Sitecore CMS 6

Fundamental Concepts

An Introduction to the Sitecore 6 Central Constructs and Concepts

Table of Contents

Chapter 1

Introduction

Welcome to Sitecore CMS 6!

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

Reference Materials and Cookbooks

Throughout the document, readers will be referred to the Sitecore Reference documentation. Of particular interest are the Data Definition Reference, Client Configuration Reference, and Presentation Component Reference:
http://sdn5.sitecore.net/Products/Sitecore%20V5/Sitecore%20CMS%206/Documentation.aspx

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 Content Author's Cookbook which can be downloaded here:
http://sdn.sitecore.net/End%20User/Sitecore%206%20Cookbooks.aspx.

This document contains the following chapters:

Chapter 2

Eight Fundamental Concepts

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

Data	Presentation
Databases	Layouts
Items	Sublayouts
Data Templates	Renderings
Fields	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 SDN site (http://sdn.sitecore.net) and in the SDN Forums (http://sdn.sitecore.net/Forum.aspx). Details are omitted here in order to introduce the high-level concepts before diving in to Sitecore at a more granular level.

Chapter 3

Databases

Sitecore stores content in three databases: Web, Master, 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 has reached a final workflow state.

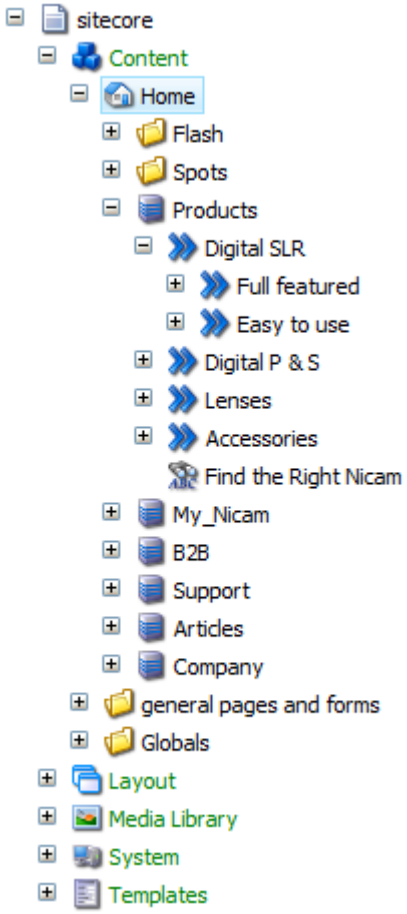
The Core database stores configuration information for Sitecore.

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.

Chapter 4

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. If you manage multiple sites with a single instance of Sitecore, you create additional sites as branches of the content tree. 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/Products/Digital SLR/Full featured.aspx, Sitecore looks up the corresponding item in the Sitecore content tree, in this case /sitecore/content/Home/Products/Digital SLR/Full Featured.

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 “Data Templates.” Data Templates have a special role in Sitecore and provide the underlying structure of all content items.

Chapter 5

Data Templates and Fields

5.1 Terminology Overview

As the name suggests, Data 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 Data Template describes two things: how content is structured and how business users enter content.

To understand Data 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 Data Templates. Data Templates have fields. We instantiate a Data Template to create an item in the content tree.

OO Term	**Sitecore Term**
Classes	Data Templates
Properties	Fields
Objects	Items

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

5.2 Fields

Data 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 Data 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 Data Templates

For more information on Data Templates, see the Data Definition Reference, Chapter 2.

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 or inline editing mode.

To determine the UI control that content authors 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:

- Single-Line Text (a single line of text)
- Multi-Line Text (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)
- Droplists (presents users with a drop-down list)

Updating our Data Template model, we can add field types:

Product	
Field Name	*Field Type*
Title	Text
Menu Title	Single-Line Text
Short Description	Mulit-line Text
Description	Rich Text
Image	Image
Price	Single-Line 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 content author will use to enter data into a field.

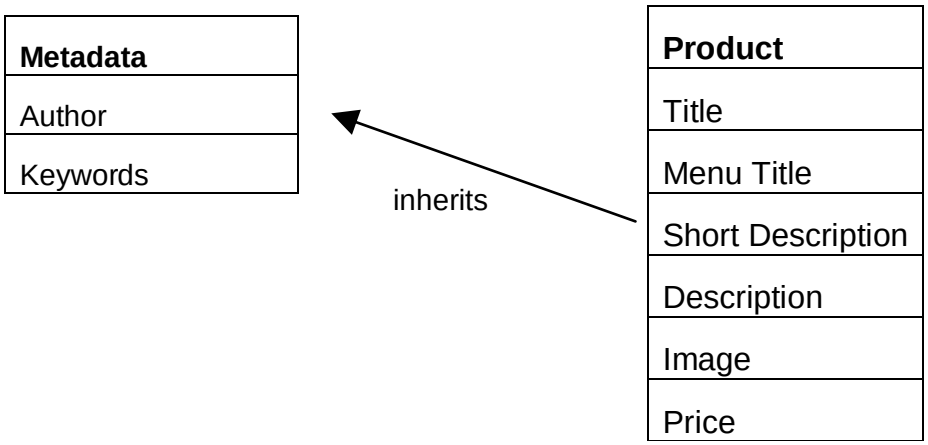
Field Reference

For a complete field reference, see the Data Definition Reference, Chapter 4.2 Understanding Field Types.

