Understanding Sitecore Fundamentals

Author:	Derek Roberti, Sitecore USA
Date:	February 22, 2007
Release:	Revision 1.0
Language:	English

Table of Contents:

Chapter 1

Introduction

To understand how Sitecore works, developers need to be familiar with ten fundamental concepts. This document provides an introduction to each of these concepts, with pointers to useful resources.

1.1 Related Documents

Throughout the document, readers will be referred to the Sitecore Developer's Cookbook, which can be downloaded here: http://sdn5.sitecore.net/Developer/Developers%20Cookbook.aspx. The Developer's Cookbook provides step-by-step instructions on completing common Sitecore tasks.

It will also be useful for developers to familiarize themselves with the business-user interfaces in Sitecore. These UIs support business users in basic content authoring and editing scenarios. For information on how to edit content in Sitecore, see the End User guide which can be downloaded here: http://sdn5.sitecore.net/End%20User/Using%20Sitecore.aspx

Chapter 2

Top 10 Concepts

The ten concepts discussed in this document can be divided into data concepts and presentation concepts:

Data	Presentation
Databases	Devices
Items	Layouts
Templates	Sublayouts
Fields	Renderings
Masters	Placeholders

The data concepts focus on how Sitecore stores and structures data as well as how business users enter data into Sitecore. The presentation concepts describe how Sitecore presents data based on the requesting agent (a web browser, RSS reader, etc.).

This division between data constructs and presentation constructs emphasizes the separation of content and presentation that is fundamental to how Sitecore approaches managing web-based content. Some of the advantages of this separation of content and presentation include:

- Business users enter content once. The same content can be presented in different parts of the site (for example, in a summary page, a detail page and a navigation component).
- Content can be presented differently based on which device is making the request. For example, Sitecore can present content differently to web browsers, PDAs or RSS readers.
- Presentation is consistent across the site. Because business users are focused on crafting content and not presentation, branding, usability and accessibility can be kept consistent across the site.

This discussion of the fundamental Sitecore concepts is intended to be introductory in nature. All of these concepts are discussed in greater detail on the SDN5 site (http://sdn5.sitecore.net) and in the SDN5 Forums. Details are left out here in order to allow developers to be introduced to the high-level concepts before diving in to Sitecore at a more granular level.

2.1 Databases

Sitecore stores content in seven databases: Web, Master, Security, Extranet, Recycle Bin, Archive and Core.

The Web and Master databases store content information, including content, metadata and security settings on content items.

Master	The Master database includes all versions of all content, in all languages, published and unpublished, in all states of workflow. This is where business users author and edit content.
Web	The Web database includes the latest version of published content that is not in workflow.

The Security and Extranet databases store user and role information for business users and public visitors to your website.

Security	The Security database stores user and role information for business users, i.e. the authors, editors and developers that will be accessing the Sitecore user interfaces.
Extranet	The Extranet database stores user and role information for public visitors to your website.

The Recycle Bin and Archive databases store content that is no longer in use.

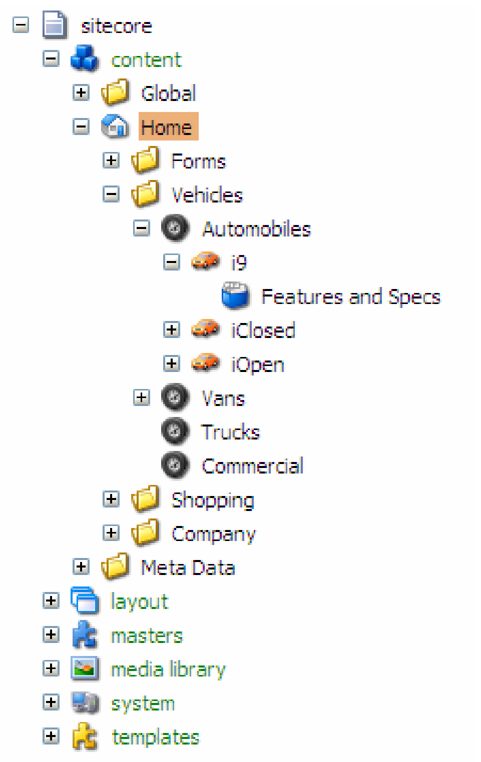
Recycle Bin	The Recycle Bin works like the Windows Recycle Bin. When users delete content items, they are moved to the Recycle Bin. Items stay in the Recycle Bin until an administrator empties the Recycle Bin.
Archive	The Archive database is similar to the Recycle Bin, except that it doesn't get emptied. The Archive is a convenient place for business users to place content that is no longer in use, but which could be of use in the future. Moving the content items out of the Master database removes clutter from the Sitecore content tree. (More about the content tree later.)

