The Sitecore Developer’s Cookbook

Task-based Solutions for Developing with Sitecore 5.3

Author: Kerry Bellerose, Sitecore Corporation
Date: February 8, 2007
Release: Revision 1.1.0 (Sitecore V5.3.0)
Language: English

Sitecore® is a registered trademark. All other brand and product names are the property of their respective holders.

The contents of this document are the property of Sitecore.
Copyright © 2001-2007 Sitecore. All rights reserved.
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Introduction</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Who Should Read this Book</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>How to Use this Book</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>A Brief Introduction to Sitecore</td>
<td>4</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Understanding Sitecore Content</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Understanding Sitecore Presentation Objects</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2</th>
<th>Visual Studio 2005</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Configuring a Visual Studio 2005 Web Application Project</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Prerequisites</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Create an ASP.NET Web Application Project</td>
<td>10</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Physically Move the Project Files</td>
<td>11</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Open the Project File and Complete the Configuration</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Editing a Layout, Sublayout, or XSLT Rendering</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>Creating Code-Behind for Layouts and/or Sublayouts</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>Templates</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Creating a Template</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Setting a Template’s Base Template</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Creating a Field Section</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Creating a Field</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Editing Advanced Field Attributes</td>
<td>19</td>
</tr>
<tr>
<td>3.6</td>
<td>Changing the Order of Fields in the Content Editor</td>
<td>19</td>
</tr>
<tr>
<td>3.7</td>
<td>Setting the Icon Associated with a Template or Field Section</td>
<td>20</td>
</tr>
<tr>
<td>3.8</td>
<td>Locating Useful Icons</td>
<td>21</td>
</tr>
<tr>
<td>3.9</td>
<td>Setting Standard Values on a Template</td>
<td>22</td>
</tr>
<tr>
<td>3.10</td>
<td>Assigning Masters for all Items based on a Template</td>
<td>22</td>
</tr>
<tr>
<td>3.11</td>
<td>Assigning Layouts, Sublayouts, and Renderings</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Masters</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Creating a Master</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Editing an Existing Master</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Creating a Master Hierarchy</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>The Content Tree</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Assigning Masters for a Single Item</td>
<td>28</td>
</tr>
<tr>
<td>5.2</td>
<td>Assigning Layouts, Sublayouts, and Renderings to a Single Item</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Media</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Upload Multiple Images via a ZIP Archive</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 7</th>
<th>The Developer Center</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Creating a Layout, Sublayout, or XSLT Rendering</td>
<td>34</td>
</tr>
<tr>
<td>7.2</td>
<td>Editing an Existing Layout, Sublayout, or XSLT Rendering</td>
<td>35</td>
</tr>
<tr>
<td>7.3</td>
<td>Adding a Placeholder to a Layout or Sublayout</td>
<td>35</td>
</tr>
<tr>
<td>7.4</td>
<td>Statically Placing Sublayouts and XSLT Renderings</td>
<td>36</td>
</tr>
<tr>
<td>7.5</td>
<td>Creating a Table</td>
<td>37</td>
</tr>
<tr>
<td>7.6</td>
<td>Editing a Table</td>
<td>37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 8</th>
<th>Security and User Management</th>
<th>38</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Creating a User</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 9</th>
<th>Workflows</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Creating a Workflow</td>
<td>40</td>
</tr>
<tr>
<td>9.2</td>
<td>Creating a Workflow State</td>
<td>40</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>General Tips and Tricks</td>
<td>Page</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>------</td>
</tr>
<tr>
<td>9.3</td>
<td>Creating a Workflow Command</td>
<td>40</td>
</tr>
<tr>
<td>9.4</td>
<td>Creating a Workflow Action</td>
<td>41</td>
</tr>
<tr>
<td>9.5</td>
<td>Provide Write and Delete Access to Items in a Workflow</td>
<td>41</td>
</tr>
<tr>
<td>9.6</td>
<td>Hide a Workflow State in the Workbox for Certain Users</td>
<td>42</td>
</tr>
<tr>
<td>9.7</td>
<td>Hide a Workflow Command for Certain Users</td>
<td>42</td>
</tr>
<tr>
<td>9.8</td>
<td>View Workflow Comments</td>
<td>43</td>
</tr>
<tr>
<td>Chapter 10</td>
<td>General Tips and Tricks</td>
<td>44</td>
</tr>
<tr>
<td>10.1</td>
<td>The Sitecore Internet Explorer Toolbar</td>
<td>44</td>
</tr>
<tr>
<td>10.1.1</td>
<td>Installation</td>
<td>44</td>
</tr>
<tr>
<td>10.1.2</td>
<td>Configuration</td>
<td>46</td>
</tr>
<tr>
<td>10.2</td>
<td>Generating “Lorem Ipsum” Filler Data</td>
<td>47</td>
</tr>
<tr>
<td>10.3</td>
<td>Pointing to an Alternative Home Item</td>
<td>47</td>
</tr>
<tr>
<td>10.4</td>
<td>Using a Custom CSS File</td>
<td>47</td>
</tr>
<tr>
<td>10.5</td>
<td>Creating a Device</td>
<td>48</td>
</tr>
<tr>
<td>10.6</td>
<td>Creating a Device Specific Site</td>
<td>48</td>
</tr>
<tr>
<td>10.7</td>
<td>Installing a Package</td>
<td>49</td>
</tr>
<tr>
<td>10.8</td>
<td>Adding a Language</td>
<td>51</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Sitecore provides a very rich set of powerful and easy to use developer tools for creating web sites. Nonetheless, the sheer breadth of the tool set can sometimes feel a bit overwhelming for developers who are just getting started and want to perform simple tasks. Identifying with these developers provided the motivation for this book, which is a standard part of the training materials given to all developers attending Sitecore’s Certified Developer training courses. Rather than attempt to describe all aspects of Sitecore, this book provides simple, step-by-step instructions for completing common tasks.