5.3 Inheritance

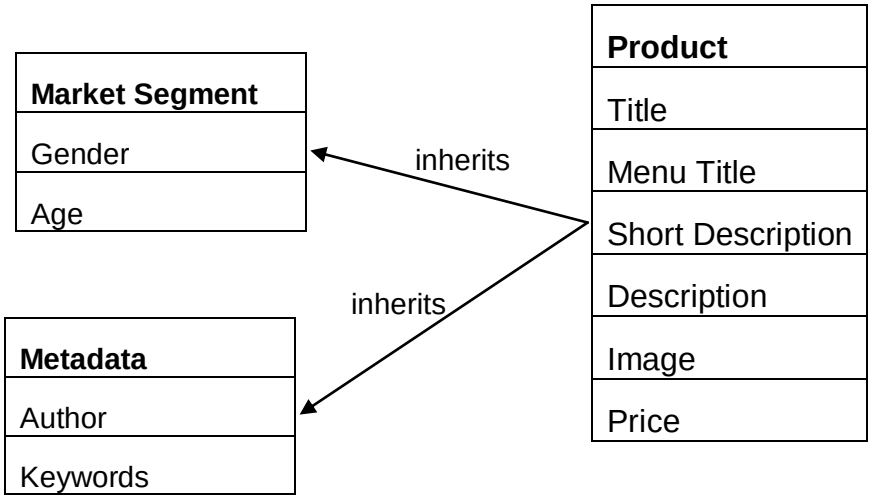
Imagine that all content items on your site should have certain metadata fields, such as "Author" and "Keywords." To add these fields to the Product Data Template is easy, but has limitations. For example, if your site has many Data Templates, you will have to add these fields to every Data Template on your site. If you need to add or modify a metadata field, you need to return to every Data Template on your 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 Data Template, we can create a separate Metadata Data Template and establish an inheritance relationship between the Product Data Template and the Metadata Data Template.



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

Note that one Data Template can inherit from any number of Data 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 Data Templates similarly to how they would model classes in an OO design.

Setting Data Template Inheritance

To see more information on Data Template inheritance, see the Data Definition Reference, Chapter 2.1.2.

5.4 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 Data 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 Data 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 Data Template. This change will affect all items based on the Data 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 or it can be configured that an empty string is automatically treated as a null value.

Setting Template Standard Values

To work with Standard Values, see the Data Definition Reference, Chapter 2.2.

Chapter 6

Adding New Items

6.1 Adding New Items Overview

While developers can create new content items using any Data Template, business users can be restricted on what item types they can create and where they can create them. For example, while developers can create new Product items anywhere in the content tree, a business user could be allowed to create new Product items only under a *specific* section of the tree. These rules are configured using Insert Options.

Insert Options provide three useful features:

- Acceptable Children
- Default Children
- Default Values

6.2 Acceptable Children

Developers understand and configure data templates. Business users don't. Configuring data 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 data templates.

Developers assign Insert Options to ensure that content authors are using the appropriate data 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.

When business users are in the Page Editor Edit mode, they will see the list of acceptable templates they can select to insert new items that you have configured for them.

Creating New Items

For additional discussion, see Content Authors Cookbook, Chapter 3.3

6.3 Default Children

Default children, or “Branches”, 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 Branches

To create a Branch Template, see Chapter 6 of the Data Definition Reference.

6.4 Default Values

We can also use template standard values to specify default values. The most important aspect of this feature is the use of the $name token, though there are other tokens such as $time and $now. When a new content item is created, Sitecore looks in each field of the template standard values for any of these tokens. If it finds $name, it will replace $name with the name of the new content item being created.

If we create an item 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.

Chapter 7

Presentation

7.1 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 see more information on devices, see Chapter 3.2 of the Presentation Component Reference.

7.2 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.

Layouts, Sublayouts and Renderings

To see more information about layouts, sublayouts or renderings, read about the Presentation Components and the Sitecore Developer Center in Chapter 2 and 4 of the Presentation Component Reference.

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.

7.3 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 Data 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.

7.4 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.

7.5 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 Data Template. In the case of our Product Data 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 Data Template will use these presentation objects when they are rendered on the site.

7.6 Configurable Placeholder Settings

While developers may choose to decide the layout of all pages, it is also possible to allow the content authors preconfigured alternate presentation choices.

The Developer can create three alternative displays of an item, for example: one with text on the left and a picture on the right, one with the picture on the right and the text on the left and one with the picture on the top and the text below it. By configuring a Placeholder Setting to allow any of these presentations, a business user with the correct permissions may be allowed to easily choose which display setting they want, without having to contact a developer.

This allows some design choices to be made easily by the content authors.

Working with Static and Dynamic Placement

See Chapter 2 of the Presentation Component Reference to see information on Placeholders and Placeholder Settings.

Chapter 8

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 9

Next Steps

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

Additional Reading

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

End-User Documentation:

http://sdn.sitecore.net/upload/sitecore6/contentauthorscookbook-usletter.pdf

Security Concepts and Cookbook:

http://sdn.sitecore.net/upload/sitecore6/securityreference-usletter.pdf

and

http://sdn5.sitecore.net/upload/sitecore6/securityadministratorscookbook-usletter.pdf

Workflows:

http://sdn.sitecore.net/upload/sitecore6/workflowreference-usletter.pdf