders
- Compatible renderings and allowed controls
- Advanced Page Editor configuration

Topic 7.1 Building for Page Editor

Introduction

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

- State the three best practices for designing HTML so that it works well with the Page Editor
- Describe how to approach layout and data
- Define why HTML should be made modular
- Indicate on a page which parts should be statically bound
- Discuss the effect of widgets changing size due to author input
- State two reasons why it's better to design for the Page editor from the start even if not initially required

Content



Demo: Using the Page Editor with Sitecore Cycling Holidays

- Log in to the **Page Editor** as **admin** (password: **b**) and navigate to the **home** page
- Enable the **Designing** mode
- In the **Home** tab, click the **New Component** option

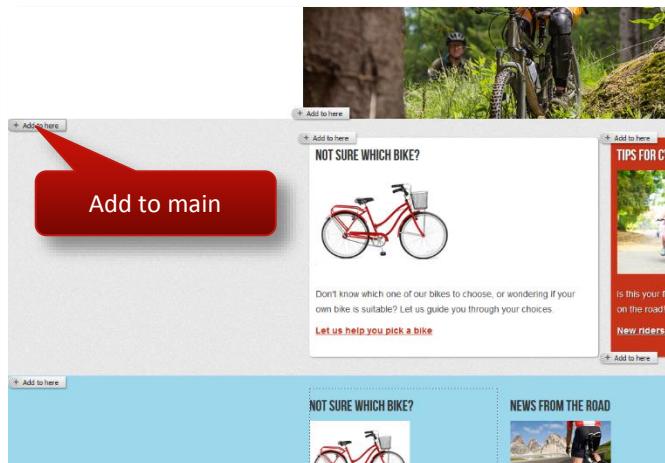


Figure 7-1 Add new component to main placeholder

- Add a new component to the **main** placeholder
- Select the **Page Container** sublayout

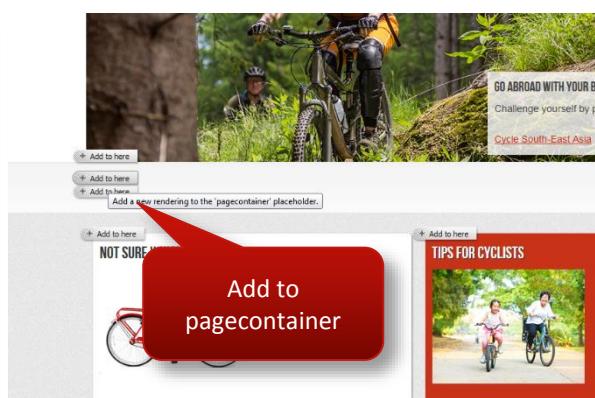


Figure 7-2 Add new component to pagecontainer placeholder

- Add a new component to the placeholder inside the **Page Container**
- Select the **Two Column** sublayout
- Select one of the **widgets** in the **Widget Container** and move the component to the right-hand column from the Two Column sublayout

9. Click the **Edit component properties** option, the width will adjust to fit the container
10. Switch to the **Editing** mode and add some content beneath the **Home** heading
11. Remove the entire **Gallery Container** sublayout from the top of the page
12. **Save and preview** the homepage

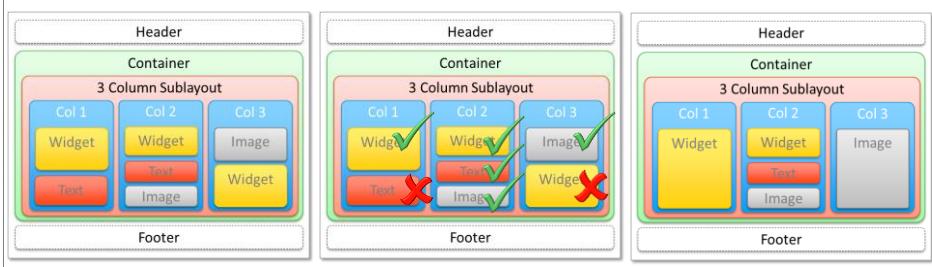
Designing for the Page Editor - components

The Page Editor enables authors to assemble pages from components

- Isolate page **components** and work out where there is **re-use of functionality**
*Note that this does **not** mean re-use of style*

Decide to what extent the layout needs to be dynamic

- Where can each discreet component be placed on a page?
- Does the hierarchy of the page need to be flexible?



Tip

Think about where it should be possible to place each discrete component on a page. Does the hierarchy of the page need to be flexible (e.g., should you be able to put widgets before and after the main site content?)

Designing for the Page Editor

Separate layout from data

- Name data templates on the data they hold – not their appearance
- For 2 or 3 column layouts create a Standard Content data template and allow users to choose between a 2 or 3 column container **sublayout**

Only bind statically when:

- **Components** should **not move** e.g. site header or footer etc.



Make content editable by:

- Always rendering content with the **FieldRenderer** or Sitecore controls



Tip

- For example, do not create Two Column Standard Content and Three Column Standard Content data templates. Create a single Standard Content data template and allow the author to choose a two or three column container sublayout
- Make sure you only bind components statically if they are very unlikely to move (e.g., the header or footer of a site). You should use dynamic binding, through placeholders, whenever possible to ensure the authors have greater flexibility
- Always render content using the FieldRenderer (or Sitecore controls). This will make content editable in Page Editor

HTML for the Page Editor

- **HTML** needs to be **modular** because a widget or navigation tree might need to work in **several locations**, and not lose style

Note Sure Which bike?
Are you not sure which one of our bikes to choose, let us guide you through your choices!

News from the road
Find out what's going on on the road!

Tips for cyclists
Is this your first cycling holiday? We'll help you get off to a good start on the road!

- Authors can put in more or less content than expected so if one widget has 400 characters and another has 20, the 400 character widget is going to extend much further down the page



Tip

In order for a component to be selectable in the Page Editor (e.g., a widget), it must always contain some form of content (e.g., empty HTML or dummy text). If your component's data source is null, you must output something

One solution is to check if you are viewing a component in the Page Editor

(`Sitecore.Context.PageMode.IsPageEditor`) and output a standard message (i.e., "This component needs content") if there is no content. This allows you to hide empty components on the website's front end so that it does not look broken to authors

Is Page Editor for you?

- If your site has many one-off, custom pages, the designing aspect of Page Editor may not be relevant, **componentize** as much as possible, but be aware of the limitations of the design
- Even if the client has no immediate plans to utilize the **Page Editor** interface, **build for it anyway** due to 3 reasons:
 1. You may want to start using Page Editor in future and find it **difficult to transition** smoothly if you have not taken that requirement into account at build time
 2. Making changes to the appearance of a page using the **Presentation Details** interface is much easier if you follow a modular approach
 3. Proper implementation of the Page Editor will **improve the authoring experience** and allow marketers to leverage the capabilities of the **Digital Marketing System**



The experience for the content editor

- Apart from fundamentally building to support the Page Editor interface, there are a number of things developers can do to improve the experience of editors using the Page Editor and reduce the margin for error, this is covered in the next few topics

Apply – Topic 7.1 – 20 min



Create a new bike page

In the following labs you will:

- Use the Page Editor to create a new item based on the Bicycle template
- Populate that item with images and text
- Add several General Widget components into placeholders specifying data sources for each one
- Submit the item through a workflow

Lab A. Create a new bicycle page item

1. Log into the **Page Editor** as **Admin** (password: **b**)
2. Navigate to the **/sitecore-cycling-holidays/Home/bikes** page using the **navigation bar**
3. Use the **Insert Page** command in the **Home** tab to insert a new Bike Page and call it **The Downhill Daredevil**



Figure 7-3 Page Editor ribbon



Knowledge Check

What will the URL of the new page be? What might be a more suitable item name?



Knowledge Check

What has been set up so that an author can only add a bike page?

Lab B. Rename Item and populate fields

1. To **rename** the item click the **Edit** button in the navigation bar
2. In the split screen of **Content Editor** and **Page Editor**, click the **Rename** command in the Ribbon and give the item a name without spaces

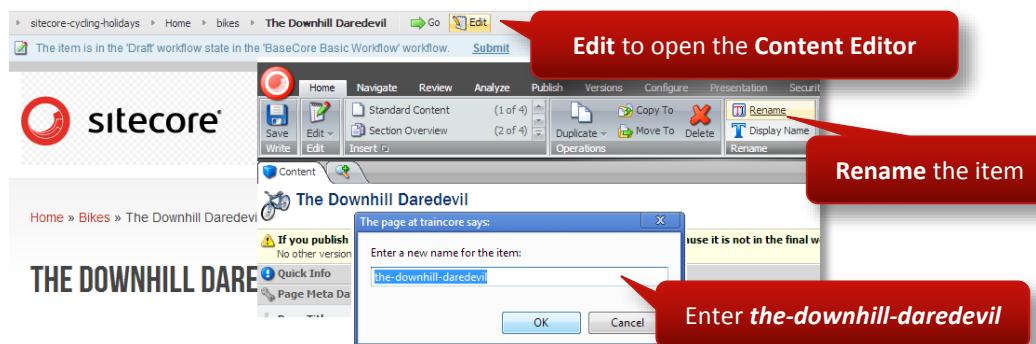


Figure 7-4 Rename item in Content Editor

3. **Save** the item
4. To exit the Content Editor, click the **Sitecore** logo on the top left and then click the **Exit** option
5. Ensure that page is in **Editing** and **Designing** mode by selecting the appropriate checkboxes in the **View** tab



Figure 7-5 Enable Editing and Design mode

6. Populate the page with a large page image and some sample rich text. (Use an existing image from the Media Library - **/BaseCore/Galleries**)

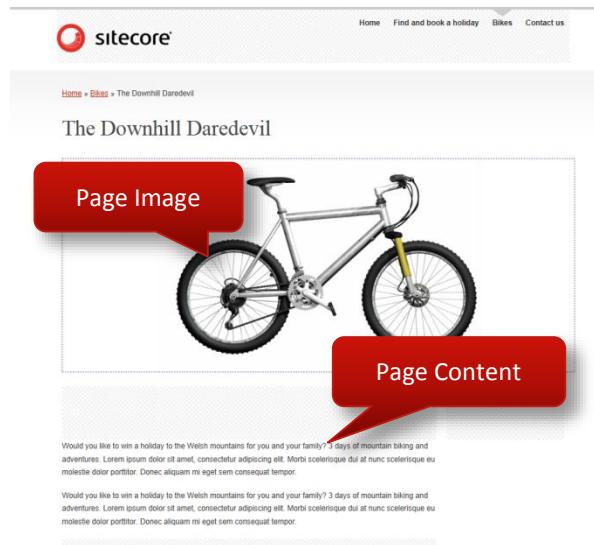


Figure 7-6 Populated page



Knowledge Check

Notice that you can insert any image from anywhere in the site, what problems do you foresee with that?

Lab C. Insert a General Widget component

1. **Insert** a component in the right-hand column of the page (the **tworightcolumn** placeholder)
2. Select the **General Widget** and select the **Open the Properties dialog box immediately** checkbox

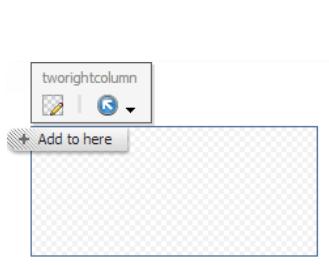


Figure 7-7 Tworightcolumn placeholder

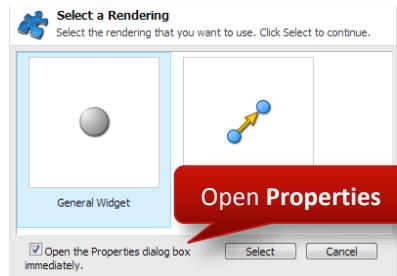


Figure 7-8 Open properties for a component

3. Set the data source to any item under **/sitecore/content/sitecore-cycling-holidays/global/reusable content/**

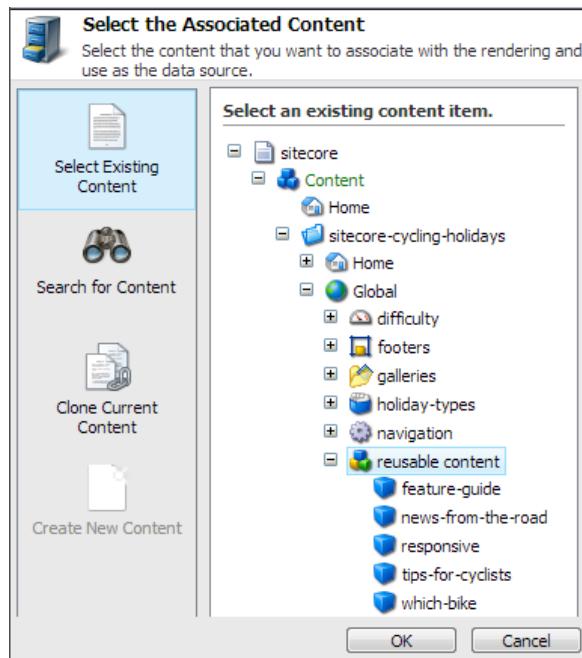


Figure 7-9 Select datasource

4. Find the **widgetscontainer** placeholder above the row of widgets on a blue background



Figure 7-100 widgetscontainer placeholder

5. Insert several widgets into the placeholder. Give each component a **data source** and set the **class**, **heading**, and **width**. (These widgets need a fixed width selected, because these widgets are being placed next to each other horizontally)



Figure 7-111 Select styling for the widget



Tip

If you forgot to change the properties for a component, select the component and then click the **Edit Component Properties** option on the **More** button's drop-down menu

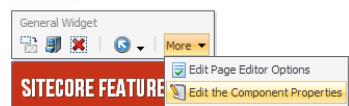


Figure 7-12 Component properties

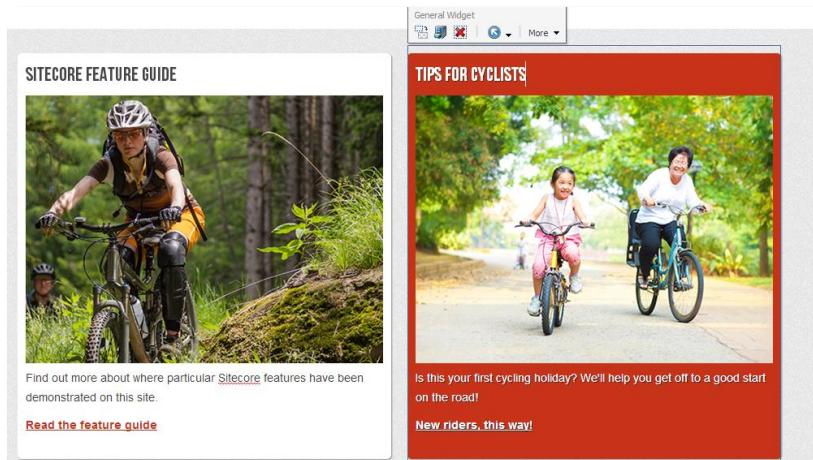


Figure 7-13 Various components

6. Make sure you **submit** the item through workflow

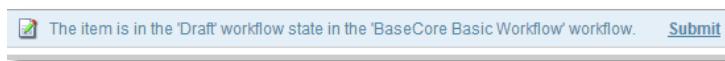


Figure 7-14 Workflow

7. **Publish**, close the Page Editor and view the completed page

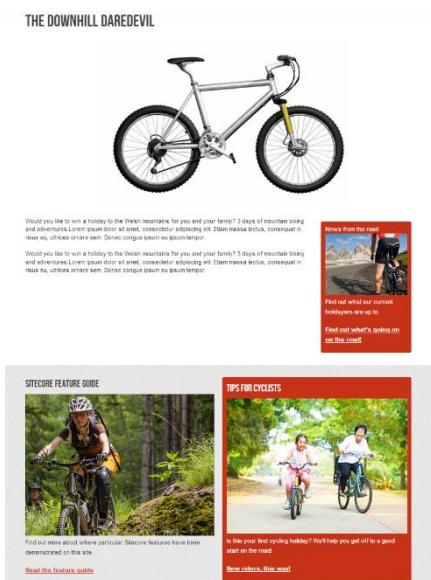


Figure 7-15 Resulting page

Review

Building for the Page Editor

Q What kind of components should be statically bound?

- ✓ Those components that will never move like headers, footers, navigation etc.

Q Why would you make HTML modular?

- ✓ So that a widget can work in several locations

Q List 3 reasons for building for the Page Editor

- ✓ Difficult to transform your implementation into Page Editor friendly solution if you didn't build with it in mind
- ✓ Making appearance changes using Presentation details is easier when page is modular
- ✓ Enhance the author experience as well as leverage the Digital Marketing System

Topic 7.2 Datasource restrictions

Introduction

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

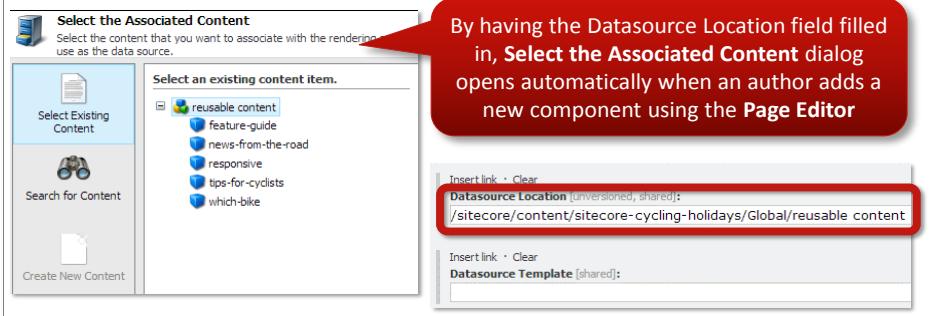
- Specify how to restrict the location and type of data source that can be selected
- State how non selectable items appear in the Sitecore tree

Content

Datasource Location field

On the component definition item

- Developers can specify the start item for a component's data source lookup using the **Datasource Location** field
- Essentially you are filtering what authors can select when they add a component to a page



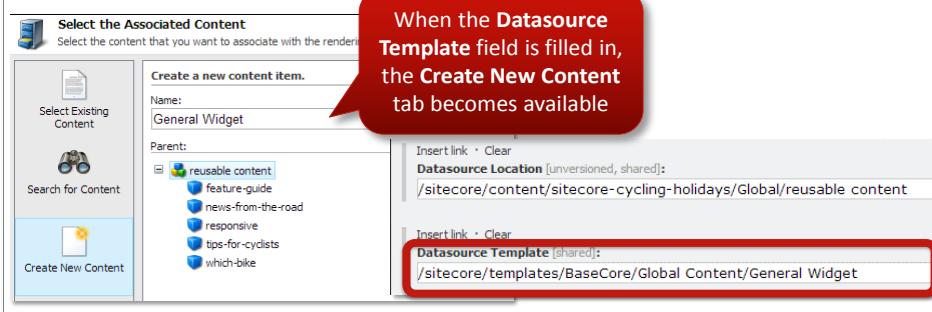
Best Practice

It is a best practice to use IDs instead of paths. This is because if the item is renamed, or changes location, the ID remains the same which ensures a working field

Datasource Template field

On the component definition item

- The **Datasource Template** field limits the type of item that can be created through the **Page Editor** for that component – e.g. a widget component can only accept a widget content item
- Only items of a particular template can be created



What if the Datasource Location field is empty?