1.1 Who Should Read this Book

This book provides a useful reference for developers working with Sitecore 5.3.

The descriptions focus on how to perform common tasks of varying complexity and are therefore particularly useful both for developers with no previous knowledge of Sitecore as well as Sitecore veterans who are just beginning to use Sitecore 5.3.

On the other hand, the text provides a minimalist approach to describing how to complete each task. Rather than attempting to describe Sitecore concepts such as the difference between a Template and a Layout, it assumes that the reader will have access to such information from other sources (such as an instructor during training or the Sitecore Developer Network website).

1.2 How to Use this Book

Readers should begin each rendezvous with this book at the Table of Contents, which provides a menu of the various tasks described. The book is organized into various chapters which collect tasks associated with a specific topic, such as Templates or the Developer Center.

Descriptions assume that the reader knows how to perform tasks related to the topic being discussed. For instance, if a task requires that the developer work with an existing object, the description will simply direct the reader to “open an existing object.” Another topic provides instructions for readers who do not know how to create such an object.

1.3 A Brief Introduction to Sitecore

Sitecore is a Content Management System and a website development environment.

Sitecore is built on top of the Microsoft .NET foundation and supports everything that the Microsoft .NET platform supports. Sitecore, in fact, is a wrapper around the .NET platform. Sitecore simplifies the process of creating .NET Web Applications. Sitecore also leverages many
standard technologies, such as XML, XSLT, XPATH and other technologies from W3C and others.

Sitecore offers user interfaces designed for each of the many different types of users involved in the process of creating and maintaining a web site. Developers use Sitecore user interfaces (together with Microsoft’s Visual Studio) to implement the infrastructure objects required to store content and present it as the web pages that make up a site. Administrators use Sitecore to manage users that have access to the Sitecore user interfaces and assign them access rights to various areas of the content and Sitecore functionality. Content Authors use Sitecore to provide the content shown on the web site.

Developers do well to remember content authors, because the choices a developer makes when defining the content infrastructure will influence how content authors experience Sitecore for a particular website. Developers have full control over the editing controls and other user interface constructs a content author will use when providing content for the web site. Good decisions made early in the process will help to ensure a successful project.

The general idea of Sitecore is very simple.

Sitecore takes content stored in a data source and dynamically transforms it into web pages.

The key to learning how to use Sitecore is understanding how Sitecore stores content and how developers define the objects, referred to as “presentation objects” that transform content into web pages. We cover content first, then return to presentation objects afterwards.

1.3.1 Understanding Sitecore Content

Sitecore separates content from presentation. This has many advantages. Separating content from presentation means that content authors can focus on content without having to worry about how it will be presented on the web site. Authors do not need to concern themselves with keeping presentation consistent. They do not need to enter content multiple times just because it appears on multiple pages. Developers who have worked with normalized relational databases are familiar with this idea.

When working with relational databases, developers create table schemas that define a number of columns, each with a specific data type. The result is a table that eventually contains a number of records which all share the same columns. A database usually contains multiple tables with specific columns that provide relationships between tables.

In Sitecore, developers create templates which are similar to relational database table schemas. A template defines the fields associated with a set of items. An item is like a relational database

<table>
<thead>
<tr>
<th>Schema</th>
<th>Relational Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>char(12)</td>
</tr>
<tr>
<td>Name</td>
<td>char(100)</td>
</tr>
<tr>
<td>Features</td>
<td>char(255)</td>
</tr>
<tr>
<td>Picture</td>
<td>blob</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Product Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>100</td>
<td>Blue Blouse</td>
</tr>
<tr>
<td>101</td>
<td>Water Bottle</td>
</tr>
<tr>
<td>102</td>
<td>HD TV</td>
</tr>
</tbody>
</table>
record and a field is like a relational database column. In Sitecore, an item also roughly corresponds to an individual web page that a visitor can request.

There are a few differences, however, between Sitecore storage and relational databases. For instance, rather than storing items in tables, Sitecore organizes items into a hierarchy called the content tree. Items defined by different templates can be interspersed throughout the content tree. Also, although fields have an associated type, in Sitecore this indicates the editing control used to edit the field. For example, a text field provides a simple text box that accepts a single line of text, whereas a rich text field displays a powerful word processor and an image field provides access to the Sitecore media library.

The content tree is similar to the Windows file system. The main difference is that the Windows file system tree is made up of folders which may contain other folders and/or documents containing data; whereas the Sitecore content tree is made up of items which contain data but can also act like folders and have sub-items.

Sitecore items can relate to one another in the same way that records in relational databases relate to one another. That is, an item can hold a field which contains the ID of another item, just like a record can contain a column that contains a foreign key pointing to a record in another table. Each Sitecore item, however, also keep track of its parent and children.

<table>
<thead>
<tr>
<th>RDB Term</th>
<th>Comments</th>
<th>Sitecore Term</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>A collection of tables containing records defined by various table schemas.</td>
<td>Content Tree</td>
<td>A collection of items associated with various templates.</td>
</tr>
<tr>
<td>Table Schema</td>
<td>Defines the columns associated with all records in a table.</td>
<td>Template</td>
<td>Defines the fields associated a set of items.</td>
</tr>
<tr>
<td>Column Definition</td>
<td>Defines a column and the data type allowed in that column.</td>
<td>Field Definition</td>
<td>Defines a field and the editing control used to edit content in that field.</td>
</tr>
<tr>
<td>Record</td>
<td>A single row in the table with values for all columns.</td>
<td>Item</td>
<td>An element in the content tree with values for all fields defined in an associated template.</td>
</tr>
</tbody>
</table>
The last major Sitecore content related concept that we have not touched on yet is masters. Developers define masters as a mechanism authors use to create new items. In some ways, masters are similar to relational database on-insert triggers, because developers can define masters to set initial values for specific fields in an item. On the other hand, masters also help developers to ensure that content authors only create items based on appropriate templates at various points in the content tree.