Finally, there is the Core database.

Core	The Core database is like a large configuration file for the Sitecore user interfaces. You may access this database if you are customizing the Sitecore UIs; for example, when adding a new application to the Sitecore desktop.

2.2 Items

Sitecore structures content into a content tree (or content hierarchy). Every node in the content tree is called an Item.



Note that the node that corresponds to /sitecore/content/Home is typically the homepage of your website. The content tree includes items that store content as well as metadata items, media items and system settings.

When a request comes in to Sitecore for http://localhost/Vehicles/Automobiles/i9/Features and Specs.aspx, sitecore looks up the corresponding item in the Sitecore content tree, in this case /sitecore/content/Home/Vehicles/Automobiles/i9/Features and Specs.

We refer to content items as "Items" as opposed to "Web Pages" for a number of reasons. First, the items in the content tree do not refer to physical files. Sitecore uses the requested URL as a record locator for the appropriate item in the content tree. Second, Sitecore serves more than just web pages. Requests can come in from RSS readers, web services, etc. that never display content on a page, *per se*. Finally, many of the content items in the content tree are never resolved as web pages. They may simply have metadata information that is used by your Sitecore implementation but which is never accessed by a public URL.

Below you will learn that all of the items in the Sitecore content tree are based on "Templates." Templates have a special role in Sitecore and provide the underlying structure of all content items.

2.3 Templates and Fields

Templates in Sitecore are data constructs, not presentation constructs. This can be initially confusing if you are accustomed to using the term "template" to refer to how content appears on a web page. In Sitecore, a template describes two things: how content is structured and how business users enter content.

To understand templates, it is useful to think about constructs in object-oriented programming (OO). In OO, we define classes. Classes have properties and methods as well as constructors. We create an object by instantiating a class. In Sitecore, we say that we define templates. Templates have fields. We instantiate templates using masters. The resulting object is an item.

OO Term	Sitecore Term
Classes	Templates
Properties	Fields
Constructors	Masters (discussed in the next section)
Objects	Items

While this comparison is strictly conceptual, it provides a useful metaphor for understanding Sitecore data constructs.

2.3.1 Fields

Templates consist of fields. A field stores a unit of data, such as an article title, a product description or a publication date. A simple template for a product may look something like this:

Product
Title
Menu Title
Short Description
Description
Image
Price

Here, the “Title” field refers to the name of the product; the “Menu Title” field refers to the name of the product as it should appear in navigation controls (such as breadcrumbs); the “Short Description” field is a description of the product as it should appear in summary pages (for example, a list of all products with brief descriptions); the “Description” field is the full description of the product; “Image” is a picture of the product; and the “Price” field is the price of the product.

Creating Templates

For step-by-step instructions on how to create a template, see the Sitecore Developer’s Cookbook, Chapter 3.1.

This means that every time a business user creates a new product on the site, they will be presented with six fields that they can update in the item editor.

To determine the UI control that business users use to edit content, specify the appropriate “field type” for each field. For example, the “Title” field can be a text field, while the “Description” field should be a rich text field. Sitecore provides many field types out-of-the-box for common data entry tasks.

Field types include:

- Text (a single line of text)
- Memo (a text area for plain text)
- Rich Text (presents the user with a WYSIWYG editor)
- Date (provides a "calendar-picker" interface)
- Image (allows users to choose an image from the Media Library)
- Lookup (presents users with a drop-down list)

Updating our template model, we can add field types:

Product	
Field Name	*Field Type*
Title	Text
Menu Title	Text
Short Description	Memo
Description	Rich Text
Image	Image
Price	Text

Note that field types do not specify a "data type" in the way you would expect from a standard database. The field type specifically designates which user interface control a business user will use to enter data into a field.