On the component definition item

- If the **Datasource Template** field is filled in but the **Datasource Location** is empty
- Select the Associated Content** dialog does **not open automatically**, but can still be accessed by clicking



Demo - Apply restrictions

Apply Restrictions Using the Page Editor

- In Page Editor **design** mode, select the **General Widget** component and in the floating toolbar select **Edit Page Editor Options**
- Fill in the **datasource location** field: **/sitecore/content/sitecore-cycling-holidays/Global/reusable content** then click **OK**
- Add another widget component, a prompt appears to “**select associated content**”
- Notice that you cannot create any new items here, only select existing ones, click **cancel**
- Go into **Edit Page Editor Options** again and this time set the **Template Location** field to point to **/BaseCore/Global Content/General Widget**
- Add another widget to the placeholder, notice that you can now **create new content**, this content is stored in the **datasource location** – location in the tree, and uses the template that you selected
- Show the definition item the **Content Editor**
- Switch to **Content Editor** and navigate to the **General Widget definition item**, path is: **/sitecore/layout/Sublayouts/BaseCore/Widgets/General Widget**
- Notice that the **Datasource Location** and the **Datasource Template** fields are the same as in the Page Editor
- Apply restrictions using **Sitecore Rocks**
- Navigate to the **Home** item and view its **presentation details**
- Double-click** on one of the **General Widget components** to view its **properties**
- Right-click on the **General Widget** component and click the **Edit** option to navigate to the actual **component definition item**
- Notice the same fields as you see in the Content Editor also notice the IDs in fields – this is the developer view

Apply – Topic 7.2 – 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 data source

Lab A. Set Datasource Location

1. If you are not already in the **Page Editor**, log in again
2. If you are not already in design mode, then activate **design** mode
3. Stay on the **Home** page and select the **General Widget** component

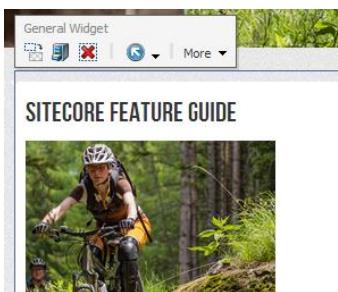


Figure 7-16 Select General Widget component

4. In the **floating toolbar**, click the **More** button and then select **Edit Page Editor Options**

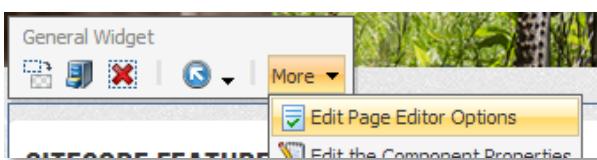


Figure 7-17 Edit Page Editor Options

5. Fill in the **Datasource Location** field with: `/sitecore/content/sitecore-cycling-holidays/Global/reusable content`, then click **OK**



Figure 7-18 Datasource Location field

Lab B. Add a component

1. In the Ribbon, select the command to **add a new component**



Figure 7-19 Add new component using the Page Editor ribbon

2. Click the **Add to Here** button to add a General Widget to the **widgetscontainer placeholder** under the big red widget

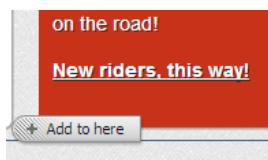


Figure 7-20 Add new widget to widgetscontainer placeholder

3. Notice the **Select the Associated Content** dialog box opens and you can now select items from the datasource that you specified in step 5 of **Lab A**
4. Notice the **Create New Content** tab is greyed out, this means that you cannot create new content, but can only select existing ones
5. Select any **item** from that list and click **OK** and then **save**

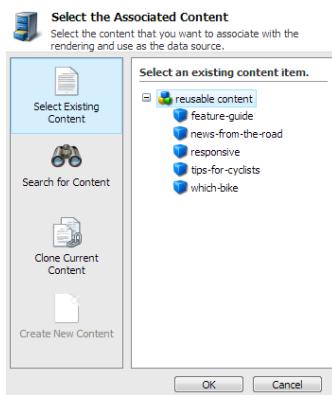


Figure 7-21 Select an item from the list

Lab C. Set Datasource Template

1. Select the **General Widget component** as you did in [Figure 7-16 Select General Widget component](#)
2. Go to its **Page Editor Options** as you did in [Figure 7-17 Edit Page Editor Options](#)
3. This time set the **Datasource Template** field to point to </BaseCore/Global Content/General Widget>

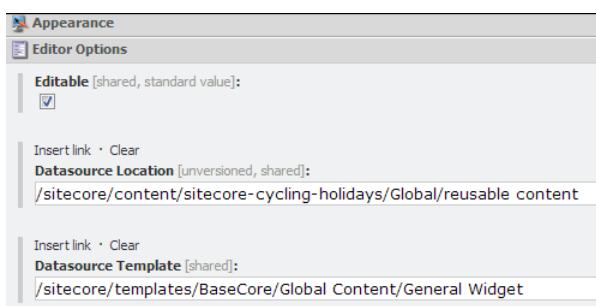


Figure 7-20 Datasource Template field

Lab D. Add another component

1. Add another widget to the **widgetscontainer** placeholder
2. Notice that you can **create new content** based on the **Datasource Template** field
3. This new content is stored in the **Datasource Location** that was specified
4. Either **create a new piece** of content or **select** from an **existing** content

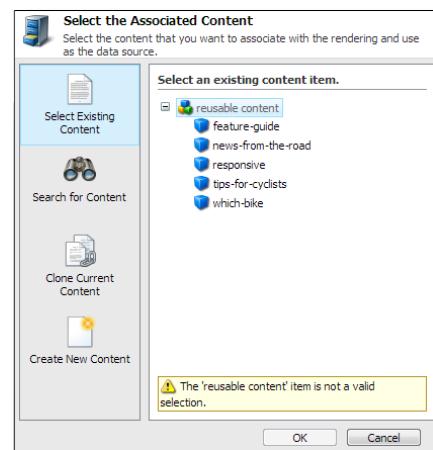


Figure 7-22 Create New Content tab is now visible

Review

Datasource restrictions

- | | |
|---|--|
| <p>Q What does the Datasource Location field automatically do in the Page Editor?</p> <p>✓ Opens the Select the Associated Content dialog when authors add a new component</p> <p>Q What happens if the Datasource Location and the Datasource Template field are filled in?</p> <p>✓ The Select the Associated Content dialog opens and authors can create a new content item using the template specified</p> | <p>Q What happens if the Datasource Location field is empty but the Datasource Template 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 greyed out</p> |
|---|--|

Topic 7.3 Parameters and parameter templates

Introduction

Objectives

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

- Set parameter properties per instance of a component
- Describe how parameter templates limit the type of content an author can enter as a parameter

Content



Demo: Using a parameter template

- Create and assign using Sitecore Rocks
- Show result in Page Editor



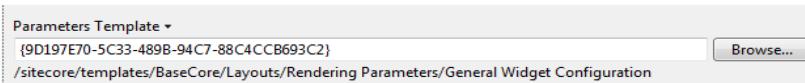
Knowledge Check

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

A better way to allow users to assign parameters is to assign parameter templates to the component

- Specify parameter value field types and sources similarly to normal item layouts
- Base the template on the **Standard Rendering Parameters** template
- Populate it with fields using exactly the same field types as you would for a regular template; e.g. droplink for a CSS class item
- On the actual component definition item, assign the parameter template



- Per instance of a component, you can now use the parameter template properties in the same way as you would on an item

Apply – Topic 7.3 – 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 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/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



Figure 7-23 Parameters Template for the Related Articles sublayout

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

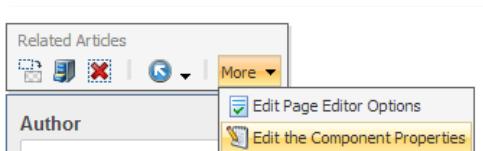


Figure 7-24 Edit Component Properties

1. Confirm that the **Class** field now appears in the component properties window
2. Select **red** from the **Class** drop-down list and click **OK** and **Save**

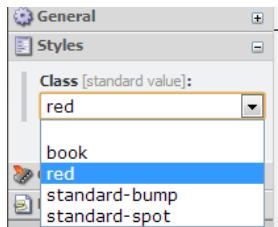


Figure 7-25 Related Articles component properties

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 item you retrieved is based on the `/sitecore/content/settings/styles/widgets/book` data template, and has the **CSS Classes** field, **render the contents** of the **CSS Classes** field
7. **Pass the following parameter** into the **Render()** method `"disable-web-editing=true"` - This parameter prevents the CSS class from being editable in the Page Editor, stopping authors from changing the CSS class by hand as well as preventing your HTML and CSS from being corrupted with Page Editor markup
8. In the .ascx file, use code blocks to output the Class string `<div class="chunk widget">` (Be sure to leave a space)
9. **Save and Deploy** the Visual Studio solution
10. **Browse** to the news article with the **Related Article** component, **confirm** that this component now has the style specified in the parameter template field

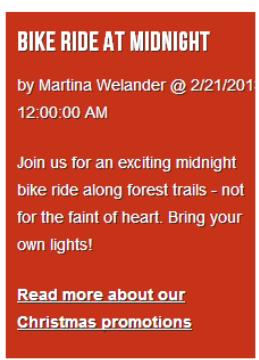


Figure 7-26 Resulting Component

Review

Parameters and Parameter Templates

- | | |
|--|---|
| <p>Q What is the purpose of parameter templates?</p> <p>✓ Parameters allow properties to be set per instance of a component</p> <p>Q What data template does your new data template need to be based on?</p> <p>✓ Standard Rendering Parameters template</p> | <p>Q 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 7.4 Placeholders

Introduction

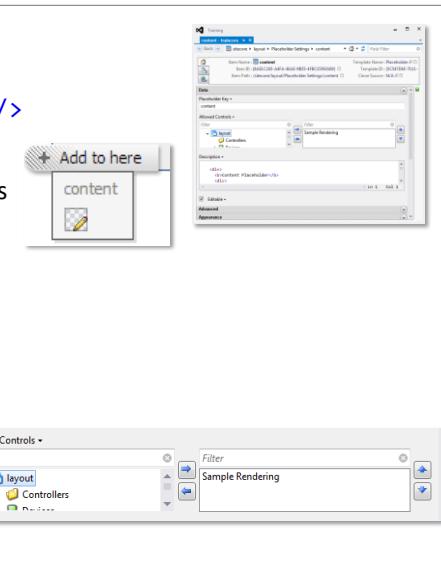
By the end of this topic you will be able to:

- Describe what a Placeholder setting item is
- Describe why they are needed
- Override a Placeholder setting for a particular item

Content

Placeholder settings items

- Placeholder settings items are definition items for the
`<sc:Placeholder Key="content" />`
 key attribute
- They are needed in order for placeholders to be **selectable in Page Editor**
- Placeholder settings items allow you to **specify** which **controls** authors can insert through the Page Editor in a placeholder with a certain placeholder key
- This has no effect for developers using Rocks, or the Presentation Details dialog in the Content Editor



Demo: Create placeholder setting override

- If for instance you have a global placeholder throughout the site, however on one of the pages you need to change the allowed controls for that placeholder, your options are:
 - A – Create a new placeholder control in the markup or
 - B – 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**

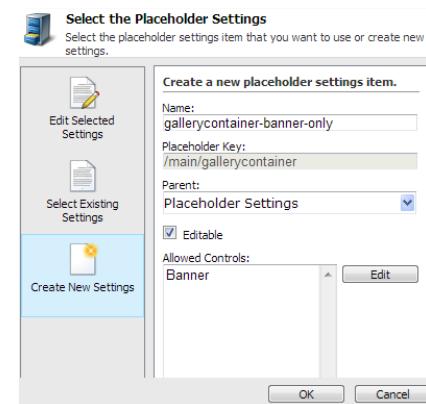


Figure 7-27 Placeholder override

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 being **created**

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**



Tip

Generally, you tend to specify overrides on the item’s template’s standard values, but you can specify placeholder overrides per item

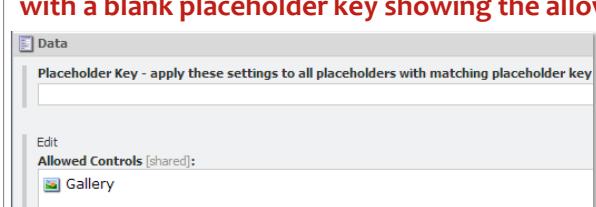
How to override a placeholder setting

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

In the Content Editor

You get a placeholder settings item  content-with-gallery with a blank placeholder key showing the allowed controls



Data
Placeholder Key - apply these settings to all placeholders with matching placeholder key

Edit
Allowed Controls [shared]:
Gallery

Why a blank key?

- You **cannot** have **duplicated keys**
- The reason for the key is selectable in the Page Editor. If you select the parent placeholder which has a key, it has a placeholder settings item pointing to this placeholder with no key. There is a **relationship between the 2 placeholders**

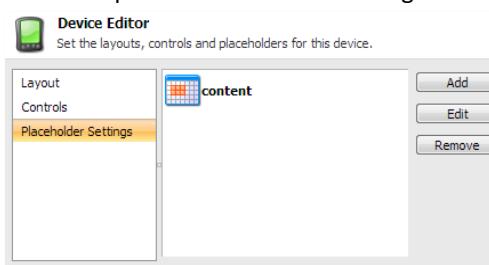
Where to override placeholder settings

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 a one off, you can override placeholders on the template **standard values** using the **Layout Presentation Details dialog**



Apply – Topic 7.4 – 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

Lab A. Override a placeholder setting in Page Editor

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. 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

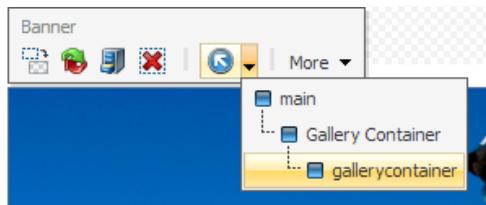


Figure 7-28 Navigate to the placeholder

4. Click **Add to here**

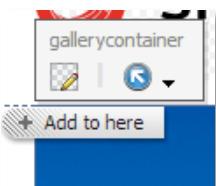


Figure 7-29 gallerycontainer placeholder

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 on the command in the **ribbon** to **Edit the placeholder settings**

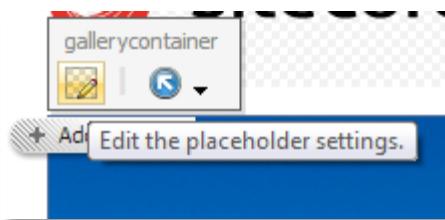


Figure 7-30 Edit the gallerycontainer placeholder through the Page Editor

7. Click the **Create New Settings** tab
8. Name the new placeholder **gallerycontainer-banneronly**
9. In the allowed controls -this is where you specify the **allowed controls** that this **placeholder** can have, set the **Banner** component as an allowed control – path is **Layout/Renderings/BaseCore/Banner** and click ok
10. **Save** the item

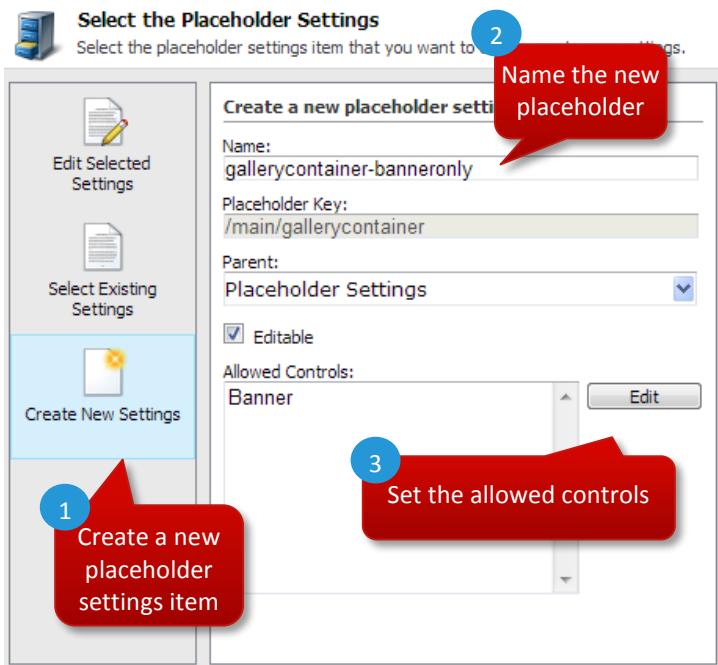


Figure 7-31 Create new placeholder settings

11. Confirm that the placeholder used now is the new one

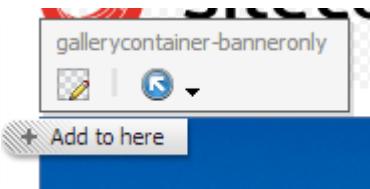


Figure 7-32 New placeholder name

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 – in the ribbon select the **Advanced** tab and click **Details**