1.3.2 Understanding Sitecore Presentation Objects

As mentioned above, Sitecore separates content from presentation. This saves work for both content authors and for developers. Separating content from presentation means that developers can define very few presentation objects to create an astonishingly rich visual appearance on a web site.

Developers need only learn about a few types of presentation objects. These include:

- **Layouts**
  
  A Sitecore Layout defines the overall appearance of a generated web page. Often a single layout can define the overall appearance for nearly all pages on a website for a particular form factor. For instance, one layout may define the appearance of the site for a typical web browser such as Internet Explorer or Firefox, while a second layout may define the appearance of pages when printed, and a third layout may define the appearance of pages when displayed on a PDA.

  Sitecore Layouts are based on .NET Web Forms (aspx files) and support all functionality associated with .NET Web Forms.

  Developers populate layouts with static HTML as well as references to other presentation objects.

- **Placeholders**
  
  A Sitecore Placeholder makes it possible to reuse Sitecore layouts. With placeholders, developers indicate that the presentation objects to include in a specific area of a layout will depend on which item in the content tree has been requested by a site visitor.

- **XSLT Renderings**
  
  A Sitecore XSLT Rendering transforms content from Sitecore’s content tree into appropriate presentation language.

  XSLT Renderings are based on W3C’s XSL standards.

- **Sublayouts**
  
  A Sitecore sublayout defines the appearance of a portion of a generated web page. Sublayouts are often used to structure areas of a page. For example, a developer may use a sublayout to define an area of the page which displays three columns of content, such as a menu, some text, and a sidebar. Sublayouts can also be used to provide interaction between the user and .NET code, such as with a login form.
Sitecore Layouts are based on .NET User Controls (ascx files) and support all functionality associated with .NET User Controls.

Developers populate sublayouts with HTML, XSLT renderings, other sublayouts, and/or placeholders.

- Devices

Sitecore allows developers to define custom devices which automatically detect what hardware or software visitors are using when requesting a page. Developers then indicate which layout and other presentation objects Sitecore should use to generate the page targeted to that particular device. Devices make it easy to support very different viewing tools using the same content.

Developers have the ability to define which presentation objects Sitecore should use when generating a web page in a very generic or very specific fashion. For example, developers can define that all content based on a given template should use one set of presentation objects. On the other hand, developers can also override this general setting and define a different set of presentation objects for an individual item in the content tree.

Confused? Don’t worry, the concepts quickly become very clear when you begin working with Sitecore.
Chapter 2

Visual Studio 2005

Most developers use a combination of the Sitecore interfaces and Visual Studio when creating a Sitecore web site. This chapter provides hints for working with Visual Studio 2005.

2.1 Configuring a Visual Studio 2005 Web Application Project

This section describes the procedure required to configure a Sitecore project with Visual Studio 2005.

2.1.1 Prerequisites

These instructions assume that you have the following installed on your computer:

- Windows XP with SP2
  
  Other Windows operating systems may work, but have not necessarily been tested with these instructions.

- .NET 2.0
- A fresh Sitecore 5.3 installation
- Visual Studio 2005
- Visual Studio 2005 Update – required to support Web Application Projects


- The Visual Studio 2005 Web Application Project

By default, Visual Studio 2005 does not support the project model used in Visual Studio 2003. This change removed a crucial benefit to developers when working with Sitecore – the ability to pre-compile applications. To restore this missing functionality, Microsoft has released the Visual Studio 2005 Web Application Project.

If you have not already done so, you may download this for free from Microsoft’s web site via the following link:

2.1.2 Create an ASP.NET Web Application Project

We create a web application project in order to generate the files we will need for our Sitecore project. This will actually generate more files than we need. In later sections we will move the exact files we need to the appropriate Sitecore folder and delete the extra files.


2. Choose the File » New » Project menu command

3. Use the ASP.NET Web Application template and provide a name for the new project.
4. Select the OK button.

Visual Studio 2005 will create the project folder and related files in a folder with the name you specified, along with additional files that will not be used.

5. Close the Solution or Exit Visual Studio 2005

2.1.3 Physically Move the Project Files

As mentioned earlier, we do not actually need all the files that Visual Studio generated. In this section, we move the files we do need to the Sitecore web root.

1. Locate the generated project folder using the Windows Explorer (as shown below).

![Image of Windows Explorer window showing the generated project folder](image1)

2. Move or copy the Properties folder and .csproj files to the Sitecore web root. In the screen shot below, the Sitecore site is named SitecoreSCD1Labs, the same as the Visual Studio project name. This was done on purpose, but is not required.

![Image of Sitecore web root directory tree](image2)
3. You must set the security for the properties folder to allow Sitecore to read from the folder. The simplest way to do this on a developer machine is to set Full Control for the Everyone user (or appropriate language variant if you have a non-English version of Windows installed) to the properties folder.

4. You may choose to delete the original folder created by Visual Studio at this point if you wish.

2.1.4 **Open the Project File and Complete the Configuration**

In this section we open the project file that we have moved and perform some final configuration changes.

1. Double click on the .csproj file to open the project in Visual Studio 2005 from the Sitecore web root.

2. Exclude the default.aspx file from the project.

3. Right click on References in the Solution Explorer and choose the Add Reference… command
4. Choose the Browse tab and select the Sitecore.Kernel.dll from the bin folder.

5. Select the Sitecore.Kernel.dll file in the Solution Explorer and set the Copy Local property to false.

6. In order to edit files from within Visual Studio, we need to make them visible and then include them in the project. Start by choosing “Show all Files” from the Solution Explorer.

7. Next, include the files you may want to edit. For example, right click on the XSL folder and choose “Include in Project”.