Field Reference

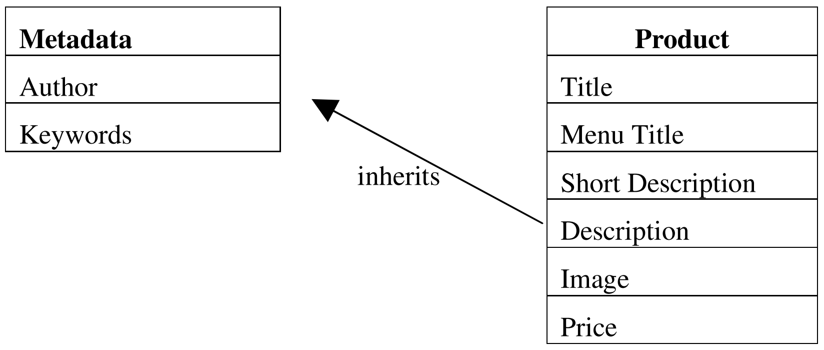
For a complete field reference, see the Sitecore Developer Network (SDN):

http://sdn5.sitecore.net/Reference/Field%20Reference.aspx

2.3.2 Inheritance

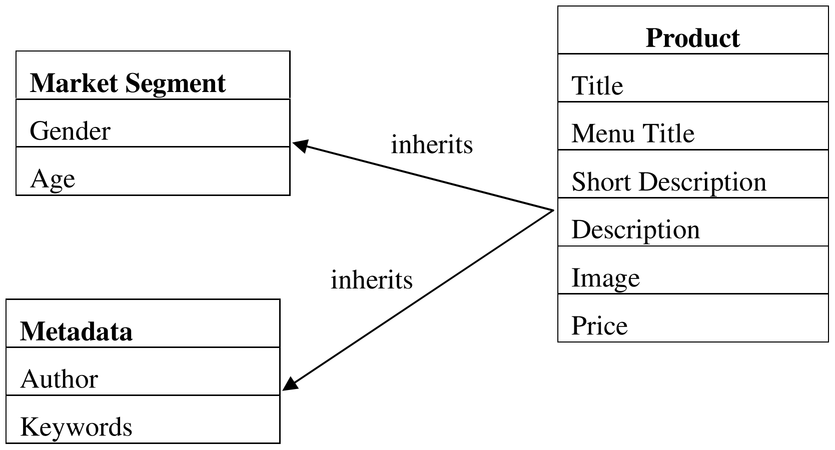
Imagine that all content items on my site should have certain metadata fields, such as "Author" and "Keywords." To add these fields to the Product template is easy, but has limitations. For example, if my site has many templates, I will have to add these fields to every template on my site. If I need to add or modify a metadata field, I will need to return to every template on my site to add or modify the appropriate field.

Sitecore provides a more flexible approach to this requirement. Instead of listing the metadata fields in the Product template, we can create a separate Metadata template and establish an inheritance relationship between the Product template and the Metadata template.



To say that "the Product template inherits from the Metadata template" means that when a business user edits a product (i.e. an item based on the Product template), they will see all of the fields in the Product template and all of the fields in the Metadata template. If all templates on the site inherit from the Metadata template, then adding a new metadata field to all content items is as easy as adding a new field to the Metadata template.

Note that one template can inherit from any number of templates. Imagine the following scenario:



Now, when a business user edits a product, they will see all of the fields for a Product, all of the fields for a Market Segment and all of the fields for Metadata. In this way, developers can model templates similarly to how they would model classes in an OO design.

Setting Template Inheritance

To define template inheritance, see the Developer's Cookbook, Chapter 3.2.

2.3.3 Standard Values

Often, developers set default values for business users. For example, when business users create new content items, workflow and presentation settings should be pre-configured. Additionally, developers may wish to pre-populate field values. For example, the "Description" field in the Product template may always have *lorem ipsum* placeholder text in it.

Sitecore addresses this requirement using "Standard Values." Standard values follow a simple rule:

If a Field is null in an Item, the Template Standard Value will be used in place of the null value.

This means that if a business user has never entered a value into a field, Sitecore will look to see if a value has been set in the template standard values. The standard values feature provides tremendous flexibility in a Sitecore solution. Imagine, for example, that all items based on the Product template need to be assigned to a different workflow. Instead of changing every single item, developers can simply change the standard value in the underlying template. This change will affect all items based on the template whose workflow field remains null at the item level.

Note that Sitecore treats null values differently from empty strings. If a business user has entered data into a field and then deleted the data, Sitecore no longer believes the field is null. If the business user wants to revert a field to the template standard values, an extra step must be taken to null the field.

Setting Template Standard Values

To work with Standard Values, see the Developer's Cookbook, Chapter 3.9. For additional discussion, see here:

http://sdn5.sitecore.net/End%20User/Template%20Manager/Standard%20Values.aspx

2.4 Masters

In OO, we create objects using constructors. Similarly, in Sitecore, we create items using masters. While developers can create new content items using templates, business users must use masters to create new items. Creating new items using masters is almost the same as creating new items based on templates. Every master is based on a template. For example, while developers can create new items based on the Product template, a business user would create a new item based on a master which, in turn, is based on the same Product template.