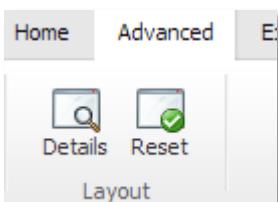


Figure 7-343 Presentation details in Page Editor

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, path is: **/Templates/BaseCore/Pages/Holiday/_Standard Values**
4. Click **Copy** then **OK** on the Layout Details dialog



Figure 7-334 Copy to option in presentation

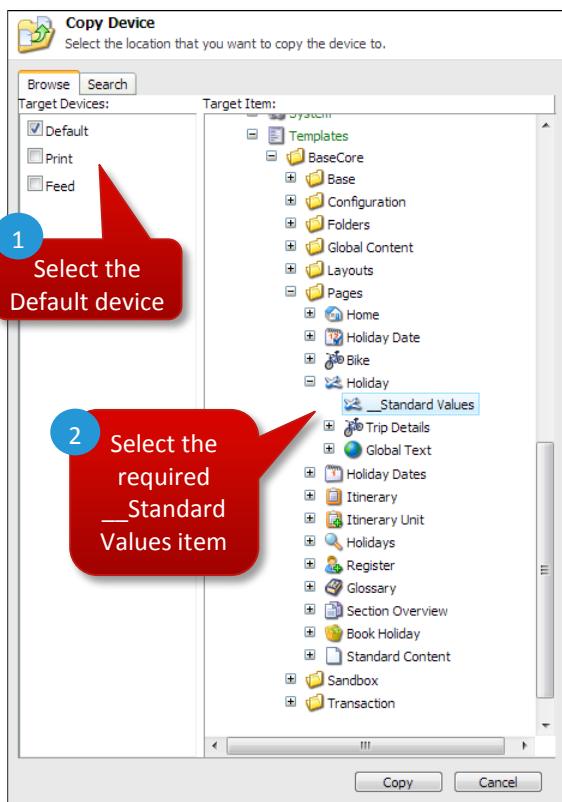


Figure 7-35 Copy presentation details

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



Tip

If you want a placeholder override to apply to all items based on the holiday data template, modify the presentation details on the holiday data template's standard values – add a new placeholder setting that overrides gallerycontainer; this change will then be reflected on all news items

Review

Placeholders

- | | |
|---|---|
| <p>Q What is a placeholder settings item?</p> <ul style="list-style-type: none"> ✓ It's a definition item for the <code><sc:Placeholder Key="myKey" /></code> key attribute <p>Q Why are they needed?</p> <ul style="list-style-type: none"> ✓ They are needed in order for placeholders to be selectable in Page Editor <p>Q What else do they allow you to specify for authors?</p> <ul style="list-style-type: none"> ✓ Which controls authors can insert in a placeholder | <p>Q Why would you override a placeholder settings item?</p> <ul style="list-style-type: none"> ✓ If you have a placeholder settings item with various allowed controls used throughout the site, but then you have one page that needs to display different allowed controls in place of that placeholder – you would override the original placeholder settings item with the required allowed controls |
|---|---|

Topic 7.5 Compatible renderings and allowed controls

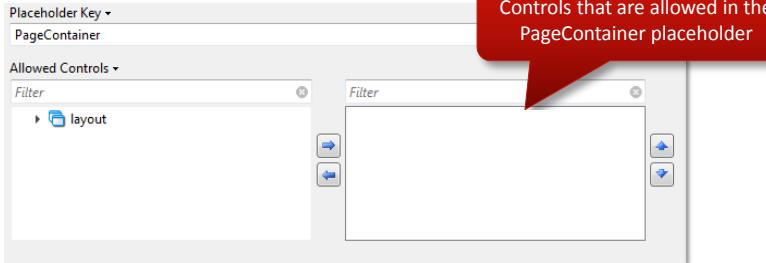
Introduction

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

- State the purpose of Allowed Controls
- Describe what the Allowed Controls list does
- Allow any component to be added to a placeholder in the Page editor
- Describe what a compatible rendering is
- State two aspects of a Compatible Rendering needed to allow it to work

Content

Allowed controls

- **Allowed controls** are specified on placeholder settings items and **determine which components can be added to a particular placeholder** through the **Page Editor**
 - If the allowed controls field is empty, any components can be added to that placeholder
- 
- Controls that are allowed in the PageContainer placeholder
- If a component is **not in the Allowed Controls** list, it **cannot be added** into that placeholder **using the Page Editor** interface

Compatible renderings for authors

- Compatible renderings are components that can be used in place of another component

A good example of a compatible rendering would be:

- Widget Wide Image and Widget Floated Image
- Banner and Scrolling Gallery



Where are Compatible Renderings defined?

Defined on the components definition item

- E.g Banner component definition item has Gallery as a replaceable component

Note: Compatible Renderings are not automatically swappable with each other

- E.g Gallery component definition item has Banner as a replaceable component



Important

Note: Compatible renderings are not by default swappable back and forth with each other, if you want the author to be able to replace Banner component with Gallery and Gallery with Banner you have to make the Banner component compatible with Gallery and then the Gallery component compatible with Banner

Compatible Renderings

Keep in mind:

- Both components need to be able to accept the **same data source**
- Both components need to be able to accept the **same parameters**

To be able to replace a rendering with a compatible one in the Page Editor

- Both renderings need to be in the Allowed Controls of the Placeholder Settings item**



Walkthrough: Allowed controls and compatible renderings

- **Allowed controls**

1. Log onto the **Page Editor** in **Design mode**
2. On the **Home** page click on the **Banner component** and notice the additional command you get in the floating toolbar – **Replace with another component**
3. The replaceable component for Banner is **Gallery**, select **Gallery**, and **save**
4. Click on **Gallery** and **replace it with Banner**
5. Select the **Banner component** again, in the **floating toolbar** click on the **blue arrow** – to navigate to the components **placeholder** (gallerycontainer)



Figure 7-36 Replace with another component command



Figure 7-37 Select the parent placeholder

6. Navigate to the **news article** you created earlier under `/sitecore/content/sitecore-cycling-holidays/Home/news`
7. Select the **tworightcolumn** placeholder and click **Add to here**

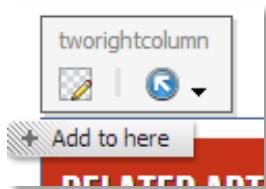


Figure 7-38 tworightcolumn placeholder



Knowledge Check

Why can authors not add Related Articles to the placeholder?

-
8. In the **floating toolbar** select the command to **edit the placeholder settings**
 9. Click **Edit** and **select the Related Articles sublayout**, then click **OK** in both dialogs

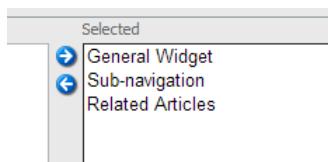


Figure 7-39 Selected pane

10. Select the **tworightcolumn** placeholder and click **Add to here**
11. Now you can **add Related Articles**, go ahead and select it, change its **CssClass** as well

- **Compatible renderings**

We want to make Related Articles compatible with either the General Widget or Sub-navigation component – this saves authors time of having to delete Related Articles and add in the Sub-navigation component

1. Staying in the Page Editor having selected **News Articles**, in the **floating toolbar** select **More > Edit Page Editor Options**
2. Scroll down to the **Compatible Renderings** field and from the left hand side double click **Sub_navigation**, path is: **/sitecore/layout/Renderings/BaseCore/Sub-navigation** and click **ok**
3. Select the component again and notice the compatible component command appeared in the floating toolbar, click the command and **replace News Articles with Sub-navigation**



Figure 7-40 Related Articles compatible rendering command

4. Notice there is **no compatible controls command** for Sub-navigation – that is because in some case you would not want that component to be automatically replaceable with Related Articles. If you do then you need to go in manually and set the Compatible Renderings field for the Sub-navigation component

Review

Compatible renderings and allowed controls

Q What are allowed controls?

- ✓ They determine which components can be added to a particular placeholder

Q Which Page Editor mode supports adding components into placeholders?

- ✓ Design mode

Q If a component is not in the Allowed Controls list, can it be added into that placeholder through the Page Editor?

- ✓ No

Q What are compatible renderings?

- ✓ Components that can be used in place of another component

Q What two elements should you keep in mind when setting up compatible renderings?

- ✓ Both components need to be able to accept the same data source and the same parameters

Q If a component is not in the Allowed Controls list, can it still be used as a compatible rendering?

- ✓ No

Topic 7.6 Advanced Page Editor configuration

Introduction

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

- Define what a Custom Experience button is and where it is displayed
- Create Custom Experience buttons
- Define an Edit Frame and where it can be displayed
- Create an Edit Frame

Content



Walkthrough – Custom Experience Buttons

1. In the **Page Editor** go to any component in **design mode** – the **floating toolbar** view the commands, these can be amended and / or extended

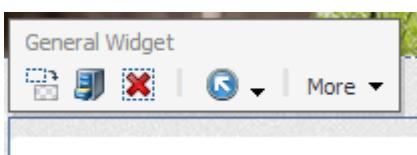


Figure 7-41 Page Editor floating toolbar commands

2. All these commands come from the **core DB**, switch to **Visual Studio**, using **Rocks** navigate to the button definition item in the core DB, path is: **/sitecore/content/Applications/WebEdit/Default Rendering Buttons/Delete**
3. In the Editor pane, change the tooltip of “**Remove component.**” To “**Remove this component NOW**”
4. Switch back to the **Page Editor** and hover over the delete button to see your new tooltip

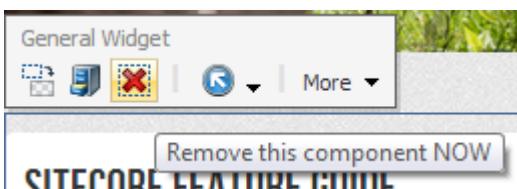


Figure 7-42 Custom command tooltip

5. You can create your own buttons or use some of Sitecore pre-existing ones – in the floating toolbar click **More > Edit Page Editor Options**
6. **Page Editor Buttons** field – select some of the controls and verify that they appear in the floating toolbar

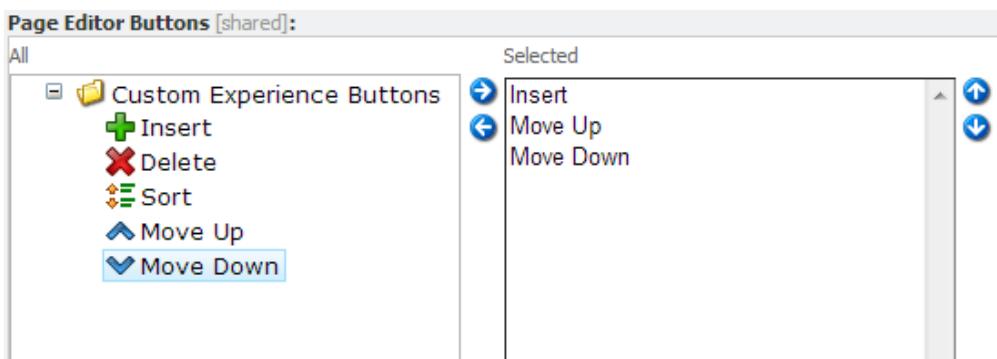


Figure 7-43 Custom Experience Buttons

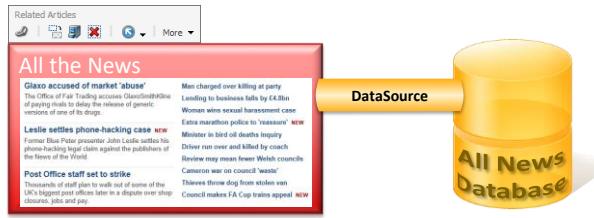


Figure 7-44 Custom Experience buttons in the floating toolbar

- These also come from the Core DB – switch to Visual Studio and in Rocks navigate to: /sitecore/content/Applications/WebEdit/Custom Experience Buttons

Custom experience buttons

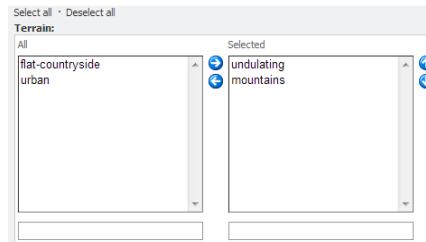
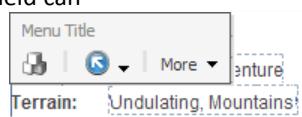
- When you select a **component** in **design mode** or a **field** in **editing mode**, a number of buttons appear in the **floating toolbar** – e.g. ‘Move component’ or ‘Set associated content’
- These **can be extended** using **custom experience buttons**
- Custom experience buttons are created in the **core database** under /core/sitecore/content/Applications/WebEdit/Custom Experience Buttons
- Custom experience buttons are **context aware** – if you have set a data source on your component, the custom experience button will execute in the context of that data source item



Types of custom experience buttons

Field editor button

- For editing a **particular field** or **set of fields** in a **small window**
- For example, a **multilist** field cannot be edited in situ in the same way as a text field can



WebEdit button

- Used for **executing a command** e.g. ‘Delete’ button on a component runs the **webedit:delete** command
- Commands are **configurable**



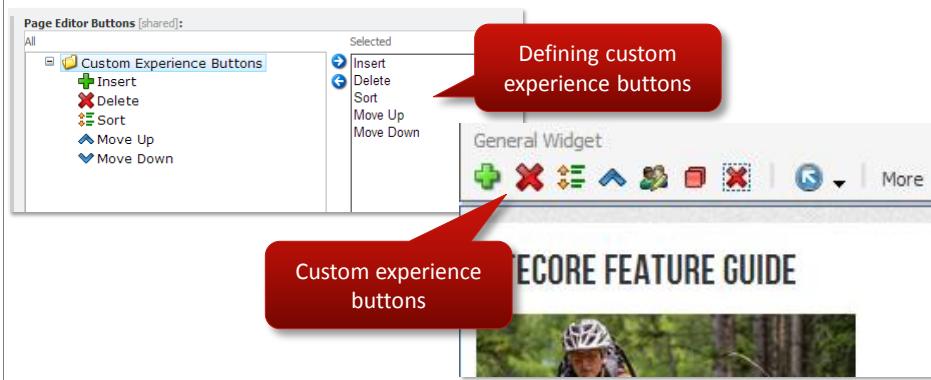
Tip

- All Sitecore interfaces use commands
- Commands are listed in the /App_Config/Commands.config in the file system – they have a name and an associated type, and are built on Sitecore.Shell.Applications.WebEdit.Commands.WebEditCommand
- Either use an existing command or create your own using the WebEditButton data template as a base

Assigning custom experience buttons

Custom experience buttons are assigned on

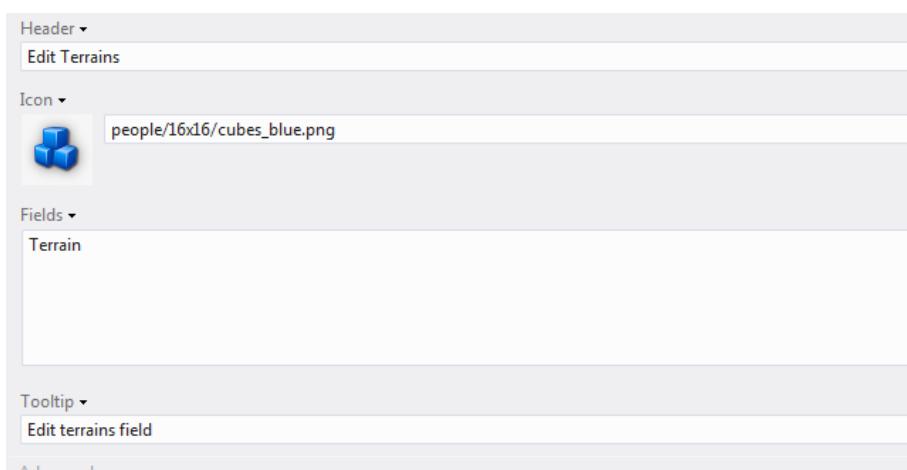
1. A **component** definition item (using the Page Editor or Content Editor)
or
2. A **field** definition item (using the Content Editor)



Demo – Edit frames

The main purpose of an edit frame is to target a particular area of a component

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



7-45 Specify a Header, Fields, and Tooltip fields

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 – e.g. **Battle of the Hills**
11. Switch on **Editing**, but **switch off Designing**
12. Click in the whitespace within the holiday summary list – your edit frame will appear around the content:
13. Click on the **command** in the **floating toolbar**, your fields will appear in pop-up

Edit frames

- By default, buttons appear around components and fields – these are also the lists you can extend using Custom Experience Buttons
- You may want to surround a particular area on a page or component with an edit frame that displays a number of buttons
- Edit frames are created in the **core database** under /core/sitecore/content/Applications/WebEdit/Edit Frame Buttons

Each edit frame has:

- An **Edit Frame Button Folder** that contains a collection of buttons
- Any number of **WebEdit** or **field editor** buttons as children
- By default, edit frames are not aware of data sources – they operate on the context item only, you can assign a datasource to them (like all other Sitecore components).