Notice that the icon changes from an outline to a colored folder.
8. Double click on the “sample rendering.xslt” to open it.

![Image of sample rendering.xslt]

If the file appears as shown above, the project is correctly configured.

9. Save the solution.

![Image of Visual Studio build]

When prompted, choose to save the specified items in the default location.

STOP

The Visual Studio project is now complete and ready to use.
2.2 Editing a Layout, Sublayout, or XSLT Rendering

Although it is possible to edit Sitecore layouts, sublayouts, and XSLT renderings from within Sitecore using the Developer Center, using Visual Studio to edit the files associated with these presentation objects offers some advantages. These instructions assume that you have configured a Visual Studio project for your Sitecore web site.

1. Open the Visual Studio solution file from the Sitecore web root.
2. Locate the file in the Solution Explorer.
   a. You may need to turn on the Show All Files toggle.
   b. If you have created the file via Sitecore while Visual Studio was open, you may need to refresh the list as well.
3. Before you may edit the file, you must include it in your project. To do so, right click the file and select “Include in Project” from the popup menu.
   
   This will change the icon for the file from the grey outline shown in the image to the right, into an appropriate icon based on the file type.
4. Finally, double click on the file name to open it in a new tab in the Visual Studio editor.

When you make changes to the file, simply save it. The changes will be reflected immediately.

Be careful about changing files simultaneously in both Visual Studio and the Sitecore Developer Center, as you may inadvertently overwrite changes made in one editor with changes made in another.

2.3 Creating Code-Behind for Layouts and/or Sublayouts

This section describes how to associate code-behind with Sitecore layouts and/or sublayouts.

1. Create the desired layout or sublayout in Sitecore using the Developer Center.
2. Delete the layout (.aspx) or sublayout (.ascx) file generated by Sitecore in your file system.

If you would like to add code-behind to an existing layout or sublayout, rename the existing file rather than deleting it. In the next step you will create a new file. Afterwards you should copy some of the contents from the original file to the new file.
3. In Visual Studio, create a new component in the existing Sitecore project. Be sure to select the same folder for the new item as the original folder you selected in Sitecore when creating the Layout or Sublayout. You may need to include the folder in your project first.

4. Choose a new Web Form (for layouts) or a Web User Control (for sublayouts). Provide the exact same name as the layout or sublayout that you created in Sitecore.
5. Make certain that the new files (there should be three with the same name and different extensions) were created in the same folder as the original Sitecore file.

6. Open the .aspx file.

7. Add directives to the new file.
   a. If you created a layout, add:

   `<%@ OutputCache VaryByParam="none" Duration="100" %>`
   `<%@ register TagPrefix="sc" Namespace="Sitecore.Web.UI.WebControls" Assembly="Sitecore.Kernel" %>`

   Also add a directive for the code-behind class (change “Main Layout” to the name of the layout that you created):


   b. If you created a sublayout, add:

   `<%@ register TagPrefix="sc" Namespace="Sitecore.Web.UI.WebControls" Assembly="Sitecore.Kernel" %>`

   If you renamed your original file, copy the contents after the directives to the new file.

   The layout or sublayout is now ready.
Chapter 3

Templates

This chapter describes how to achieve a number of template related tasks using the Template Manager.

3.1 Creating a Template

1. Open the Template Manager (available on the Sitecore menu on the Desktop interface for all Administer users and any user with the Sitecore Client Maintaining role).

2. Select the desired parent folder for the template, for example, the User Defined folder.

3. Select the __Template command in the New chunk.

4. Specify a name for the template in the provided dialog and select OK.

The new template will appear and be selected.

3.2 Setting a Template’s Base Template

1. Select the desired template.

2. Select the View tab’s Editor command to view the Base Template field.

The base template will most often be the Sitecore Standard Template, which can be found at /templates/System/Templates/Standard template.

In some cases, however, you may want to base a new template on a user defined template that itself is based on the Standard Template.

3. Select the View tab’s Editor command to return to the tabbed view of the template.

3.3 Creating a Field Section

1. Open the Template Manager and select the desired template.
2. Place the cursor box with the grey text that reads “Add a new section”

3. Provide a name for the new field section.
4. You may now provide fields for the new section.
5. Remember to select the Build button in the Template chunk of the Template Builder to generate the new field section and any fields you create.

3.4 Creating a Field
1. Open the Template Manager and select the desired template.
2. Place the cursor box with the grey text that reads “Add a new field”

3. Type the desired name.
4. Choose the desired field type from the drop list.
5. Fill in any of the other fields as desired.
6. Move to the next line to add an additional field.
7. Remember to select the Build button in the Template chunk of the Template Builder tab to generate the new fields.

3.5 Editing Advanced Field Attributes
1. Open the Template Manager and locate the parent template.
2. Open the parent template to reveal the child field sections.
3. Open the appropriate field section and select the desired field.
4. The advanced field attributes are displayed.

3.6 Changing the Order of Fields in the Content Editor
Field sections and fields are displayed in the Content Editor in the same order in which they are displayed in the Template Manager. To change their order in the Content Editor, use the Sorting commands to change the order that they are listed in the template field section that defines them.

If the fields are defined on separate templates with an inheritance relationship, you can influence how the fields are sorted across templates as follows:

1. Select the field’s advanced field attributes (see the previous section).
2. Select Standard Fields in the View tab of the Template Manager ribbon.
3. Locate the Appearance » Sortorder field.
4. Provide an integer sort order.

5. Provide a sort order for all fields across all templates. The fields will be sorted based on the integers provided.

3.7 Setting the Icon Associated with a Template or Field Section

1. Open the template or field section in the Template Manager.

2. Select the Icon command in the Configure tab.

This will provide a menu of recently used and commonly used icons.
3. If you don’t see an icon you like, select the More Icons link at the bottom of the menu, or select the icon beside the name of the template on the title bar, which will open a dialog.