Masters provide three useful features:

1. Define acceptable children.
2. Define default children.
3. Define default values.

We will look at each of these features individually below.

2.4.1 Acceptable Children

Developers understand and configure templates. Business users don't. Configuring templates is typically beyond the understanding or area of responsibility for most content authors. This is appropriate, as business users should focus on creating content, not on remembering which content items are based on which templates.

Developers use masters to ensure that business users are using the appropriate template at a specific node in the information architecture (IA). For example, business users should not create new products in the "Contact Us" section of the site. Similarly, they should not create new contact information in the "Products" section of the site.

To ensure that the correct content items are created in the correct section of the site, we configure the masters for the site section. This means we are saying "These masters may be used to create new items in this section of the site." When business users right-click content markers (the green dots) in WebEdit mode, they are seeing the masters that you have configured for them.

Creating and Configuring Masters

To create a master, see Chapter 4.1 of the Developer's Cookbook. To configure masters for a content item, see Chapter 5.1.

2.4.2 Default Children

Default children, or "master hierarchies" as they are sometimes called, define which child items should be automatically created beneath a new content item. Imagine, for example, that every time a new product is created, three new child items should be created:

- My New Product
  - Features and Specs
  - Accessories
  - Reviews

To create this hierarchy, we could rely on business users to remember to create the child items every time they create a new product. Alternatively, we could have Sitecore *automatically* create the child items every time a business user creates a new product. This reduces effort on the part of business users as well as ensuring that the correct content items are created.

Specifying Default Children

To create a master hierarchy, see Chapter 4.3 of the Developer's Cookbook.

2.4.3 Default Values

We can also use masters to specify default values. The most important aspect of this feature is the use of the $name token. When a new content item is created using a master, Sitecore looks in each field of the master. If it finds $name, it will replace $name with the name of the new content item being created.

If we create a master based on the Product template, we might put $name in the Title and Menu Title fields. Sitecore will automatically populate those fields with the name of the new item that the business user creates.

2.4.4 Template Standard Values vs. Masters

Masters may seem to provide similar functionality to template standard values, but they are distinct. Here are some important distinctions between the two:

1. Changes to masters only affect new items that are created. Changes to template standard values affect all items – new and already existing – that are based on the template.

2. Template standard values only appear in null fields, whereas masters actually copy data into fields. If a value is specified in a master, the resulting value in the new content item is not null.

3. The $name token only works with masters.

4. We use masters to configure acceptable children and master hierarchies.

5. We typically configure workflow and layout settings for items using template standard values.

So far, we have focused on modeling content in Sitecore. In the following sections, we discuss Sitecore's presentation concepts and how Sitecore connects content with presentation.

2.5 Devices

In a sense, Sitecore links content with presentation through devices. A device in Sitecore describes both attributes of the HTTP request as well as presentation preferences for the HTTP response.

In simple terms, a device is the requesting agent. For example, a browser, a cell phone and an RSS reader can all be considered devices. As developers, we want to send different presentation to each of those devices based upon pre-defined understandings of the device constraints. For example, a browser should have full navigation, a cell phone should not have large images and an RSS feed should comply with the RSS specification. Sitecore still serves the same content we created in our content tree, but it presents the content differently based on the device making the request.

Sitecore detects devices based on one of several options. Two common options are user-agent string and query string. Developers can configure Sitecore to determine the device for the current request based on its HTTP user agent string. For example, if developers want to apply special presentation settings for the Blackberry browser, they can detect Blackberry requests with the "BlackBerry8700/4.1.0" user agent string. Alternatively, devices can be detected with query string values, such as "http://www.foo.com/mypage.aspx?pda=1".

Based on the device making the request, Sitecore will serve content items using different presentation settings (commonly referred to as "Layout settings" in Sitecore).

Configure a Device

To configure a device, see Chapter 10.5 of the Developer's Cookbook.

2.6 Layouts

A layout is the highest-level presentation object in the Sitecore presentation stack. A layout is similar to an ASP.NET master page. It defines the presentation elements that persist across all requests for a particular device. On a website, a layout typically includes a site header and footer. The rest of the presentation elements are dynamically assembled at request time.

A layout is an .aspx file type (or, in ASP.NET terms, a Web Form). Developers typically create one layout per device.

Creating Layouts, Sublayouts and Renderings

To create a layout, sublayout or rendering, read about the Sitecore Developer Center in Chapter 7.1 of the Developer's Cookbook.