Assigning edit frames

- Edit frames are assigned in code using the edit frame control with the **Buttons** property set to the path to the edit frame folder in the **core** database:

```
<sc:EditFrame Buttons="My Edit Frame" runat="server" />
```

- You can **surround** any **mark up and/or content** with an edit frame
- You can set the **DataSource** property on an edit frame just like any other control

```
<sc:EditFrame Buttons="My Edit Frame"
DataSource="<%#: Item.ID %>" runat="server">
    <asp:HyperLink runat="server" ID="myHL" />
</sc:EditFrame>
```

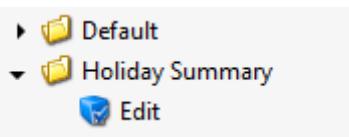
Apply – Topic 7.6 – 15 min



Edit frame

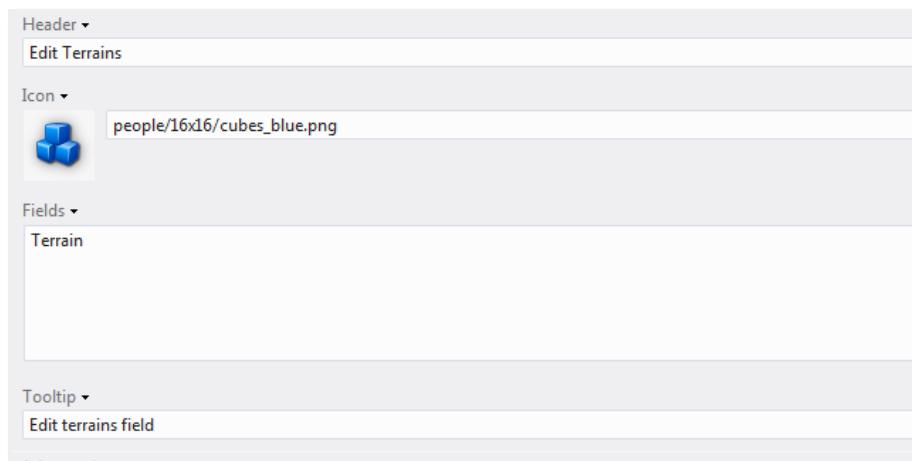
Lab A. Add an edit frame

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



7-46 Holiday summary edit buttons

5. Open the **Edit** item
6. Change the **Heading**, **Fields** and **Tooltip** as follows



7-47 Specify a Header, Fields, and Tooltip fields

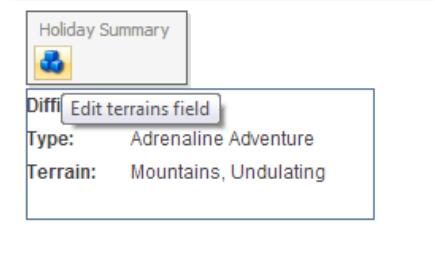
7. Switch to **Visual Studio**
8. Open the **basecore-widget-holiday-summary.ascx** sublayout - </layouts/BaseCore/widgets/basecore-widget-holiday-summary.ascx>
9. Surround the entire definition list HTML (see code sample below) with an `<sc:EditFrame />`
10. Set the **Buttons** property to the **path**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



Code sample – Edit frame

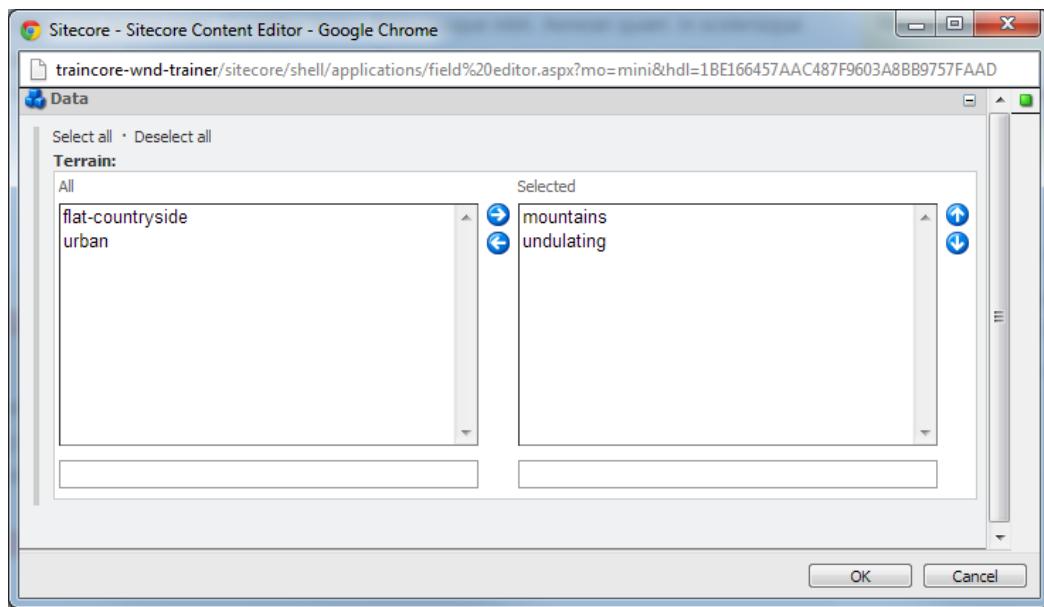
```
<sc:EditFrame ID="EditFrame1" Buttons="/sitecore/content/path/to/buttons"
runat="server">
<dl>
<!-- ... -->
</dl>
</sc:EditFrame>
```

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



7-48 Holiday Summary edit frame

15. Click on the **cubes** command in the **floating toolbar**
16. The **Terrain** field opens up in a pop-up window



7-49 Edit Terrain field in isolation



Knowledge Check

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

Review

Advanced Page Editor Configuration

- | | |
|--|--|
| <p>Q 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>Q Name two examples of Custom buttons</p> <p>✓ Field Editor and WebEdit button</p> <p>Q 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>Q 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>Q In which database do you create the Custom Experience buttons and edit frames?</p> <p>✓ Core</p> |
|--|--|

Extend

- Commands are listed in the /App_Config/Commands.config in the file system – they have a name and an associated type, and are built on the Sitecore.Shell.Applications.WebEdit.Commands.WebEditCommand

Module 8

Marketing Functionality

Contents:

- Introduction to the CEP
- Engagement value and goals
- Profiling and personalization

Topic 8.1 Introduction to the CEP

Introduction

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

- State the three main features of the CEP
- List two types of reports that are available
- Describe why componentization and use of datasources in integral to using the CEP
- Discuss why building for the CEP is much easier than retro-fitting features later on

Content

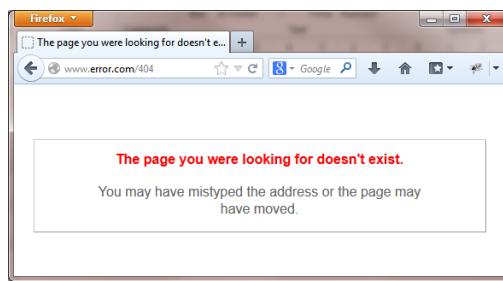
Marketing functionality

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

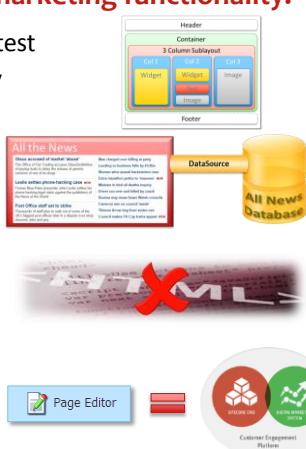
- Lead tracking
- Granular and overview reporting
- Site issue reporting
(e.g., 404, slow pages)



Building for the CEP

How you design and build your site directly affects how effectively a marketer can implement and use the CEP'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're not using the marketing functionality just yet, **build to support it in future:**

1. The implementation of digital marketing capabilities is often a **Phase 2** consideration – be prepared for it. **Retrofitting is much harder than planning ahead**



2. 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



Review

Introduction to the CEP

- Q** Name three key marketing features offered by Sitecore CEP:
- ✓ Measure engagement using **goals**, **personalize** visitor experience, optimize by **testing**

- Q** Why should you build to support marketing functionality from the very beginning?
- ✓ Clients often want to leverage marketing functionality in Phase 2 – be prepared for them or face awkward rebuilds

Topic 8.2 Engagement value and goals

Introduction

By the end of the topic you should 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 goals you have and how much they are worth is crucial to the collection of meaningful data.

Customers are strongly encouraged to attend the CEP Scoping Workshop:

<http://www.sitecore.net/unitedkingdom/Support/Consulting-Services/Business-Optimization-Services/CEP-Scoping-Workshop.aspx>



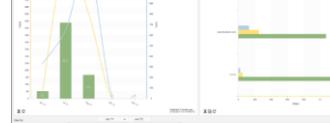
Demo – Create and assign a goal

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. In the **Review** tab, **deploy** the goal (it is automatically part of an approval process and needs to be published, we will talk 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 in the **Analyze** tab
 5. Open **another browser** (e.g., 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



| Rank | Name | Value | Visits | Revenue | Activity Level |
|------|-----------------|-------|--------|---------|----------------|
| 1 | John Doe | \$100 | 100 | \$100 | High |
| 2 | Jane Smith | \$80 | 80 | \$80 | Medium |
| 3 | Mike Johnson | \$60 | 60 | \$60 | Low |
| 4 | Sarah Lee | \$40 | 40 | \$40 | Medium |
| 5 | David Wilson | \$30 | 30 | \$30 | Low |
| 6 | Emily Davis | \$20 | 20 | \$20 | Medium |
| 7 | Alexander Green | \$10 | 10 | \$10 | Low |
| 8 | Bethany Blue | \$5 | 5 | \$5 | Low |
| 9 | Caleb Grey | \$3 | 3 | \$3 | Low |
| 10 | Daniel White | \$2 | 2 | \$2 | Low |

Top leads



Demo – Executive Insight Dashboard and Engagement analytics

1. Open the **Executive Insight Dashboard**
 2. 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
 3. Note that you can view by year, month, week or day
 4. Note the other overview statistics that the Dashboard provides: average value per visit by entry page, search key word, source and referring site
 5. By contrast, **Engagement Analytics** reports are much more granular. Expand the **Reports** option. Click on the **Sales > Top Leads by Value > Unclassified organizations** options. (Make sure that you have selected a wide date range)
 6. Note that you can apply filters, subscribe to a particular lead, or view more details about that lead (including drilling down to individual visits)

7. Locate the session where you completed the holiday booking goal. Note that the visit has an engagement value associated with it
8. Engagement Analytics reports also provide statistics on particular events to do with site health (e.g., common mistakes, 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

The screenshot shows the Sitecore Testing interface. On the left, a preview of a page with a 'General Widget' is shown, with a red callout pointing to the 'Variation Name' dropdown labeled 'Define variations'. In the center, a 'Testing' dialog box is open, showing a 'Start Test' button, a 'Stop Test' button, and a table titled 'Current Variation: B. Variation Name'. The table lists two variations: 'B. Variation Name' with a value of 0 and 'A. Variation Name' with a value of 0. A red callout points to the 'Start Test' button labeled 'Start and stop tests'. Another red callout points to the table labeled 'View results'.



Tip

Testing needs to be thought about 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 (e.g., button text or form labels), these values need to be defined on your data source (e.g., 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 data sources



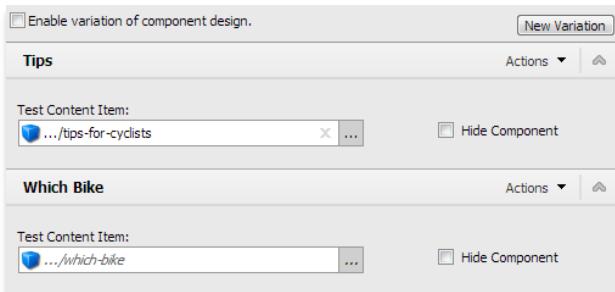
Demo - Setting Up a Multivariate Test

1. Open the homepage in **Edit mode** and make sure the **Designing** checkbox is selecting in the **View tab**
2. Select the component you wish to test. Click the **Test the component** button in the context menu



8-1 Test Component button

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



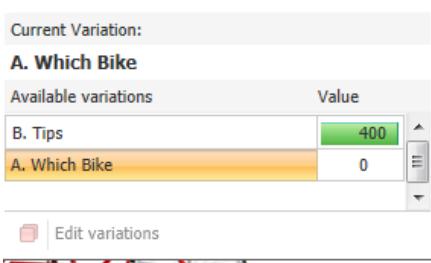
8-2 Configure personalization behavior

4. Notice that you can scroll between the variations

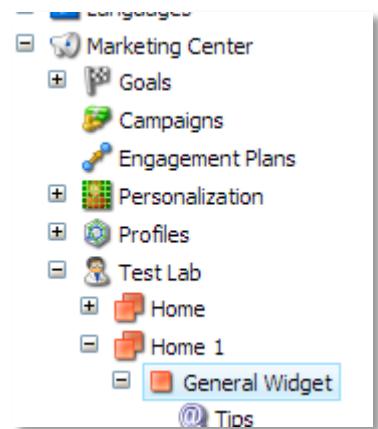


8-3 Testing different variations

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 (e.g., whether to make the variation sticky or random)
6. Look at the homepage in different browsers, using different sessions to emulate different visits (e.g., in Internet Explorer, select **File > New Session**), and 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



8-5 Test results



8-4 Test item

9. Stop the test, 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 8.2



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. In 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 in the **Analyze** tab, and selecting the goal you created

Review

Engagement Value and Goals

Q Engagement is measured using...

✓ Goals

Q Give an example of a business-critical goal in the context of the Sitecore Cycling Holidays company.

✓ Booking a holiday

Q What reporting interface would you use to see trends over time?

✓ Executive Insight Dashboard

Q Where would you go to see a detailed report on a visit?

✓ Engagement Analytics reports

Q In what way does

componentization and the use of **data sources** support testing?

✓ Tests set up to change component or component data source out of the box – hardcoded components and/or content not supported

Topic 8.3 Profiling and Personalization

Introduction

By the end of this topic you should be able to

- Set up profile dimensions and keys
- Describe a profile card in terms of how it 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

Why profile?

- To profile content means to **tag** content with **keys and values** – e.g., the Battle of the Hills holiday is **Adrenaline: 10** but **Relaxation: 0**
- Profiling supports **personalization** – **predict** the type of content a visitor is interested in – if I am interested in bikes for experts, why show me tips for new cyclists?

The screenshot shows a website layout. On the left, there's a main content area with a large image of a bicycle and the text 'BIKES FOR EXPERTS'. On the right, there's a sidebar with the heading 'TIPS FOR CYCLISTS' and a small image of two people cycling. Below the image, the text reads: 'Is this your first cycling holiday? We'll help you get off to a good start on the road!'. At the bottom of the sidebar, there's a red link that says 'New riders, this way!'. The overall design is clean and professional.

💻 Draw it – Profiling taxonomy

- As a marketer, you may want to associate your content with specific keywords (e.g., for “cars” you might have “SUV,” “hatchback,” “sedan,” “off-road,” “family, rally,” etc.). These keywords are called **Profile Keys**
- Those keywords can be grouped (e.g., “family rally” and “off-road” are **suitability**, whereas “SUV,” “hatchback” and “sedan” are **types**). The groupings are called **Profile Dimensions**



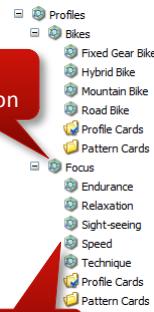
Profiling taxonomy

1. Open the **Marketing Center**
2. Look at the profile dimensions and profile keys that have been set up in Sitecore
3. Look at the minimum and maximum values available for each key

Profiling taxonomy

Score pages based on what type of content they contain

- Set up a **taxonomy** for your implementation – the rules for how content is **classified** in your implementation
- You can have any number of **Profile Dimensions** (e.g., **Focus**)
- Each dimension can have any number of **Profile Keys** (e.g., **Endurance**)
- Each profile key has a **minimum and maximum value** that can be assigned to it – e.g., the Battle of the Hills holiday would score low for **Relaxation** and **Road Bike**



Profile keys



Draw it – Profiling in bulk

Marketers may find themselves assigning the same combination of scores multiple times:

Family: 8

SUV: 10

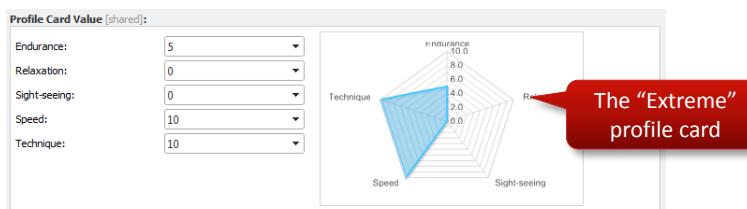
Off-Road: 8

These scores can be grouped into a profiling pre-set that can be used again and again

Profile cards

For ease of use, group your values into profile cards

- You can create a set of **profile key values** within a dimension that can be re-used
- Give the profile card a name that represents the **combination of values** – e.g., “Extreme”
- The “Extreme” profile card scores high for speed and technique, but low for relaxation and sight-seeing

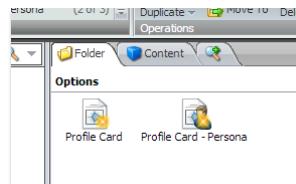


The “Extreme” profile card

Profile cards – personas

If you work in Information Architecture or User Experience, you may be used to representing your target demographic as a persona

- Sitecore has a special profile card for personas
- They do exactly the same thing as regular profile cards
- The only difference is that you can add a bit more information about your fictional character
- Marketing tool only – makes no difference to a developer



Profiling content

Score items in each relevant profile dimension

- Bulk assign values using a **profile card**
or
- Assign **custom values for each profile key** (this will be saved as a custom profile card for that particular item)



Best Practice

Encourage authors to make content profiling a routine task (or risk ending up with 40,000 items that require profiling) and 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 – Create and assign a profile card

1. Using the Sitecore Desktop, open the **Marketing Center** and navigate to **/profiles**
2. Expand **Setting**, **Focus**, and **Bikes**. Note the different profile keys available
3. Click on one and note that you can set the **MinValue**, **MaxValue**, and **Default**
4. Look at one of the profile cards Note that you can assign pre-defined values for the profile keys in the dimension you have created the profile card – grouping common sets of profile keys and values makes it easy to tag similar content
5. Create a persona card under **/Focus/Profile Cards**. Note that it has the same fields as a regular profile cards, as well as some extra fields that identify it as a persona

6. Call the persona card **Mike**, and assign profile values that match the description for his persona (e.g., a technically competent cyclist [Technique: 8] and a speed freak [Speed: 10]). Provide a name, description and image
7. Open the Content Editor and navigate to </sitecore/content/sitecore-cycling-holidays/home/holidays/cycle-south-east-asia>
8. Click the **icon** to open the profile cards dialogue box
9. Edit the **Focus** dimension and assign the **Mike persona profile card**
10. Save the item
11. Note that the **Mike** profile card now appears as an icon at the top of the page



Personalization

- As users navigate around the site, they acquire profile points depending on the type of content they visit
- You can **personalize content** based on the total **profile points** a user has accumulated
- You can also personalize based on what **goals or events** a user has triggered
- Personalization is done at **component level**
- You can either profile on **individual profile keys** (e.g. Focus > Technique) or create a **pattern card** that matches multiple profile keys within a profile dimension (note that **profile cards** are only a tool used to tag content – you might have a pattern card that looks identical to a profile card)

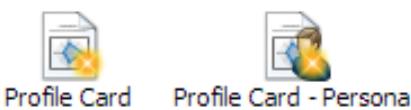


Demo – Personalize based on a pattern card

1. In the Page Editor, navigate to the homepage
2. Select the **Gallery component** and click the **personalization** icon
3. If the visitor matches the **Mike** card, change the data source to another gallery

Profile cards vs. pattern cards

- **Profile cards** are a tool for marketers to categorize content in bulk



- **Pattern cards** are a tool for personalizing content based on points a visitor has accumulated



- You might have 20 profile cards used by authors and only 4 tightly controlled pattern cards
- You might have exactly the same profile cards as pattern cards

Review

Profiling and personalization

Q What feature might an author use to bulk-assign profile points?