Here you can type the name of the icon if you know it, or use the browse button to open a dialog where you can search for icons by name (although the Browse dialog is relatively slow because of the large number of available icons. The next section provides a better suggestion for how to locate a useful icon).

If you change the icon in using this dialog, you must refresh the parent item in the content tree to see the new icon.

3.8 Locating Useful Icons

Sitecore includes thousands of icons that you can assign to templates and field sections to improve usability of the Content Editor. The best way to browse these icons is to use the Windows Explorer’s own Thumbnail view. You can locate the icons under the folder:

<website root>/Sitecore/shell/Themes/Standard
This folder contains many subfolders with related icons. Under each folder, icons are separated into folders based on their size. Most icons are available in 16x16, 24x24, 32x32, and 48x48 pixels, and Sitecore automatically chooses an appropriate size when displaying icons at various places in the Content Editor and other interfaces.

### 3.9 Setting Standard Values on a Template

1. Open the Template Manager and select the desired template.
2. Select the Standard Values command in the View chunk.

   Sitecore will create a **Standard Values** item and select it.
3. Provide the desired values in the associated fields.

### 3.10 Assigning Masters for all Items based on a Template

1. Open the Template Manager and select the standard values on the desired template.
2. Select the Masters chunk Assign command in Configure tab of the top ribbon.
3. Select the desired masters.

### 3.11 Assigning Layouts, Sublayouts, and Renderings

Developers associate presentation objects (Layouts, Sublayouts, and XSLT Renderings) to templates to instruct Sitecore to use these objects when generating a page for items which are based on the template in question.

1. Open the Template Manager and select the standard values on the desired template.
2. Select the Layout command in the Presentation tab of the top ribbon.

3. Select the Edit button for the desired device.

4. Select the desired layout from the Layout menu in the Device Editor dialog.

5. If the layout contains placeholders, you may associate sublayouts and renderings to the placeholders by repeating these steps for each object:
   a. Select the Add command on the Device Editor dialog.
   b. Select the desired sublayout or rendering from the Rendering droplist.
   c. Type the placeholder key in the Placeholder field.
   d. Select the OK command.
6. Use the Move Up and Move Down buttons to reorder the sublayouts and renderings if necessary.

The order is important. Items placed higher in the list will be included in the layout first.

7. Select the OK command in the Device Editor dialog when you are done.
Chapter 4

Masters

This chapter describes how to masters using the Template Manager.

4.1 Creating a Master

1. Open the Template Manager and select the desired parent template for the master.
2. Select the Masters tab.
3. Select the Master command in the New chunk.
4. Provide a name for the master.

Sitecore will create the new master and display it in the list of masters.

4.2 Editing an Existing Master

1. Open the Template Manager and select the desired master’s parent template.
2. Select the Masters tab.
3. Click on the desired master from the list of masters.

This will open the master in a separate window.

Edit the fields in the field area.

4.3 Creating a Master Hierarchy

1. Follow the instructions for editing an existing master in the previous section.
2. Turn on the Content Tree toggle in the View tab.
3. Right-click on the parent master and select a master for the sub-item, or, if you are logged in as an administrator user, select the Add from Master command, or the Add from Template command.
4. Select the desired template or master and provide a name for the sub-item.

Sitecore creates the new master sub-item.

Provide initial field values if desired.
Chapter 5

The Content Tree

Sitecore provides the Content Editor for managing the content tree. This chapter describes how to perform developer specific activities in the Content Editor.

5.1 Assigning Masters for a Single Item

Setting masters on an item or a template’s standard values influences which masters are shown in the New chunk and the various “add an item” menus available in the Content Editor and WebEdit interfaces. In most cases, developers prefer to assign masters to the standard values associated with a template or to an appropriate master, but overriding at the item level is possible.

1. Select the item in the Content Editor.
2. Select the Configure tab.
3. Select the Assign command in the Masters chunk.
4. Choose the masters you want to include for the current item via the Set Masters dialog.
5.2 Assigning Layouts, Sublayouts, and Renderings to a Single Item

Assigning layouts, sublayouts, and renderings to a single item influences how that item is shown on the website.

1. Select the item in the Content Editor.
2. Select the Presentation tab.
3. Select the Layout command in the Layout chunk.
4. Select the Edit button associated with the desired device, such as the Default device.
5. Select the desired layout from the Layout droplist.

This will open the Device Editor dialog.
6. Select the Add command to add sublayouts and renderings which will be associated with specific placeholders.

![Sitecore Developer's Cookbook](image)

7. Specify the placeholder key in the Placeholder field.

```
Placeholder: phMain
```

8. Select the OK button to complete the Add command.

9. Continue using the Add command to add all the desired sublayouts and renderings.

![Sitecore Developer's Cookbook](image)
10. Use the Move Up and Move Down buttons to reorder the sublayouts and renderings if necessary.

The order is important. Items placed higher in the list will be included in the layout first.

11. Select the OK button to complete the operation.
Chapter 6

Media

Sitecore includes a powerful media library. This chapter offers instructions for performing some common tasks.

6.1 Upload Multiple Images via a ZIP Archive

1. Open the Media Library.
2. Select the parent folder for the uploaded images.
3. Select the Media tab.
4. Select the Advanced command in the Upload chunk.

   Sitecore displays the Batch Upload dialog.

5. Use the Browse command and locate the desired ZIP archive.
6. Select the “Unpack ZIP Archives” toggle.
7. Select the Upload command.
Sitecore uploads and unzips the archive and displays all files in the Uploaded Media Items portion of the dialog.