The "big picture" of presentation in Sitecore is that developers define presentation objects modularly and reuse presentation objects in many different presentation scenarios across the site. As we will learn below, developers typically define many small units of presentation (for example, a breadcrumb, a product list or a new article) and assemble the appropriate presentation elements when the system receives an HTTP request.

2.7 Sublayouts

A sublayout is used in two different scenarios: to define a presentation area or to include .NET functionality such as an interactive form. A sublayout is simply an .ascx file (a User Control, in ASP.NET terms) that can be included in a layout or sublayout.

In the first case, a sublayout may be used to define an area on a web page. If you have a two-column design that includes a left navigation and a main content area, you can define this two column space in your sublayout (either through HTML tables or CSS).

In the second case, for example, a search form can be created as a sublayout and included in all content items (or their corresponding templates) that require search functionality. Note that almost anything you can do in ASP.NET code behind you can do in a Sitecore sublayout. Developers accustomed to coding in C# or Visual Basic will find themselves at home with Sitecore sublayouts.

2.8 Renderings

A rendering is typically used to render content. An example could be a breadcrumb, the text of an article or a list of products. Renderings use XSL to transform Sitecore's native XML into appropriate presentation code (i.e. XHTML or XML). Renderings can be used with layouts or sublayouts.

Renderings and sublayouts can both be used to render content. It is often a developer's choice as to which technology to use. Renderings allow rapid prototyping and easy caching, while sublayouts may perform more quickly and are easy for those familiar with ASP.NET but unfamiliar with XSL. Renderings are generally not appropriate for anything but the most simple forms.

2.9 Static and Dynamic Placement

Imagine that you have defined all of your layouts, sublayouts and renderings for your site. How do you assemble these in response to an HTTP request? Sitecore provides you with two approaches, both of which are typically used on a site: static placement and dynamic placement.

Static placement refers to the placement of a rendering or a sublayout on to a sublayout or layout. If a rendering or sublayout is statically placed, it will always render when the sublayout or layout it is associated with gets rendered. For example, you may want your site logo to render in every HTTP response. To address this, you can create a rendering that presents the logo and statically place the rendering on your layout. That means that every time the layout is used, the logo will also be rendered.

Dynamic placement refers to the association of a sublayout or rendering with a placeholder. A placeholder is an object in Sitecore that serves as an anchor point for other presentation objects. Every placeholder is named using its "key" attribute. Renderings and sublayouts can be associated with placeholders based on the placeholder key.

For example, I can associate my product rendering with the "main" placeholder. This means that when the product item is requested, the product rendering will be placed in the same location in the HTTP response as the placeholder whose key is "main."

Dynamically placed renderings and sublayouts are typically configured in the standard values of a template. In the case of our Product template, we can configure the template standard values to use a particular layout and any number of sublayouts and renderings. All items that are based on the product template will use these presentation objects when they are rendered on the site.

Working with Static and Dynamic Placement

Chapter 7.4 of the Developer's Cookbook describes how to statically place renderings and sublayouts on to sublayouts and layouts.

Chapter 7.3 describes how to add a placeholder to a sublayout or a layout. Chapter 3.11 describes how to associate dynamically placed renderings and sublayouts on a template's standard values.

2.10 Processing a Request

It all comes together when a request comes in to the server. Here are the steps Sitecore takes:

1. Sitecore analyzes the request to determine which device is making the request.
2. Sitecore determines which content item is being requested based on the path in the URL.
3. Sitecore looks at the item's presentation settings. If there are no presentation settings at the item level, Sitecore looks at the item's template standard values.
4. Based on the presentation settings, Sitecore uses the appropriate layout and adds all statically-placed sublayouts and renderings.
5. Sitecore then adds all of the dynamically placed sublayouts and renderings and returns an HTTP response.

Chapter 3

Next Steps

The ten fundamental concepts can take you far in Sitecore. Additional concepts – security, workflows, multi-lingual functionality, etc. – are addressed on the SDN5 site (http://sdn5.sitecore.net) and in the Sitecore Certified Developer training.

Additional Reading

The following links can be useful in furthering your understanding of Sitecore:

Fundamental Concepts -- http://sdn5.sitecore.net/Developer/Fundamental%20Concepts.aspx

End-User Documentation -- http://sdn5.sitecore.net/End%20User/Using%20Sitecore.aspx

Security Concepts -- http://sdn5.sitecore.net/Articles/Security/Common%20Security%20Concepts.aspx

Workflows -- http://sdn5.sitecore.net/Articles/Workflow/Understanding%20Workflows.aspx