✓ Profile card

Q Profile keys are grouped into...

✓ Profile dimensions

Q Personalization is done on individual profile keys or how closely a visitor matches a particular...

✓ Pattern card

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 9

Dealing with Your Data

Contents:

- Item buckets
- Search

Topic 9.1 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 bucket-able 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

1. Using the Desktop interface, open the **Content Editor**
2. Right-click in the **gutter** and select **Item Buckets**, it will highlight items in the tree that are buckets
3. Navigate to **/sitecore/content/sitecore-cycling-holidays/Bookings**
4. Click the **Search** tab
5. Type * to show all results
6. Click on a search result to open it, demonstrate that all normal practices are still possible on items within a bucket by duplicating and editing an item
7. Click the **Search** button to refresh the search, the duplicated item should appear.
8. Note the list of filters in the right column on the screen. These are known as result **facets** (e.g., by default, Sitecore facets on **Template Name** and **Language**, among other things)

Searching a bucket

Plain text search supports

1. wildcards (*)
 2. replacements (?)
 3. exact text phrases
- Use any number of **search filters** to restrict your search
 - You can also perform **search operations** on the results – e.g., copy results to another location

What is 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 the Sitecore search interface. On the left is the Sitecore navigation tree, which includes 'Content', 'Home', 'sitecore-cycling-holidays' (with 'Bookings' under it), and 'sitecore-cycling-holidays-corporate'. A red callout box points to the 'Bookings' item in the tree with the text: "'Bookings' item bucket with hidden descendants within traditional tree structure". The main search interface on the right shows a search bar with a placeholder of '*'. Below the search bar, a red callout box points to the search interface with the text: 'Buckets search interface'. The search results area displays two results found in 0.00.2063 seconds under the 'Bookings' item. The results show a 'text' item with the following details: Template: Holiday Booking Location: Bookings, Version: 1, Created: 2013-04-16 By: sitecore\admin. It has three language versions: en-US, fr-FR, and sv-SE, each with a 'Translate' button. To the right of the results are facets for Language (English 2), Template (holiday booking 2), and Author (sitecore\admin 2).

- 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 – e.g., products, repositories, orders
- Automatically organises 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:

1. You **do not** care about the structure of the data stored in the bucket
2. You anticipate a large volume of items
3. You would benefit from searching for items rather than locating them in the tree



Demo – Creating a bucket

The reusable content folder is a good candidate for being turned into a bucket because we are not concerned with the hierarchy of the items beneath it. It is easier for authors to search for what they need

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 (e.g., Gallery View for image heavy items), or choose 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 *. Notice that there are no widgets listed in the search results. Why is this the case?
8. If you expand the reusable content item, its children all appear as normal items because the items have not been made bucketable
9. Select one of the **children** of the reusable content item. Notice the message at the top of the item:

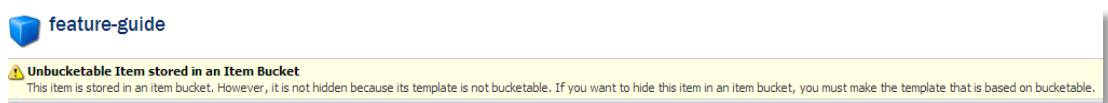


Figure 9-1 Content Editor warning relating to buckets

10. Navigate to the item's data template and create a standard values item if one does not exist
11. On the data template's standard values, find the **Item Buckets** field section, and select the **Bucketable** checkbox
12. **Save** the item and navigate back to the bucket
13. Because you have made changes pertaining to the bucketability 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
14. In the **Configure** tab, click the **Sync** button. The bucket's children disappear
15. Try searching – you will now see the children of the reusable content item in your search results
16. 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
17. In the **View** tab, check 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



Tip

You can make a particular template a bucket by default by selecting the template's standard values and clicking the **Bucket** command in 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** (e.g., 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 (e.g., 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 – e.g., 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 they will be included in the index that the bucket queries (*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





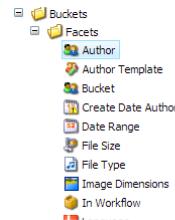
Demo – Configuring facets

1. Using the Content Editor, navigate to **/System/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.
5. Navigate to the **Image Dimensions** facet. Notice that the field name here is *calculateddimension*. This is another *calculated* field whose contents exist in the index.
6. 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?

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 9.1 – 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**:



Figure 9-2 The buckets chunk



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**



Figure 9-3 The 'Bucketable' checkbox

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 are hidden and that search and faceting works as expected

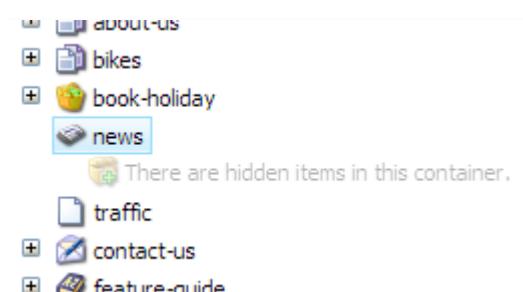


Figure 9-4 Bucketed news items

Review

Item Buckets

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

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 9.2 Search

Introduction

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 it is stored
- Facet on a set of results

Content

Simple searching

- The item buckets search functionality uses the Sitecore search API
- You can leverage the same API to search an index from the website's front end

A simple search example

```
Get index by name  
Create a search context  
Return type  
Return an instance of  
IQueryable<T> and use standard  
LINQ statements
```

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

Basic index configuration

When you search, you are querying an index of Sitecore items

- Defined in /App_Config/Include and prefixed with Sitecore.ContentSearch
- Configuration files determine **how** Sitecore items should be indexed

Index-specific configuration, like the index name and database, is defined in individual files – e.g.,
Sitecore.ContentSearch.Lucene.Index.Master.config

Shared configuration is defined in a default configuration file – e.g.,
Sitecore.ContentSearch.Lucene.DefaultIndexConfiguration.config
(you can **override any defaults** in the index-specific configuration files)

The provider model

- Search uses a **provider model**; you can plug in whatever search platform you want – Sitecore ships with support for **Lucene** and **Solr**, but you can build your own (e.g., for ElasticSearch)
- Identical code to search** is translated by provider into something that the respective search technology understands



Lucene and Solr

- Lucene** is a full-text search library written in Java
- Solr** is a **stand-alone web app** that wraps Lucene and extends it to include more advanced search features (e.g., faceting, distributed indexing, replication, centralized configuration, rich document handling, etc.)

(Solr index and search commands sent to REST-like service)

Which one should I use?

If you want your solution to **scale**, use **Solr**.



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, whereas Solr offers a scalable, enterprise-level solution



Walkthrough – Index Viewer

1. Using the **Sitecore Desktop**, click **Start > Control Panel > Indexing > Indexing Manage**.
2. Notice that there is an additional option for Solr:

 Generate the Solr Schema.xml file
3. By default, Sitecore defines three indexes – one per database (master, core, and web)
4. Click the **Rebuild** button
5. Using the file system, navigate to the **/Data/indexes** folder. You'll see one folder per index

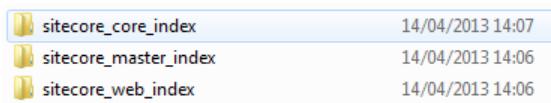


Figure 9-6 The index folders

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

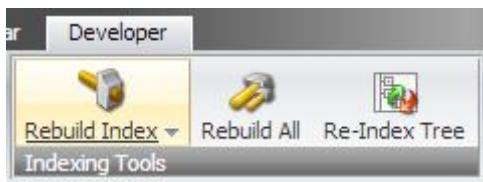


Figure 9-7 The rebuild Index buttons

Searching an index with LINQ queries

Once you have retrieved an index, how do you search it?

- Create a **search context** from your index
- Retrieve an **IQueryable<T>** object from the search context
- Use **standard LINQ queries** to query your data – much like LINQ to SQL or LINQ to Objects

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

Create a search context
(already a singleton)

Get an
IQueryable<T>

Query stored fields
using LINQ

e.g., any indexed item where
heading contains bikes

Default Sitecore indexes

More information about each index

Rebuild Search Index
Select All Unselect All

Local Indexes
 sitecore_core_index
 sitecore_master_index
 sitecore_web_index

Index Statistics

sitecore_core_index

Rebuild Time: Never Run

Approximate Throughput: 0 items per second

Has Deletions: True

Is Clean: False

Out of Date : False

Document Count : 19848

Figure 9-5 Selecting the index



Demo – Executing a simple search

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



Code sample – Sitecore property

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

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



Code sample – Developer-created property

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

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



Code sample – Using ‘Between’

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

LINQLayer `IQueryable<T>` options

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

| Supported | Unsupported |
|-------------------------------|-----------------------|
| <code>.Where</code> | <code>.GroupBy</code> |
| <code>.Any</code> | <code>.Union</code> |
| <code>.OrderBy</code> | <code>.Except</code> |
| <code>.SingleOrDefault</code> | <code>.All</code> |
| <code>.Count</code> | |
| <code>.Select</code> | |

- Custom extensions

| Additional extensions | |
|---|--|
| <code>.Paths</code> (limit item path) | <code>.Between</code> (exclude/include start and end ranges) |
| <code>.Boost</code> (make this part more important) | <code>.FacetOn</code> |
| <code>.Page</code> (does skip/take) | |

LINQ examples

Fuzzy search

```
queryable.Where(x => (x["page_heading"].Like("citecoar", 0.8)));
```



Pagination

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

- combines `.Skip()` and `.Take()`



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



Tip

To allow wildcard searches, use the following extension (note that this works for wildcards as well as replacements):

```
.Where(x => x.MatchWildcard("my * search term"))
```



Demo – `.Paths()` and Pagination

- Use `.Paths()` to restrict search location
- Use `.Page()` to paginate

Custom search result type

Sitecore provides `SearchResultItem` to get you started:

You could create your own public .NET class with two key features:

1. An empty **constructor**    **Getters & Setters**
2. Public properties with getters and setters and/or a public indexer to hold the search result data  **OR**  **Public indexer**
- The LINQ provider will automatically **map document fields** in the index to **properties** on the custom search type by the names of the properties

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

(Recommend inheriting `SearchResultItem`)

Standard .NET class

PageHeading property mapped to the page_heading index



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` to avoid having to remember standard field names



Demo – Create a Search Result Type

1. Create a custom search type called **`CustomResult`** – include a number of different fields from base templates
2. Inherit from Sitecore's base **`SearchResultItem`** class
3. Any index fields with a space are stored as **`field_name`**. Decorate the corresponding property (fields without spaces do not need to be decorated)
4. Refactor simple search to use your new custom class. You will now be able to query by properties rather than string fields

Querying Properties

Instead of

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

You can use:

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

Supported types:

- | | | | |
|--------------------------------------|------------------|--------------------|----------------------------|
| ● .NET built-in integral types | 123456 | ● DateTime | 24th Oct |
| ● .NET built-in floating-point types | 123.123 | ● GUID | ABCD000134532 |
| ● Boolean | Yes/No | ● Sitecore ID | ag2hvf56433 |
| ● String | \$abcdefg | ● Sitecore ShortID | 23ghd4 |



Tip

Sitecore provides a starting point for a search result item in:

Sitecore.ContentSearch.SearchTypes.SearchResultItem. Inheriting gives you properties like: .Template, .Language, .ID, etc.



Tip

The item's GUID is stored in a field called _group, so your ID property would look like this:

```
[IndexField("_group")]
public string Id
{
    get
    {
        return ShortID.Decode(this._group);
    }
}
```

Getting results

A query returns an `IEnumerable<T>`

```
var results = 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);
```



Best Practice

When outputting 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



Demo – Search results object

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 – saves you from having to do multiple queries), and a collection of hits

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 your ‘Where’ to filter down result set

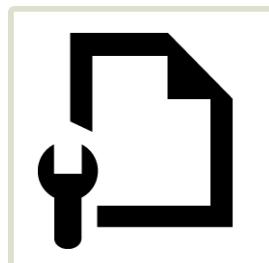


Tip - Security

Huge improvements to security are being made in Updates 1 and 2. Until then, refer to Stack Overflow for options: [http://stackoverflow.com/questions/16683487/indexing-sitecore-item-security-and-restricting-returned-search-results](http://stackoverflow.com/questions/16683487/indexing-sitecore-item-security-and-restrictingReturned-search-results)

Index configuration

- By default, the majority of index configuration is shared between indexes and stored in `.DefaultIndexConfiguration.config` per provider type
- So for Lucene the 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., single-line text fields) or individual field (e.g., the Page Heading field) level

```
<fieldTypes hint="raw:AddFieldByFieldType" />
<fieldNames hint="raw:AddFieldByFieldName" />
```

- The configuration files allow you to control what goes into 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

Standard field using a particular analyzer

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



Tip

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



Tip

*If you have spaces in your field name (e.g., Page Heading), they will be replaced with underscores. Your indexed field name will be **page_heading**. Decorate your search result class properties with **IndexField** to specify an index field name that is different from the property name. For example:*

```
[IndexField("page_heading")]
public string PageHeading { get; set; }
```

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

Stores the original value in the index

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

Places just a pointer to the field in the index

- 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

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 Guide to Item Buckets and Search

Apply – Topic 9.2 – 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

```
namespace Training.BaseCore.Layouts.Content
{
    /// <summary>
    /// The news listing sublayout.
    /// </summary>
    public partial class NewsListing : System.Web.UI.UserControl
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Item newsItem = Sitecore.Context.Item;

            if (newsItem.TemplateID == TemplateReferences.NewsListing)
            {
                List<Item> descendants = newsItem.Axes.GetDescendants().ToList();

                if (descendants.Any())
                {
                    rpNewsListing.DataSource = descendants.Where(x => !String.IsNullOrEmpty(x["Page Heading"]));
                    rpNewsListing.DataBind();
                }
            }
        }
    }
}
```

Figure 9-8 Old news listing code

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-BED 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.SitecoreSearchResult** base class
5. Add the properties specified in **9-9 NewsResult class properties**
6. Use the code example below as a guide, remember that **spaces** in field names are **replaced by underscores**:



Code sample – Search class property

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

| Property name | Property decoration |
|---------------|---|
| PageHeading | <code>[IndexField("page_heading")]</code> |
| PageSummary | <code>[IndexField("page_summary")]</code> |

9-9 NewsResult class properties



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)
8. 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 9 > Topic 9.2 > Lab A > **basecore-news-listing.ascx.cs**). It is recommended that you refer back to the materials on how to construct a query
9. **Retrieve the web index** – either by retrieving it **by name** (`sitecore_web_index`), or by using a **SitecoreIndexableItem**. (This object constructed by passing a Sitecore item in as a parameter – see the code example below. For ease, retrieve 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);
```

10. Create a search context from that index
11. Use the `GetQueryable()` method on the search context object to retrieve an `IQueryable`
12. 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 (e.g., `Home` would have the ID for the `Content` and `sitecore` items):

```
x.Paths.Contains(item.ID)
```

13. Order by the `CreatedDate` property (found on the `SitecoreSearchResult` base class)
14. 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();
```

15. The `SearchResults` object has a property called `.Hits`, it's a collection of `SearchHit` objects, each of which has a `.Document` property. The `.Document` property is a `NewsResult` object.
16. Select out a list of `NewsResult` objects:



Code sample – Search class property

```
results.Hits.Select(x => x.Document)
```

17. Bind results to a repeater, delete the existing repeater and use the sample provided in the **student resource folder** (WND Labs > Module 9 > Topic 9.2 > Lab A > `basecore-news-listing.ascx`)
18. Make sure you change the `ItemType` attribute on the repeater to your custom results class
19. Use the format in the file to output properties of your objects. Notice the colon before the # symbol. All data should come from the index; you should not need to retrieve the item from Sitecore. For this particular exercise, retrieve the item by ID from the database and use that to retrieve the URL – in upcoming version of Sitecore 7, the item URL will be stored in the index and a trip to the database will not be required



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

20. In the file system, locate the **/App_Config/Include/Sitecore.ContentSearch.Lucene.DefaultIndexConfiguration.config** file in your web root
21. In the **raw:AddFieldByFieldName** section, add the **Author**, **Date**, **Page Heading**, and **Page Summary**. You can duplicate the sample entry for **text** or **title** at the very bottom of the list. Make sure you use the field name as it would appear in Sitecore, not as it would appear in the index
22. Make sure the **storageType** attribute is set to **YES**
23. 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)
24. **Save and deploy** the solution
25. Navigate to the news listing page on the site
26. 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