8. Select on individual items to edit them or select the Close command to complete the upload.
Chapter 7

The Developer Center

The Developer Center offers a rich set of functionality for Sitecore developers. Indeed, developers may choose to access the Developer Center directly from the login page, just as content authors may choose to access the Content Editor. The Developer Center provides access to all areas of Sitecore, including the content tree, templates, security settings, and so on. The Developer Center is used primarily, however, when creating and editing presentation layer objects, such as layouts, sublayouts, and XSLT renderings. This chapter focuses on using these features within the Developer Center.

7.1 Creating a Layout, Sublayout, or XSLT Rendering

Developers must always create layouts, sublayouts, and XSLT renderings via the Developer Center. The Developer Center creates both the associated Sitecore item and .NET file.

1. Open the Developer Center
2. Select the Start Page or choose the View » Start Page menu command.
3. In the Getting Started, choose the appropriate “Create a New…” command.

![Getting Started](image)

4. This opens a wizard which asks for:
   a. The name of the new presentation layer object
   b. The location of the Sitecore item that describes the object
   c. The location of the .NET file that implements the object

When you have completed the wizard, Sitecore creates the object and opens the related file in a new tab.
7.2 Editing an Existing Layout, Sublayout, or XSLT Rendering

1. Open the Developer Center
2. Check to see if the desired object exists in the Recent Files portlet of the Start Page. You may want to update the list by using the Refresh button.
3. If the file you want is in the list, select it with a single mouse click to open the file in a new tab.
4. Otherwise, select the Open command in the same portlet, which opens the Open Solution Item dialog.
5. Locate and select the desired item and select the Open command.

![Open Solution Item dialog](image)

7.3 Adding a Placeholder to a Layout or Sublayout

Placeholders allow you to dynamically assign sublayouts and renderings to a specific location in a layout or sublayout.

1. Open the layout or sublayout in the Developer Center
2. Select the View » Toolbox menu command.
3. Click and drag the Placeholder from the Common area to the desired location on the layout or sublayout.

This will create a small box that corresponds to the placeholder at the location that you released the mouse button.

![Placeholder](image)
4. Double click this box to open the Placeholder properties dialog.

5. At a minimum, set the key you will use to indicate this placeholder when assigning sublayouts and XSLT renderings to a placeholder for a template or item.

You may also set other parameters, set a Data source (which is passed into the sc_item variable in XSLT renderings), and enable caching.

6. Select the OK button to complete the operation.

7.4 Statically Placing Sublayouts and XSLT Renderings

When a sublayout or rendering will always appear at a certain location, there is no need to use placeholders. Simply place the desired item at the desired location.

1. Open the target layout or sublayout in the Developer Center.
2. Select the View » Solution Explorer menu command.
3. Locate the desired item.
4. Click and drag the desired item to the target location on the layout or sublayout which will create a small box that corresponds to the item.
7.5 Creating a Table

The Developer Center provides a table wizard to simplify the process of creating tables in layouts and sublayouts.

1. Open the desired layout or sublayout in the Developer Center.
2. Position the cursor where you would like the table to be inserted.
3. Choose the Table Wizard command in the toolbar and move the mouse over the grid to indicate the size of the desired table.
4. When the size is appropriate, click the left mouse button.

A minimally sized table will appear at the cursor location.

7.6 Editing a Table

1. Open the desired layout or sublayout in theDeveloper Center.
2. Place the cursor in an existing table.
3. Choose the Table Wizard command in the toolbar.

The commands at the bottom of the wizard are now enabled. Move the mouse over each command to see flyover help describing its function.

The wizard includes commands to insert and delete rows and columns, merge and delete cells, and to edit cell and table properties.
Chapter 8

Security and User Management

This chapter offers hints for associated with Security and User Management.

8.1 Creating a User

There are several ways to create a Sitecore user. The instructions below describe the quickest.

1. Log in to the Sitecore Desktop.
2. Open the Control Panel.
4. Select the Create a New User task.
5. Fill in the user information requested in the provided dialog.
6. Choose the roles you want to assign to the new user.

7. Choose a location for the user (users are customarily placed in the User Defined folder).

8. Review the summary information and select the Create button to complete the operation.

9. Select the Finish button to exit the wizard.

The user now exists and will be able to log in.
Chapter 9

Workflows

This chapter offers hints for configuring workflows.

9.1 Creating a Workflow

Workflows are defined as sub-items under the /sitecore/system/Workflows item. You must have either the Sitecore Client Developing role or be an Administrator user to configure Workflows.

1. Open the Content Editor.
2. Enable “Entire Tree” and “Hidden Items” in the View tab.
3. Select the /sitecore/system/Workflows item.
4. Select __Workflow master from the New chunk on the Home tab.
5. Provide a name for the new workflow.
6. You must set an initial state for the workflow after you have created at least one state.

9.2 Creating a Workflow State

1. Select the Workflow item for the target workflow in the Content Editor.
2. Select the __Workflow State master from the New chunk on the Home tab.
3. Provide a name for the new state.
4. Choose whether the state should be a “final” state.

   Items contained in final states are publishable. Items contained in states which are not final are not publishable. Workflows may have any number of final states, but usually have at least one.

9.3 Creating a Workflow Command

1. Select the Workflow state item for the target state in the Content Editor.
2. Select the __Workflow Command master from the New chunk on the Home tab.
3. Provide a name for the new state.
4. Choose the next state for items when this command is chosen.
5. Change the default icon for the command.

Sitecore will display the icon you choose beside the command in appropriate locations in the Content Editor and Workbox.

9.4 Creating a Workflow Action

You may associate actions to either states or commands. An action is a side effect that takes place when the user selects a command or an item enters a state. The action has associated .NET code. You may use code provided by Sitecore or create your own custom code. In the example below, we use the Sitecore provided code which automatically publishes an item.

1. Select the target Workflow state or command in the Content Editor.
2. Select the __Workflow Action master from the New chunk on the Home tab.
3. Provide a value for the Type field, for example:

```
```

In this example, “Sitecore.Workflows” corresponds to the namespace of a class named “Simple” which contains a method named “PublishAction”. “Sitecore.Kernel” corresponds to the name of the library file containing the class. The library file, which has the extension .dll, is expected to be in the web root/bin folder.