Review

Search

- | | |
|--|---|
| Q What defines how Sitecore items should be indexed? ✓ Index configuration files in /App_Config/Include | Q What is the name of Sitecore's default search result class? ✓ SearchResultItem |
| Q Search is built using a provider model. What does this mean? ✓ You can plug in whatever search provider you want | Q When building your own search result class, how do you account for fields with spaces? ✓ Decorate with [IndexField("heading")] |
| Q What syntax do you use to query an index? ✓ LINQ | Q What method should you use to return a rich results object? ✓ .GetResults(); |

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

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>

Module 10

Recommended Practices

Contents:

- Working with media
- Caching and performance
- Publishing
- Installing and scaling Sitecore
- How to deal with deployment
- Team development and the development environment
- Basic security
- Workflow

Topic 10.1 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 real file ending

Content

Storing media

Media items are stored in the database

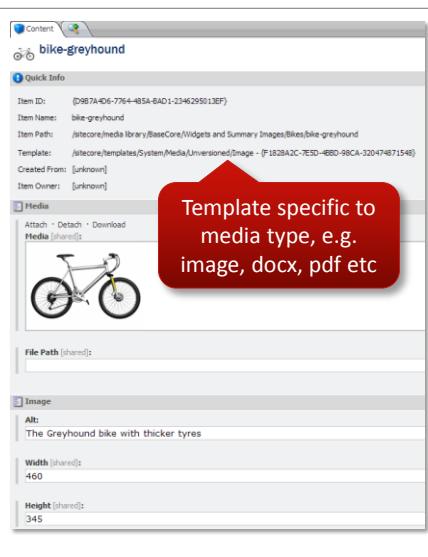
- Media items are **ordinary** items
- Located in the **content tree**
- Based on a **template** which is **extended**

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**



Tip

There is a file upload folder on in the website folder. This folder (if enabled in the web.config) will monitor media items being dropped in there. If files do get added, then they appear in the media library. Keep in mind when deploying your solution if you are storing media files on the file system, they need to be replicated to your new environment. It is recommended to 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 database files (MDF) remain lean. It can be inefficient and, depending on the size, might actually hurt performance

Assignment

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

*Note: When rendering media URLs, use EnsurePrefix for media items so the URLs start correctly with **/media** instead of **~/media***



Walkthrough – Changing file extensions

1. Go to **Traincore**
2. Inspect element for an image - **.ashx**
3. Change the **Media.RequestExtension** value
4. Inspect element again. The true extension is revealed

Review

Working with media

- | | |
|--|--|
| <p>Q How and where is media stored?</p> <p>✓ In the database as a regular item based on a template</p> <p>Q What template is the Pdf media type using?</p> <p>✓ PDF</p> <p>Q Are those templates configurable?</p> <p>✓ Yes</p> <p>Q In which file can you change what template Sitecore uses for that particular media item</p> <p>✓ Web.Config</p> | <p>Q Can you override the types and properties for the media types?</p> <p>✓ Yes</p> <p>Q What is the default Sitecore extension for media items?</p> <p>✓ .ASHX</p> <p>Q What property do you need to make a change to so that media is displayed with the real file ending?</p> <p>✓ Media.RequestExtension</p> |
|--|--|

Topic 10.2 Caching and performance

Introduction

Objectives

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

- List the five caching layers
- Name an action that clears HTML cache
- Discuss prefetch cache
- Discuss where you can clear all caches
- Describe where caching options are set
- State where cache settings are defined
- Explain the purpose of Profile and Debug mode in the Page Editor
- Describe the best practice for publishing

Content

Overview of caching

- When a request comes in, Sitecore checks for the item's presentation details in the **HTML cache**
 - If **item not found**, it **moves to the next cache layer** until it reaches the DB
-
- ```
graph LR; HTML[HTML] --> Item[Item]; Item --> Data[Data]; Data --> Prefetch[Prefetch]; Prefetch --> Database[Database]
```
- When **Sitecore finds a cached** version of the **item**, **request is returned** and it uses all the data to **regenerate** the **caching** level **above it**
  - **Caches are generated from each other**
  - There are more specialized caches for XSL, standard values, paths, registry etc.

## 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** 



Use 'clear all' to clear all Sitecore caches

| Actions                       |       |      |       |         |
|-------------------------------|-------|------|-------|---------|
| Totals                        |       |      |       |         |
| Entries: 9710, Size: 36880108 |       |      |       |         |
| Caches (104)                  |       |      |       |         |
| Name                          | Count | Size | Delta | MaxSize |
| AccessResultCache             | 232   | 12KB | 12KB  | 2MB     |
| admin[filtered items]         | 0     | 0    | 0     | 2MB     |
| admin[html]                   | 0     | 0    | 0     | 0       |
| admin[registry]               | 0     | 0    | 0     | 5MB     |
| admin[viewstate]              | 0     | 0    | 0     | 5MB     |
| admin[xsl]                    | 0     | 0    | 0     | 10MB    |
| sharedData                    | 1     | 50   | 50    | 50MB    |



### 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>
```

Figure 10-1 HtmlCacheClearer event handler in Web.config

## HTML Cache

- Stores **rendered HTML** for each component
- Defined on the definition item, standard values or on per instance of the component

### How to cache a component?

- Select the **component definition** item and search for **Caching** field section
- Vary by** allows you to store a **different version** of the component based on e.g. user, datasource, device etc.



Dynamically generated content in a component cannot be cached



You **cache components** which will only get cleared once a **publish** is done



### Tip

*HTML caching is defined on individual components. This is another reason to make sure you componentize your page*



### Explanation

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



### Walkthrough – Clearing cache

1. Set up HTML cache for a component
2. Navigate to <http://traincore/sitecore/admin/cache.aspx>
3. Notice that maximum cache sizes are listed in the column on the right side of the window
4. 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](#)



Prefetch



### Tip

*See **cache reference guide** for more information:*

<http://sdn.sitecore.net/Reference/Sitecore%206/Cache%20Configuration%20Reference.aspx>



### Tip

**Rocks caches interface** - using Rocks right click on an instance – in command type **cache**, 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

- 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
- Look at various cache configuration files in /App\_Config/Prefetch
- Look at the statistics page

## Profiling and debugging

**Debug mode – runs against the WEB database**

- View the **profile**, **cache settings** and **HTML output** of individual components in the **Page Editor**
- Trace follows **HttpRequestPipeline** steps detailing the journey

Contextual debug information

Profile table  
Individual component summary

Top 3 items taken the most time to render



### Tip

- CMS Performance Tuning Guide:  
<http://sdn.sitecore.net/Reference/Sitecore%207/CMS%20Performance%20Tuning%20Guide.aspx>
- CMS Diagnostics Guide:  
<http://sdn.sitecore.net/Reference/Sitecore%207/CMS%20Diagnostics%20Guide.aspx>



## Walkthrough – Use profiler and debugger

1. Log in to the **Sitecore Desktop** and start the site in debug mode by clicking **Sitecore > Debug**
2. Note that the ribbon at the top of the page allows you to enable/disable profiling and tracing:



**Figure 10-2 Page Editor Ribbon in Debug mode**

3. **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):

The screenshot shows the 'subl\_main\_2' component details. It includes the following statistics:

- Render Time:** 8.24ms
- Items Read:** 82
- Data Cache Misses:** 2
- Data Cache Hits:** 144
- Cache:** Not used

**Figure 10-3 Debug information per component**

4. The information about individual components is summarized in the profile table at the bottom of the page
5. 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, etc.)
6. 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

## Review

### Caching and performance

|                                                                     |                                                                                                               |
|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| <b>Q</b> Name the 5 cache layers                                    | <b>Q</b> Cache settings are defined in what file?                                                             |
| ✓ HTML, Item, Data, Prefetch, Database                              | ✓ web.config                                                                                                  |
| <b>Q</b> What operation clears the HTML cache?                      | <b>Q</b> Name 3 places where caching options can be defined                                                   |
| ✓ Publishing                                                        | ✓ Definition item, standard values or on per instance of the component                                        |
| <b>Q</b> Which cache is populated when the application starts?      | <b>Q</b> What 2 modes in Page Editor shows profile, cache settings and HTML output for individual components? |
| ✓ Prefetch cache                                                    | ✓ Profile and Debug mode                                                                                      |
| <b>Q</b> What is the path for the .aspx page that clears all cache? |                                                                                                               |
| ✓ /Sitecore/admin/cache.aspx                                        |                                                                                                               |

## Topic 10.3 Publishing

### Introduction

#### Objectives

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

- Name three elements that publishing restrictions apply to
- Discuss what live mode is
- Discuss limitations of live mode
- State the file name to use for live mode
- Describe the different publish modes

### Content

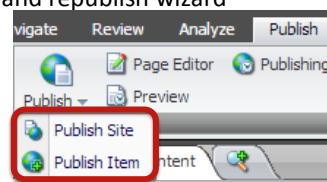
#### Types of publishing

##### 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

1. Numbered **version** of an item

2. The **item** itself

3. **Targets**

| Version | Publisable From     | Publisable To        | Publisable From | Publisable To |
|---------|---------------------|----------------------|-----------------|---------------|
| 1.      | 6/11/2014           |                      |                 |               |
| 2.      | 5/7/2014<br>6:00 AM | 5/14/2014<br>6:00 AM |                 |               |

**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



## Demo – Publishing options and restrictions

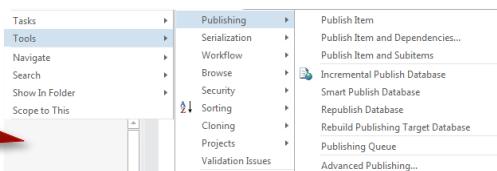
- View and set options and restrictions in Rocks and Content Editor

## 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-bi |
| about-us                | /sitecore/content/sitecore-cycling-holidays/Home/about-us |
| bikes-for-racing        | /sitecore/content/sitecore-cycling-holidays/Home/which-bi |
| can-i-bring-my-own-bike | /sitecore/content/sitecore-cycling-holidays/Home/which-bi |
| Home                    | /sitecore/content/sitecore-cycling-holidays/Home          |
| main                    | /sitecore/layout/Placeholder Settings/BaseCore/main       |

## Publishing API example

- To have your CRUD (Change Remove Update Delete) item to appear in the web database, you have to publish it and any dependencies that were changed

```
Item item; //item to publish
var publishingTargets = PublishManager.GetPublishingTargets(item.Database)
 .Select(i => Sitecore.Data.Database.GetDatabase(i[FieldIDs.PublishingTargetDatabase]))
 .ToArray(); //getting all the publishing targets, e.g. "web" database
PublishManager.PublishItem(
 item,
 publishingTargets,
 new Language[] { item.Language },
 deepBool, //deep=1 - publish children, deep=0 - do not publish children
 true); //set to true performs smart publish instead of a republish
PublishMode.Incremental or PublishMode.SingleItem
```

- Publish only single items!!!**

## Review

---

### Publishing

- |                                                                                                                                          |                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <b>Q</b> Name 3 publishing modes<br>✓ Incremental , smart and republish                                                                  | <b>Q</b> Which method do you use to publish through the API?<br>✓ .PublishItem |
| <b>Q</b> Which Sitecore interface do you use to set publishing restrictions?<br>✓ Content Editor                                         |                                                                                |
| <b>Q</b> Publishing restrictions can be applied to _____ , _____ and _____<br>✓ Numbered versions of items , the item itself and targets |                                                                                |

## Extend

---

- 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 10.4 Installing and scaling Sitecore

### Introduction

#### Objectives

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

- State two ways that Sitecore can be installed
- Discuss the installers functionality
- Name two tools that can be used to install Sitecore using scripts
- Know what three system components that get installed
- Name three folders that get installed
- Describe what the Data folder and the Website folder contain
- Discuss the Include folder
- 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

<http://sdn.sitecore.net/>

Home ▶ Downloads ▶  
Sitecore 7 ▶ Sitecore CMS + DMS

**Sitecore Installer recommended!**

- Sets correct file permissions
- Checks perquisites
- Sets correct DLLs 64-bit or 32-bit
- Removes installations (keeps Website folder)
- **Database / Client only**
- Logs the installation

ZIP archive of CMS site root is available too for manual installation





**Installation Guide** - Installation Guide details the standard Installer and zip installations  
**Installation Troubleshooting** - If you are having installation errors and problems



#### 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*

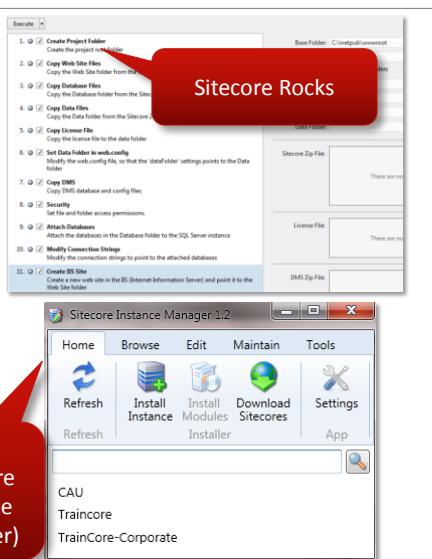


#### Important

When using the .exe for installation, if you do not add/remove programs, but just delete the files and databases, etc. you will leave entries in your registry

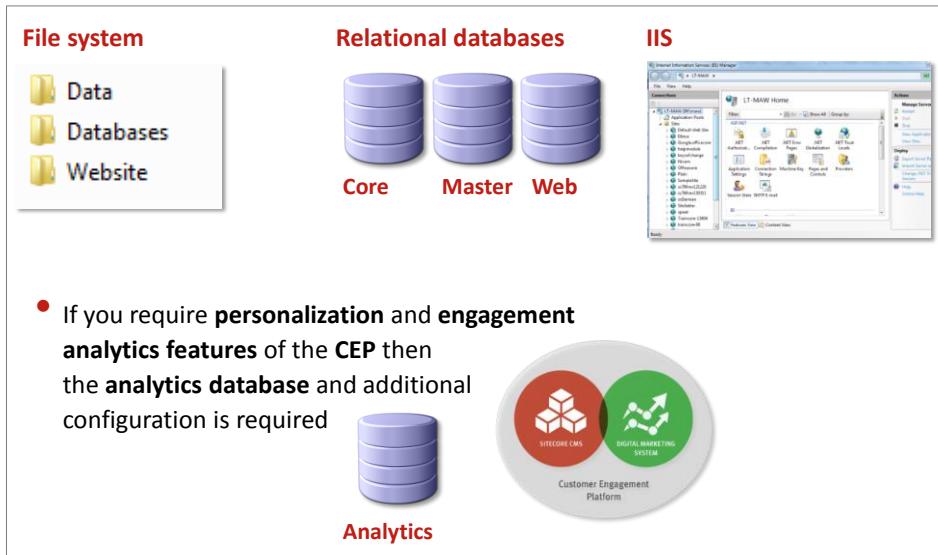
## Scripts (SIM or Sitecore Rocks)

- 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

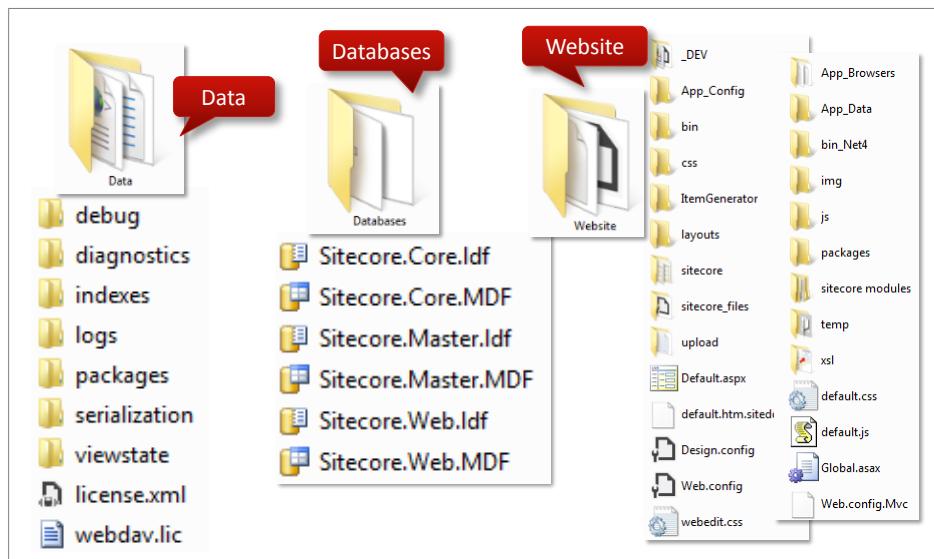


- **SIM download and documentation:**  
[http://marketplace.sitecore.net/en/Modules/Sitecore\\_Instance\\_Manager.aspx](http://marketplace.sitecore.net/en/Modules/Sitecore_Instance_Manager.aspx)
- **Sitecore Rocks:**  
[http://marketplace.sitecore.net/Modules/Sitecore\\_Rocks.aspx](http://marketplace.sitecore.net/Modules/Sitecore_Rocks.aspx)

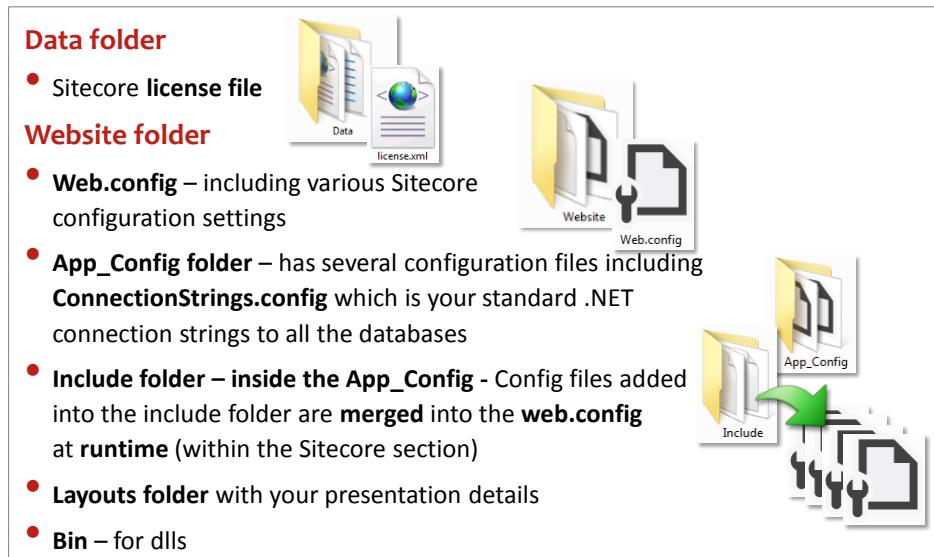
## System components that get installed



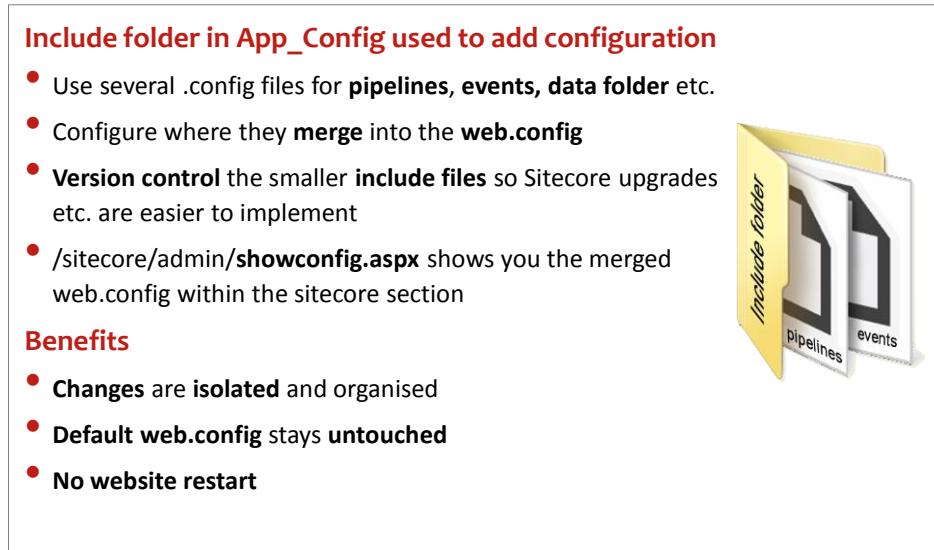
## Website root folders



## Key files and folders in Sitecore



## Config files and versioning



## Scaling infrastructure

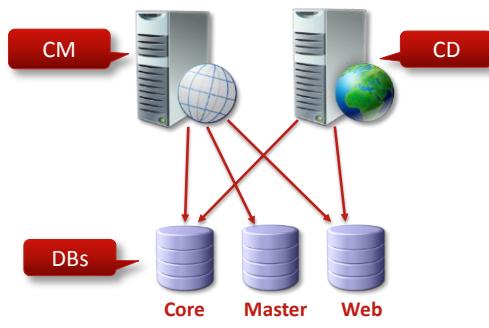
### Developer environment

- Single Sitecore client
- All 3 databases on the same machine

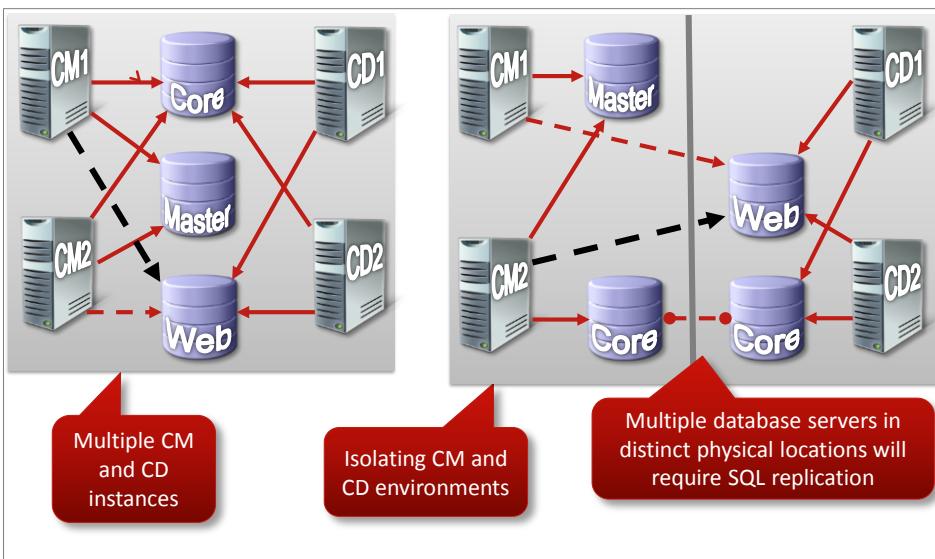


### Minimum recommendation for live/production environment

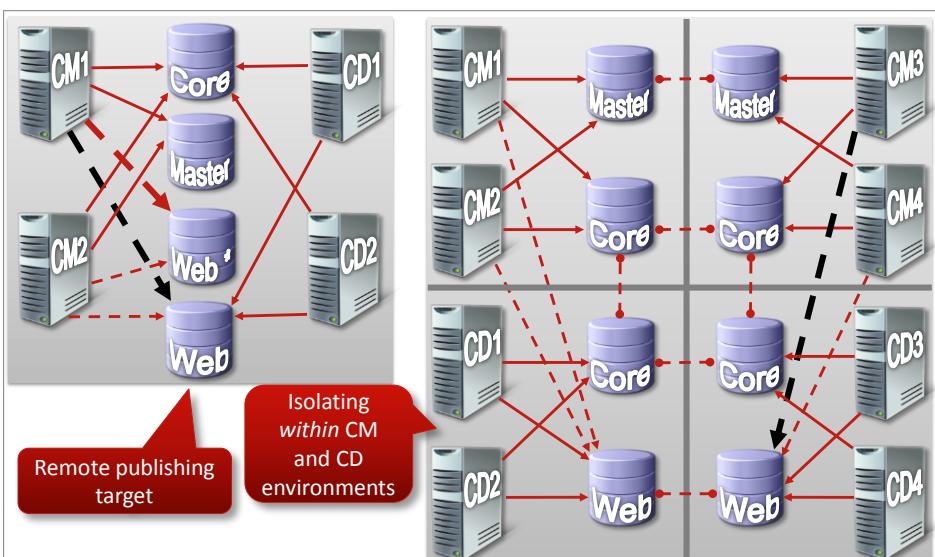
- Content management server has access to core, master, and web
- Content delivery server only has access to core and web
- No master on CD
- Databases are shared between instances
- Separate database server



## Complex scaling scenarios



## More complex scaling scenarios





## Important

Requirements depend 100% on the size of your solution. Depending on 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>

## Review

### Installing and Scaling Sitecore

|                                                                             |                                                         |
|-----------------------------------------------------------------------------|---------------------------------------------------------|
| <b>Q</b> List some benefits of using Sitecore Installer                     | <b>Q</b> Name 3 system components that get installed    |
| ✓ Checks prerequisites, correct DLLs, logs, remove an existing installation | ✓ File system, databases and IIS installed              |
| <b>Q</b> Can you do a manual installation?                                  | <b>Q</b> What should you include for CEP functionality? |
| ✓ Yes                                                                       | ✓ Analytics DB                                          |
| <b>Q</b> Name some features of Sitecore Instance Manager                    | <b>Q</b> In which folder is the license file found?     |
| ✓ Packages, customize, extend, reinstall & remove instances                 | ✓ Data folder                                           |
| <b>Q</b> Name some features of Rocks                                        | <b>Q</b> Where is the Include folder?                   |
| ✓ Packages, pick steps to include and remove instances                      | ✓ App_Config                                            |

|                                                                         |
|-------------------------------------------------------------------------|
| <b>Q</b> How can you view the merged web.config?                        |
| ✓ /sitecore/admin/showconfig.aspx                                       |
| <b>Q</b> What is the minimum recommendation for production environment? |
| ✓ CM with core, master and web                                          |
| ✓ CD with core and web                                                  |
| ✓ Separate DB server                                                    |
| ✓ Use scaling guide                                                     |

## Topic 10.5 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
- List two ways of adding items into packages using the Content Editor
- Name the aspx used to for upgrading Sitecore

### Content

#### Setting up a solution

| Development model                                                                                                   | Benefits                                                                                                                                                                                                                                                                                                                                                                                                                            | Drawbacks                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Inside</b> the web root:<br>  | <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>                                            |
| <b>Outside</b> the web root:<br> | <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> |



#### Walkthrough – Working outside the web root

1. In the file system, look at the *Traincore* solution and the *Traincore* website
2. Note that all developer-created files live in the solution
3. Note that the solution references a libraries folder of .dlls. These are not the .dlls in the web root, because the location and name might change depending on the developer's local environment
4. Re-iterate that deploying the solution moves files from the *Traincore* solution to the *Traincore* website – look at the publishing settings to confirm deploying
5. Show that any changes to the web.config are patched in using files in the *include* folder. In the case of Traincore, we have patched in events, pipelines, and site definition configuration

## 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

- Navigate to the serialization page – <http://traincore/sitecore/admin/serialization.aspx>
- Serialize one of the databases in its entirety
- Navigate to the **/Data/serialization** folder and note that a folder with serialized items is created as a result
- Log into the Content Editor and demonstrate that databases, individual items, or item trees can be serialized from the *Developer* tab:

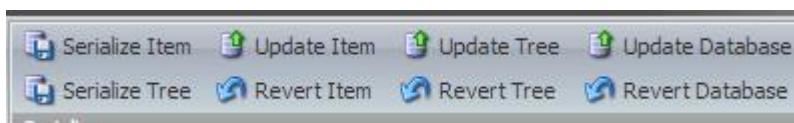


Figure 10-4 Content Editor Ribbon Developer tab

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

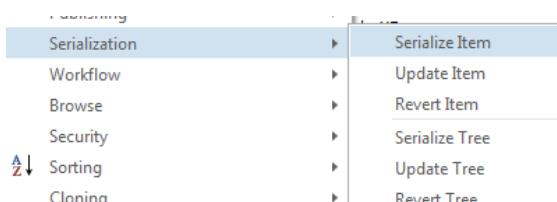


Figure 10-5 Rocks context menu for serializing



## Important

Serializing the database will not serialize users and roles. However, you can serialize them from the Role Manager and User Manager interfaces in the Sitecore Desktop

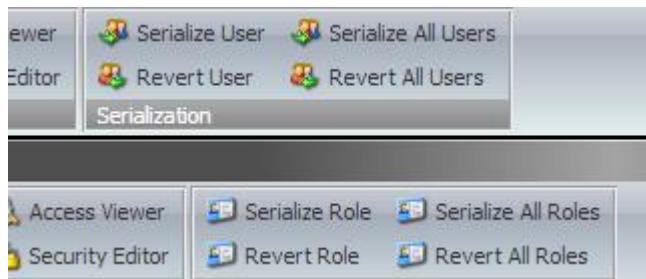


Figure 10-6 Serializing users in the Content Editor



## 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**



## Important

When packaging security accounts, all passwords are lost and have to be reset

## Update packages

**When you are upgrading Sitecore**

- Go to <http://sdn.Sitecore.net>
- **Download the update package**
- In your Sitecore instance go to /sitecore/admin/updateinstallationwizard.aspx
- **Remember:** do a database backup before upgrading

Welcome to Sitecore update installation wizard  
This wizard helps you to install a package.

Please remember to backup the databases and the web.config file.

[Choose a package >](#)

- **Follow the wizard**
- Go back to SDN for further configuration changes



### Demo – Create a Package Using 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 (e.g. 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

1. In Visual Studio, 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, choose 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 into the package
5. **Drag and drop** files from the **/Website** folder into the **package** and notice that they are added into the **File** tab
6. **Note:** you cannot drag and drop files from your solution, because solution files are independent of the Sitecore instance

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

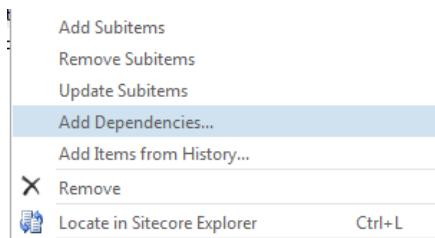


Figure 10-7 Add Dependencies in Rocks for a package

8. You are also given the option to **add dependencies**. (e.g., If you add an item, you also want to add its template, associated renderings, and potentially any other items it references)
9. **Build** a package, and **open windows explorer** when prompted, notice that a **.zip** file is created in the same location as your solution, however Sitecore stores packages in the **Data folder** of the Sitecore installation
10. **Show how you would go about installing it, mention anti packages – which is done in the next lab**

## Apply – Topic 10.5 – 5 min



### Install a package

#### Lab A. Install a package using Rocks

Using Sitecore Rocks, install the package you just created in the walkthrough:

1. To install a package, right-click on 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 (**Data folder**, in your Sitecore installation)

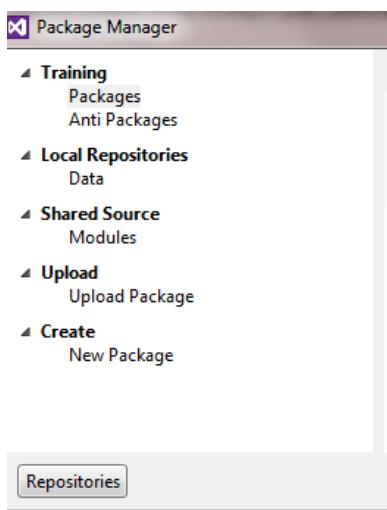


Figure 10-8 Package repository

3. Notice that you can generate an anti-package that functions as an uninstaller

## Review

### Team development and development environment

- |                                                                                                                               |                                                                                                                                              |
|-------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Q</b> Name 2 development models</p> <p>✓ Inside and outside the web root</p>                                            | <p><b>Q</b> What are Sitecore packages and what do they contain?</p> <p>✓ zip files that contain items and code files</p>                    |
| <p><b>Q</b> Which is the best practice?</p> <p>✓ Development outside the web root</p>                                         | <p><b>Q</b> What is the name of the .aspx that you go to when upgrading Sitecore?</p> <p>✓ /sitecore/admin/updateinstallationwizard.aspx</p> |
| <p><b>Q</b> What are the options to source control your items?</p> <p>✓ You can serialize your items and files or use TDS</p> |                                                                                                                                              |

## Extend

---

- Sitecore NuGet: <http://vsplugins.sitecore.net/Sitecore-NuGet.ashx>

## Topic 10.6 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

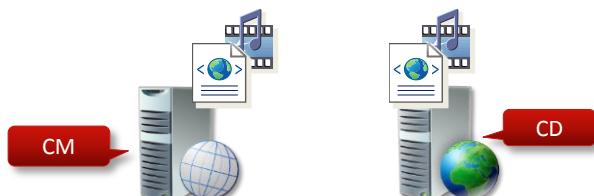
### Content

#### Difference between publishing and deployment

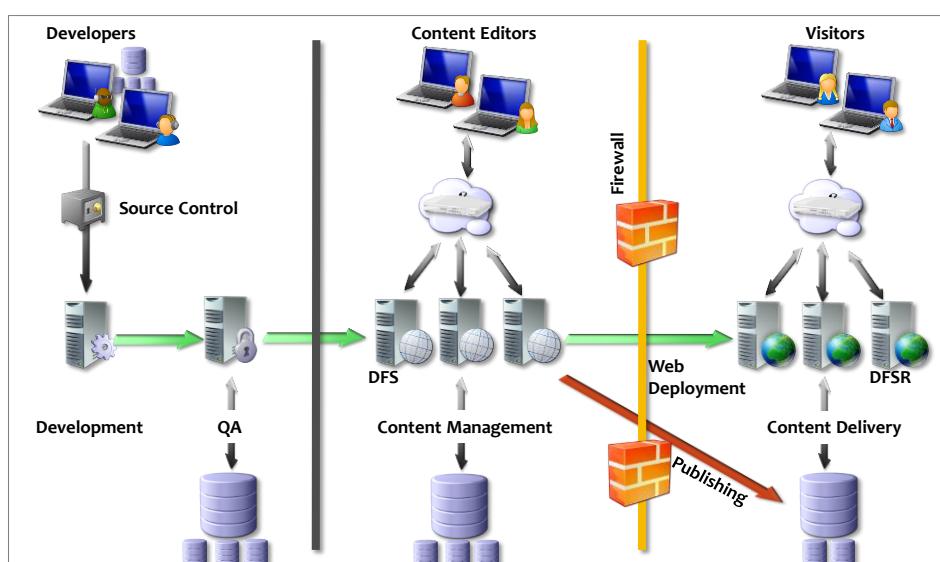
- Sitecore **publish** means **items** have been synced between various **databases**



- **Deployment** means **files** and potentially content syncing between various environments



#### Deployment between environments



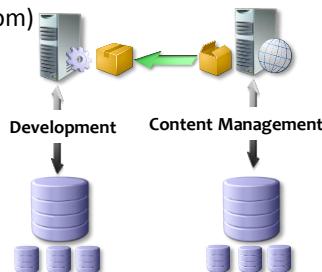
## Points to remember

### 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
  1. Database backups
  2. Packages



## Review

### How to deal with deployment

**Q** What is the difference between Sitecore publish and deployment

✓ Publishing synchronises items between various databases, deployment synchronises environments

**Q** How would you update your developer environment with the authored environment?

✓ DB backups / packages

**Q** Name some points to remember when deploying to the authors environment

✓ Content freeze, DB backups, be careful what items you override, optionally deploy to a temp location

## Topic 10.7 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

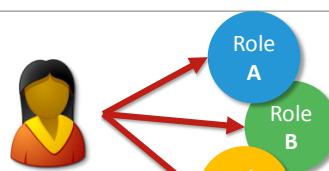
- **Security Editor** and **Access Viewer** - accessible via the Sitecore Desktop interface
- **Security Editor** used to define access rights
- **Access Viewer** used to view the resulting access right
- Allows you to define access rights **per user or per role**
- Used for **authors** and **website users**
- Permissions: **Allow / deny** or **leave unspecified** each access right



### 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**

#### Write access to an item



#### Client

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



## Demo – Security

- Restrict access to a part of the tree
- Set up the access right on a role
- Override on a user



## Best Practices

Define a role per task (e.g., news editor, 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

## Review

---

### Security

- |                                                                                              |                                                                                                             |
|----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Q</b> Which application is used to define access rights<br><br>✓ Security Editor          | <b>Q</b> Can access be explicitly given to a user to override the role access right?<br><br>✓ Yes           |
| <b>Q</b> Which application is used to view the resolved access rights<br><br>✓ Access Viewer | <b>Q</b> Name 3 permissions that are applied to access rights<br><br>✓ Allow, deny and not specified = deny |
| <b>Q</b> You can assign access rights to _____ and _____<br><br>✓ Users and roles            |                                                                                                             |

## 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 10.8 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 some sensitive material that needs to be approved by different groups in the organization before it can go live. Workflows allow you to set one source of responsibility for that*



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

- Push an item through the workflow
- Talk about versioning, tracking
- It's a good idea to have cleanup 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:

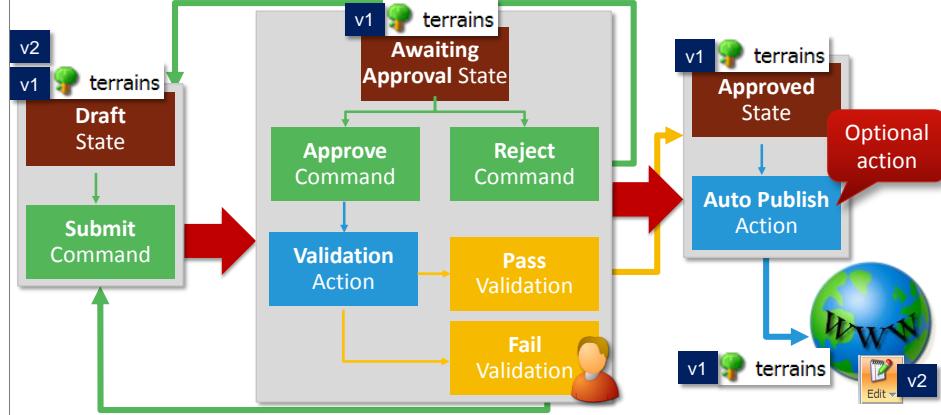
- **States** that an item moves through
- Each state can have **commands** and **actions**
- Items are **publishable** in the **final state**

The screenshot shows the Sitecore Workflows interface. At the top, there is a list of workflows: 'BaseCore Basic Workflow' and 'BaseCore Holidays Workflow'. A red box labeled 'Workflow name' points to the 'BaseCore Holidays Workflow' entry. Below this is a detailed view of the 'visit-the-philippines' item. The item is in the 'Draft' state, which is highlighted with a red box labeled 'State name'. The item has a version history entry: 'Last change: admin changed from ? to Draft on Monday, June 17, 2013. More'. Below the item details is a toolbar with buttons for 'Open', 'Diff', and 'Submit'. A red box labeled 'Item name' points to the 'visit-the-philippines' link. At the bottom of the view, there is a status bar: 'Awaiting Approval - (none)'. A red box labeled 'Workflow commands' points to the 'Submit (selected)' and 'Submit (all)' buttons.

#### Can be used for content versioning

## Workflow

- Series of **states** that items must go through before being automatically published
- Items can move between states with either using **commands** or **actions**
- When published and edited again, a **new numbered version** gets created



### It is recommended to

- 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, old numbered versions



Figure 0-9 Clean up numbered versions

## Review

### Workflows

- |                                                         |                                                  |
|---------------------------------------------------------|--------------------------------------------------|
| <b>Q</b> What are workflows used for?                   | <b>Q</b> When is an item publishable?            |
| ✓ Content approval, versioning and tracking             | ✓ In the final state                             |
| <b>Q</b> Items go through a series of _____             | <b>Q</b> When is a new numbered version created? |
| ✓ States                                                | ✓ When the published item is edited again        |
| <b>Q</b> Each state can contain certain _____ and _____ |                                                  |
| ✓ Commands and actions                                  |                                                  |

# Module 11

## Optional Topics

### Contents:

- Branches
- Pipelines and events
- Rules

## Topic 11.1 Branch Templates

### Introduction

#### Objectives

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

- Describe a solution to the problem of having to repeat the process of having to create a certain tree structure

### 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 see how to:

- Create branch templates in the Template Manager
- How to assign this new branch template as insert options for the Holidays data template

- To create a new branch template, navigate to [/sitecore/templates/Branches/User Defined](/sitecore/templates/Branches/User%20Defined) (ensure that **Hidden items** is ticked in the View tab) and select **New Branch**
- Create this new branch to start from the **Holiday** data template and click **Create**
- Now if you click the **\$name** that got created and Click on the Home tab to 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**



Figure 11-1 Create branch template

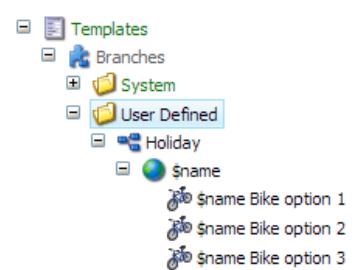


Figure 11-2 Branched tree



## Knowledge Check

What will happen to the \$name token when an item is created based on the Holiday Branch template?

4. Assign this Holiday branch template as an insert option for the Holiday Listing
5. Let's create an instance of this branch template, navigate to /sitecore/content/home/Holidays item and from the insert options select Holiday
6. Name it
7. A new sub-tree has been created and all \$name tokens have been replaced with Experience Budapest:



Figure 11-3 Insert an instance of Holiday branch template

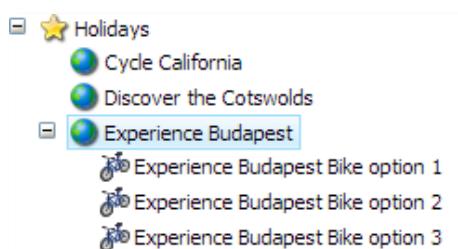


Figure 11-4 Experience Budapest item



### Note

Branch templates can be assigned as insert options on standard values in exactly the same way as regular templates

## Review

### 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 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 3 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**

#### There are two types of pipelines



- <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.IgnoreList, 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" />
<processor type="Sitecore.Pipelines.HttpRequest.DeviceResolver, Sitecore.Kernel" />
```