4. Provide any parameters in the Parameter field, for example:

```
depth=1
```

In this example, this string is passed to the PublishAction method, which is free to use it as it wishes. In the case of the PublishAction method provided by Sitecore, the depth parameter indicates whether sub-items should also be published.

9.5 Provide Write and Delete Access to Items in a Workflow

By default, all users inherit read access to all workflows (via the Everyone role). Users do not, however, have write or delete access to items in workflow states unless explicitly granted access. Users must have Write access to items in order to perform workflow commands.

1. Open the Security Editor.
2. Select a specific role.

For example, to provide access for all users, select the Everyone role. To provide access to a specific group of users, such as all the users who have been assigned the “Automobile Author” role, select that role.

3. Select the target workflow or state item.

Note, states inherit security settings from their parent workflow unless “Inherit” is explicitly disabled for the corresponding item.

4. Change the “Write” and/or “Delete” access right settings from “Inherit” to “Allow”.

The users in question will now have the specific access rights.
9.6 Hide a Workflow State in the Workbox for Certain Users

Users who have read access to a workflow state will see that state in their workbox as long as the state includes workflow commands that they also have read access to. If business requirements state that a given workflow state should be hidden from a given set of users, you may restrict access to that state for those users in one of two ways:

- Hide all workflow commands in the state from the users in question
- Explicitly hide the workflow state itself from the users in question

Please refer to the appropriate section to learn how to hide workflow commands for specific users.

Explicitly hiding a state can be achieved in one of two ways:

- Turn off the “Inherits” toggle for the workflow state item and do not grant read access to the workflow state for the user and all the roles assigned to the user.
- Deny read access for the workflow state item to the user or any of the roles assigned to the user.

Each of these approaches has its advantages and disadvantages.

- Turning off the “Inherits” toggle will require that you explicitly grant access to all roles that should see the state in the Workbox. Therefore, this approach works best when the number of users or roles which should see the state in the Workbox is small.
- On the other hand, denying read access takes precedence over allowing access, which means that any user that has multiple roles may “inadvertently” not be able to see the workflow state. In other words, denying read access can have unanticipated results.

In general, we recommend turning off the “Inherits” toggle and explicitly allowing access rights when the number of roles that require access is manageable.

9.7 Hide a Workflow Command for Certain Users

The Content Editor and Workbox will only display workflow commands for non-Administrator users in cases where the following statements are true:

- The user has write access to the associated item.
- The user has write access to the command’s parent workflow state.
- The user has read access to the workflow command itself.

Therefore, configuring Sitecore’s security settings such that a user does not meet any one of these criteria will hide the workflow command from the user. In cases where users should have write access to both the item and the workflow state, there are two ways to deny read access to the workflow command.

- Turn off the “Inherits” toggle for the workflow command item and do not grant read access to the workflow command for the user and all the roles assigned to the user.
• Deny read access for the workflow command item to the user or any of the roles assigned to the user.

Each of these approaches has its advantages and disadvantages.

• Turning off the “Inherits” toggle will require that you explicitly grant access to all roles that should see the workflow command. Therefore, this approach works best when the number of users or roles which should see the command is small.

• On the other hand, denying read access takes precedence over allowing access, which means that any user that has multiple roles may “inadvertently” not be able to see the workflow command. In other words, denying read access can have unanticipated results.

In general, we recommend turning off the “Inherits” toggle and explicitly allowing access rights when the number of users and roles that require access is manageable.

9.8 View Workflow Comments

Whenever a user executes a workflow command, Sitecore offers a dialog where they can provide a comment. You can view these comments, as well as a history of workflow actions taken via:

• The History command in the Content Editor Review tab Workflow chunk.

• The More command associated with the item in the Workbox.
Chapter 10

General Tips and Tricks

This chapter offers a final grab bag of additional information.

10.1 The Sitecore Internet Explorer Toolbar

The Sitecore Internet Explorer Toolbar is an unsupported tool that dates back from the Sitecore V3 and V4 days, but still provides useful features for Sitecore V5 users. Installing it is optional.

The toolbar provides the following features:

- Change between multiple Sitecore sites via a drop list.
- Optionally change the IIS web root directory when switching between sites.
  
  This is particularly useful when developing on Windows XP, which only supports a single IIS web site root directory.

- Buttons to automatically switch between the published site and the Sitecore desktop.
- Tools to kill the ASP.NET worker process or restart IIS.
- Note the Debug and Trace buttons do not work with Sitecore V5.

10.1.1 Installation

The following steps describe how to install the toolbar.

1. Copy the SitecoreBar.dll file to the C:\Windows folder.
2. In a Command Prompt, type:

   C:\> C:\Windows\System32\regsvr32 C:\Windows\SitecoreBar.dll

   Which will display the following dialog:

   ![RegSvr32 dialog](image)

   DllRegisterServer in c:\windows\sitecorebar.dll succeeded.

3. Open an Internet Explorer window.

4. Select View » Toolbars » Sitecore menu command to display the toolbar.

   ![Internet Explorer toolbar](image)

   You may need to reposition the toolbar to make it fully visible.
10.1.2 Configuration

1. Right click on the toolbar and select the Edit websites command.

   ![Sitecore Websites dialog box]

   This displays the Sitecore Websites dialog box.

2. Select the New button to create a new website reference.

   ![Sitecore Site dialog]

   This displays the Sitecore Site dialog.

3. Provide a name and URL for the site.

4. Choose Sitecore V4, which works with Sitecore V5 as well.

5. You may provide a username and password to automatically log into the Sitecore client.

6. If the site is installed on the local machine, choose “Change IIS home path” and provide the location of the “website” directory of the site.