#### 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
 }
 }
}
```

## Creating pipelines option 1

There are **3 options** for using pipelines to customize Sitecore behaviour

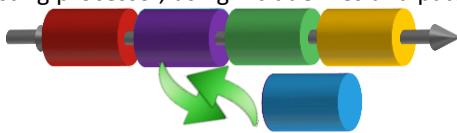
### Option 1: Add processors to existing pipelines

1. Locate the processor that you want to extend – e.g. `insertRenderings`
2. 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`
3. Use **include files** to patch your processor into the pipeline
4. 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
5. Decide what args the `Process()` accepts – in this case `InsertRenderingsArgs`
6. 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:

1. Inheriting from that processor giving you access to the methods in the original `AddRenderings`

```
class MyCustomAddRenderings : Sitecore.Pipelines.InsertRenderings.Processors.AddRenderings
```

2. Writing a custom processor can involve copying all or some of processor's code

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. 
  - 2. Modified 5/8/2013 4:30
  - 1. Modified 5/1/2013 2:00
- 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**

Event handler  
that is executed  
when an item is  
saved

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

- 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

Using events <http://sdn.sitecore.net/Articles/API/Using%20Events.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



## Review

### Pipelines and events

**Q** What are pipelines

- ✓ Break down operations into multiple configurable steps

**Q** Where are pipelines defined?

- ✓ In the web.config as <processors /> and <pipelines />

**Q** In what folder on the file system do you store your pipeline configuration files?

- ✓ The include folder

**Q** What method do all processors implement?

- ✓ Process()

**Q** What are the 3 options for creating pipelines

- ✓ Add processor to existing pipeline
- ✓ Override and extend existing processors
- ✓ Create new pipelines that are called from code

**Q** Give an example of an event that you can subscribe to

- ✓ Item saved / created / updated / deleted / moved / renamed / etc.

**Q** How do you subscribe to events?

- ✓ Using event handlers

**Q** Where are events defined?

- ✓ In the web.config

## Topic 11.3 Rules

### Introduction

#### Objectives

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

- Define a rule
- Name the 2 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
- In Module 8, you used **conditional renderings**, which utilized the rules engine for personalization - If a user has 5 points, change the datasource of a component
- Rules are part of the core Sitecore functionality and can be used everywhere e.g. **conditional insert options** (if a user is \_\_\_, show \_\_\_\_)

**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

#### 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: **Items**  
where the page index compares to number  
where the item ID compares to value  
where the item name compares to value  
where the item template is specific template  
where the item language compares to value  
where the item has a layout  
where the parent name compares to value  
where the parent template is specific template  
where the item is publishable now

Select the actions for the rule: **Actions**  
hide rendering  
process multivariate tests  
set data source to Item  
set parameters to value  
set placeholder to value  
set rendering to Item  
run specific script



## Walkthrough: Implementation of insert options rule

Navigate to **/sitecore/system/settings/rules/insert options/rules**, all the rules here are run automatically all the time, what we want to do is create a new rule that says If the item name is **Links**, the insert options for that item will be a template called **Simple Link**

1. Create a new Insert Option Rule called **Links**
2. In the Name field enter **Links**
3. In the Rule field enter:
  - 4. where the item name is equal to Links  
add Simple Link insert option  
(/BaseCore/Global Content/Simple Link)
5. Under **/sitecore/content/sitecore-cycling-holidays/Global** create a Links item using the common Sitecore folder
6. Notice the insert option includes Simple Link
7. There are many other conditions and actions that are useful regardless of what 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 one we just used and notice how they are defined

### Rule 1

where the item name is equal to Links

add Simple Link insert option

**Figure 11-5 If the item name is Links, set the insert options to be a template called Simple Link**



### 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 e.g. 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 – there is a module that does this in the Sitecore Market Place*
- *Hide Insert options based on what level the tree is in*
- *You may wish to boost items when indexing data, for example items based on their depth in the information architecture relative to another item - give the children of the home item the greatest boost, their children (grandchildren of the home item) significant boost, and their nested descendants greater boost than default, but less than the items closer to the home item - <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>*

## Review

### Rules

**Q** A rule consists of a \_\_\_\_\_ and \_\_\_\_\_

✓ Condition and action

**Q** What should the final result be for a condition for the action to be executed?

✓ True

**Q** In what other scenarios can rules be used, besides personalization?

✓ E.g. Insert Options

### Extend

- Rules engine cookbook  
<http://sdn.sitecore.net/Reference/Sitecore%206/Rules%20Engine%20Cookbook.aspx>

# Module 12

## Support Community

### Contents:

- Sitecore Support
- Sitecore Community
- Modules

## Topic 12.1 Sitecore Community

### Introduction

#### Objectives

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

- Know what Sitecore Developer Network
- Access Sitecore Marketplace
- Know how to access Sitecore MVPs

### Content

#### Sitecore Developer Network

- <http://sdn.sitecore.net/>

Products and downloads

Documentation

Forum

Support best practice

A screenshot of the Sitecore Developer Network (SDN) website. The header features the Sitecore logo and navigation links for Products, Documentation, Downloads, Support, Scrapbook, MVP, and Forum. Below the header, a banner reads "Welcome to SDN, the developer network for Sitecore!". The main content area is divided into several sections: "Products" (links to release documents and Sitecore products), "Documentation" (links to programmatic reference and developer classes), "Downloads" (links to files available for download), "Missed Sitecore Symposium 2012?" (a recap of the event with a link to presentation materials), "Support" (links to support forums and resources), "Scrapbook" (links to quick tips and other documents from site support staff), "MVP" (links to information about the Sitecore Most Valuable Professionals program), and "Forum" (links to discussions in Sitecore related forums). At the bottom, there are footer links for Legal notice, Copyright © 1998 - 2013 Sitecore AG, and Privacy policy, along with social media icons for Facebook and LinkedIn.

#### Sitecore Marketplace

- <http://marketplace.sitecore.net/>

Official modules

User-contributed modules

A screenshot of the Sitecore Marketplace website. The header features the Sitecore logo and navigation links for DISCOVER and CONTRIBUTE. The main banner is red with the text "Discover Sitecore modules" and "SITECORE MARKETPLACE IS HOME TO OVER 100 SITECORE MODULES. GET A GLIMPSE BY EXPLORING MODULES RIGHT HERE...". Below the banner is a search bar with a magnifying glass icon. A call-to-action button says "SIGN UP NOW" with "IT'S FREE" underneath. Below the button, it says "OR SIGN IN WITH EXISTING ACCOUNT:" followed by Facebook and LinkedIn icons. At the bottom, there is a footer link for "See what you get!".

## Sitecore MVPs

- <http://sdn.sitecore.net/MVP.aspx>

Technical community leaders

Technical previews and pre-releases

The screenshot shows the Sitecore MVP page on the Sitecore Developer Network. At the top, there's a navigation bar with links for Home, Products, Documentation, Downloads, Support, Scrapbook, MVP, and Forum. A user is logged in as 'michaelschaefer'. Below the navigation, there's a section titled 'Sitecore MVP Award' which describes the award given to exceptional technical community leaders. It also features sections for 'MVP list', 'Sitecore MVP Award', and 'Benefits'. At the bottom, there are links for Legal Notice, Copyright, and Privacy Policy.

- John West Blog

<http://www.sitecore.net/Community/Technical-Blogs/John-West-Sitecore-Blog.aspx>

## Sitecore Learning

- [Sitecore Learning Management System](#)
- <http://sitecorelearning.net/login/index.php>

The screenshot shows the Sitecore Learning login page. The header says 'sitecore LEARNING'. On the left, there's a 'Returning to this web site?' form with fields for Username and Password, and checkboxes for 'Remember username' and 'Forgot your username or password?'. On the right, there's a 'Welcome to Sitecore Learning' section with a smiley face icon, instructions for logging in, and a link for 'Create new account'. The footer includes copyright information and a note about not being logged in.

## Review

---

### Sitecore Community

- Q** What is the URL for SDN?
  - ✓ <http://sdn.Sitecore.net>
- Q** What is the name of the official place to get modules from?
  - ✓ Marketplace
- Q** What perks do you get being a MVP?
  - ✓ Technical previews and pre-releases

## Topic 12.2 Sitecore Support

### Introduction

---

#### Objectives

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

- State the URL of Sitecore support portal
- Understand best practices for contacting support

### Content

---

#### Best practice for contacting support

<http://support.sitecore.net/> (SDN login details)

- One problem per issue
- Timestamp your responses
- Attach relevant configuration files
- Attach recent log files
- Attach exceptions as plain text documents
- Attach a screenshot of any UI errors
- Attach any files relating to the issue
- Helpdesk best practices -  
<http://sdn.sitecore.net/Support/Helpdesk%20Best%20Practices.aspx>

Public FTP for files larger than 10MB

<http://sdn.sitecore.net/Support/Sitecore%20Public%20FTP.aspx>

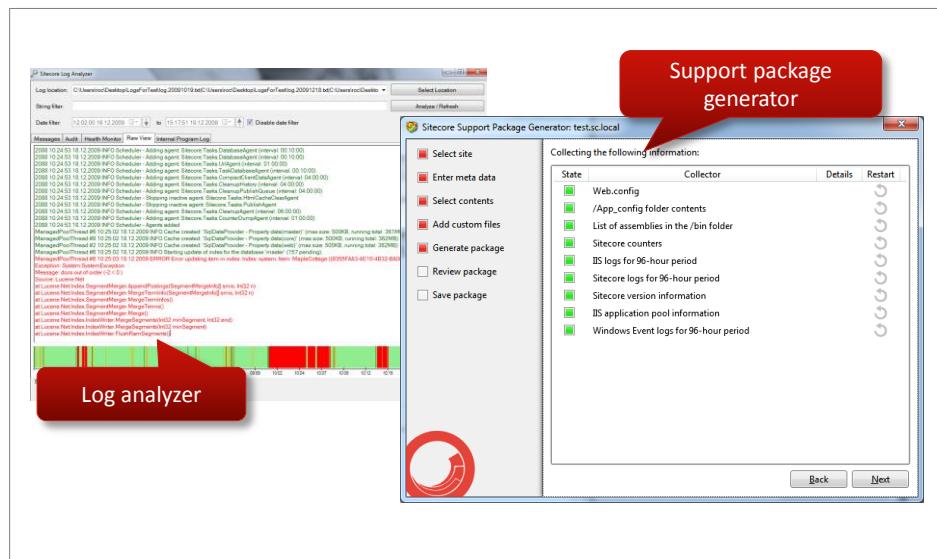
#### Best practice for contacting support cont

- Try reproducing the issue on a clean installation before contacting support
- Look at log files using log analyzer

[http://marketplace.sitecore.net/en/Modules/Sitecore\\_Log\\_Analyzer.aspx](http://marketplace.sitecore.net/en/Modules/Sitecore_Log_Analyzer.aspx)

- Search marketplace before asking for code examples
- ShowConfig.aspx , compare it with stock one

## Tools



## Review

### Sitecore Support

**Q** What is the URL for the support portal?

✓ <http://support.Sitecore.net>

**Q** Name some best practices

- ✓ 1 problem per issue
- ✓ Timestamp responses
- ✓ Attach configuration, log files, screenshots
- ✓ Reproduce issue on clean installation
- ✓ Attach any files relating to the issue

## Topic 12.3 Modules

### Introduction

---

#### Objectives

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

- Name 2 modules

### Content

---

#### Modules

<http://sdn.sitecore.net/Resources/Sitecore%206.aspx>

- Email Campaign Manager
- Web Forms for Marketers
- Social Connected
- Sitecore Item Web API
- App Center

### Review

---

#### Modules

**Q** Name any 2 modules?

- ✓ ECM / WFFM / Social / Sitecore  
Item Web API