7. Select OK to close the dialog.

![STOP] The IE Toolbar is now installed and configured.
10.2 Generating “Lorem Ipsum” Filler Data

Developers often want to add a large amount of filler text to simulate actual site content during development. This helps when designing pages and viewing the site in progress.

Sitecore’s template standard values provide a useful location for such filler data, as it can be removed throughout the site relatively easily.

To generate “Lorem Ipsum” dummy text, the standard since the 1500s, visit the following site:

http://www.lipsum.com/

10.3 Pointing to an Alternative Home Item

Sitecore expects the default item for a site, the site’s “Home” item, to be called “Home”, but this is not required. If you want to call the Home item something else, you just need to make a few changes to the web.config file found in the Sitecore website folder (usually found under the Inetpub folder in the Windows Explorer).

The changes include:

- Find the element: `<site name="shell"...`  
  Change the attribute: `contentStartItem="/Alternative Home"`

- Find the element: `<site name="website"...`  
  Change the attribute: `startItem="/Alternative Home"`

- Find the element: `<setting name="DefaultItem"...`  
  Change the attribute: `value="/Alternative Home"`

10.4 Using a Custom CSS File

Sitecore is configured to use the default.css file in the site web root, but you may use a different CSS file if you wish. Sitecore specifies a CSS file in several locations. The web.config file specifies a CSS file that influences the CSS classes provided in the rich text editor.

1. Copy the desired CSS file to the site web root (or note its location).
2. Open web.config in a text editor.
3. Locate the WebStylesheet setting.
4. Change the setting to reference the desired CSS file.

Sitecore uses a template which is copied when creating new layouts. Change this file to point to the desired CSS file.

1. Open the `{web root}\sitecore\shell\Templates\layout.aspx file.
2. Change the reference to the default.css to the desired CSS file.

   This will influence newly created layouts.

You must manually open and update any existing layout files.
10.5 Creating a Device

Sitecore provides two default devices: Default and Printer. You are free to create as many custom devices as you like. To create a new device, you must be logged in as either an Administrator user or a user with the Sitecore Client Developing or Sitecore Client Maintaining roles.

1. Open the Content Editor
2. In the View tab, turn on the Entire Tree and Hidden Items toggles.
3. Open the /sitecore/layouts item.
4. Select the /sitecore/layouts/Devices item.
5. Select the New __Device master.
6. Provide a name for the new device, such as “PDA”.
7. Specify an icon for the device.
8. Specify any desired attributes for the device.
9. Save your changes.

The device will now appear in the Layout menu in the Presentation tab.

10.6 Creating a Device Specific Site

Sitecore can support multiple published sites. The standard approach to supporting an additional site is to modify the <sites> element in the web.config file. This section describes how to create and configure a new site.

1. Locate the <sites> element in the web.config file.

   Notice the comment above the <sites> element that defines all the properties of the <site> element. The properties provide configuration details about the site, such as which database Sitecore should use when generating pages for the site, what URL corresponds to the site, and so on.

   You will notice that Sitecore comes configured with many sites. The order that the sites are defined in the <sites> element is important, as it is this order that attempts to match a given site with an incoming request.

2. Locate the “website” site.

   This site corresponds to the published website.

3. Copy the element associated with the website site and paste the copy immediately above the “website” element.

4. Change the name attribute to something appropriate, such as “pda”.

5. If you wish to access the site via a special hostname, provide the hostname attribute, such as hostname=”pda.localhost”.

   Using pda.localhost is a good approach when testing a site on your local machine. To use
this approach, however, you will need to update the machine’s host file, found in
\c:\windows\system32\drivers\etc\hosts, to add the line:

127.0.0.1  pda.localhost

6. Provide the device attribute, such as device=”pda”, where “pda” is the name of the device
Sitecore should select when handling requests for this site.
7. The rest of the elements can stay the same as the copied element.

Here is an example device specific site element:

```xml
<site name="pda" hostName="pda.localhost" device="pda" virtualFolder="/"
physicalFolder="/" rootPath="/sitecore/content" startItem="/home"
database="web" domain="extranet" allowDebug="true" cacheHtml="true"
htmlCacheSize="10MB" enablePreview="true" enableWebEdit="true"
enableDebugger="true" disableClientData="false">
</site>
```

10.7 Installing a Package

The following instructions describe how to upload a Sitecore package. Please note that Sitecore
does not currently include an “uninstall package” feature. Please backup your site before
installing a package.

1. Open the Sitecore Desktop interface.
2. Open the Sitecore » Development Tools » Installation Wizard application.

This will open the Installation Wizard dialog.
3. These instructions assume that you have not yet uploaded the package you would like to install.

Select the Upload button.

This will open the Select Files dialog.

4. Select the Browse… button.

5. Locate the desired package in the file system and select the Open button.

6. Select the Next button on the Select Files dialog.

7. Select the Next button
8. Select the Finish button after the file has been uploaded.
9. On the Installation Wizard, select the Next button and follow the instructions.

### 10.8 Adding a Language

Only Administrator users may add or remove languages.

1. Log in as an administrator user, such as admin.
2. Open the Control Panel.
3. Select the Globalization page.
4. Select the Add a New Language task.
5. Provide ISO codes for the language and country/region code.

6. In most cases you will be able to accept the default codepage, encoding, and charset, but you may provide the values you require if necessary.
7. Provide a spell checker dictionary path. Dictionaries are placed in the webroot bin folder by default. Sitecore provides dictionaries for many common languages. Dictionaries are stored in Open Office format, so you may download dictionaries for many languages for free over the internet.

8. Close the wizard when the operation is complete.

9. Select the new language in the Content Editor.

It will be located as a sub-item of the /sitecore/system/languages item.

10. Configure the icon for the item to correspond to the appropriate country flag.

11. The language will now appear in appropriate language menus.