



# Sitecore CMS 6.2

# Sitecore Dynamic Links

*A Developer's Guide to Constructing URLs with Sitecore*

## Table of Contents

|           |  |    |
|-----------|--|----|
| Chapter 1 | Introduction.....  | 3  |
| Chapter 2 | Sitecore Dynamic Links.....  | 4  |
| 2.1       | Sitecore Dynamic Link Management .....   | 5  |
| 2.1.1     | Dynamic Link Configuration .....   | 5  |
|           | The Rendering.SiteResolving Setting .....  | 6  |
|           | The LinkItemNotFound Setting .....   | 7  |
| 2.1.2     | How to Access the URL of a Content Item .....  | 7  |
| 2.1.3     | How to Access the URL of a Media Item .....  | 7  |
| 2.2       | Search Engine Optimized (SEO) URLs .....   | 9  |
| 2.3       | IIS and ASP.NET URLs .....   | 10 |
| 2.3.1     | IIS7 or Later Integrated Mode ASP.NET Managed Pipeline.....                                | 10 |
| 2.3.2     | IIS HTTP 404 Page .....  | 10 |
| 2.3.3     | IIS Wildcards .....  | 12 |
|           | IIS 5.1 or Earlier (Windows XP) .....  | 12 |
|           | IIS6 (Windows 2003).....   | 13 |
|           | IIS7 or Later Classic Managed Pipeline Mode (Windows Vista, Windows 2008, and Windows 7) . | 13 |
| 2.3.4     | URL Rewriting ISAPI Filter .....   | 14 |

# Chapter 1

## Introduction

This document describes how to configure Sitecore dynamic link management. Sitecore administrators and developers can use this information to configure and implement Search Engine Optimization (SEO) and other link management features.

This document contains the following chapters:

- Chapter 1 – Introduction
- Chapter 2 – Sitecore Dynamic Links

## Chapter 2

# Sitecore Dynamic Links

This chapter describes ways that you can configure IIS configurations to use ASP.NET to process URLs that it would otherwise manage as static files, how Sitecore generates URLs dynamically, and Search Engine Optimization (SEO) techniques that you can use with Sitecore.

This chapter contains the following sections:

- Sitecore Dynamic Link Management
- Search Engine Optimized (SEO) URLs
- IIS and ASP.NET URLs

## 2.1 Sitecore Dynamic Link Management

Sitecore URLs do not correspond to files on the file system, but to items in Sitecore databases. Using URLs that correspond to data points (items) provides numerous advantages over using URLs that correspond to files. For example, data-driven URLs make it easy to share content between multiple devices, translate content into multiple languages, and reuse, update, and change presentation components. With dynamic URLs, you can include information in the URL path that might otherwise require a query string parameter. For example, you can specify a content language using the path prefix `/en` instead of the URL query string parameter `sc_lang=en`.

Items often reference other items, such as when you embed an image or a link in a field value. Because users can move and rename items, Sitecore references items using their unique IDs rather than their paths. Presentation components invoke the `renderField` pipeline, or use the `FieldRenderer` Web control, to transform IDs in each field value to the friendly URLs of the corresponding items.<sup>1</sup>

### 2.1.1 Dynamic Link Configuration

You can control friendly URLs by setting the following attributes of the `/configuration/sitecore/providers/add` element in `web.config` with name `sitecore`.

- **type**: You can override the link provider by setting the `type` attribute to the appropriate .NET class signature.<sup>2</sup>
- **addAspxExtension**: Whether to include the `.aspx` extension in URLs (`true` or `false`). If you set `addAspxExtension` to `false`, you must configure IIS to process all requests with ASP.NET as described in the section `IIS and ASP.NET URLs`.
- **alwaysIncludeServerUrl**: Whether to include the HTTP protocol and domain (`http://localhost`) in friendly URLs (`true` or `false`).
- **encodeNames**: Whether to encode names in paths according to the `/configuration/sitecore/encodeNameReplacements/replace` elements in `web.config` (`true` or `false`).
- **languageEmbedding**: Whether to include the language in the URL (`always`, `never`, or `asNeeded`). When `languageEmbedding` is `asNeeded`, Sitecore includes the language in the URL if it cannot determine the context site from the incoming HTTP request, if that HTTP request does not include a cookie that specifies a language, or if the language of the linked item differs from the context language.

#### Note

When `languageEmbedding` is `asNeeded`, Sitecore includes the language in the URL if it cannot determine the context site from the incoming HTTP request, if that HTTP request does not include a cookie that specifies a language, or if the language of the linked item differs from the context language.

- **languageLocation**: Whether to specify language as the first step in the URL path or using the `sc_lang` URL query string parameter (`filePath` or `queryString`).
- **useDisplayName**: Whether to use item display names or item names when constructing URLs (`true` or `false`). If the `useDisplayName` attribute is `true`, different languages can have different URLs for the same content item.

<sup>1</sup> For more information about the `RenderField` Web control, see the Presentation Component Reference at <http://sdn.sitecore.net/Reference/Sitecore%206/Presentation%20Component%20Reference.aspx>.

<sup>2</sup> For an example of a custom link provider, see <http://trac.sitecore.net/LinkProvider>.

**Important**

Consider the impact of configuration changes on other applications that rely on existing URLs, including Web analytics solutions and search engines.

**Note**

You may see other URL formats in Sitecore user interfaces, such as the raw values of a Rich Text Editor (RTE) fields. Presentation constructs transform such values into friendly URLs before transmitting markup to Web clients.

**Tip**

Before configuring link management, see the search engine optimization techniques and considerations described in the section Search Engine Optimized (SEO) URLs.

**Note**

For information about the effect of the `Rendering.SiteResolving` setting, see the section The `Rendering.SiteResolving` Setting.

## The `Rendering.SiteResolving` Setting

The `/configuration/sitecore/settings/setting` element in `web.config` with name `Rendering.SiteResolving` controls whether the link manager determines the hostname to include in URLs from the managed Web site definitions.

The link manager determines the managed Web site for each linked item. The managed Web site is the first define site definition that specifies a start item that is the linked item or one of its ancestors.

A single instance of Sitecore supports multiple managed Web sites. By default, administrators configure managed Web sites using `/configuration/sitecore/sites` elements in `web.config`. Among other properties, each `/configuration/sitecore/sites/site` element specifies a start item, which represents the home page for that managed Web site. Under the default configuration, the start item for the managed Web site named `website` is `/sitecore/content/home`.

```
<site name="website" virtualFolder="/" physicalFolder="/"
  rootPath="/sitecore/content" startItem="/home" ...
```

If the `Rendering.SiteResolving` setting is `false`, if the link manager cannot determine the managed Web site associated with the linked item, or if that managed Web site associated with the linked item is the context site, then the link manager uses the hostname in the current HTTP request, or does not include a host name or protocol depending on the value of the `alwaysIncludeServerUrl` attribute of the link manager. For more information about the `alwaysIncludeServerUrl` attribute of the link manager, see the section Dynamic Link Configuration.

If the `Rendering.SiteResolving` setting is `true`, and the dynamic link manager can determine a logical Web site for the linked item, and that site is not the context site, and the `targetHostName` attribute of that site has a value, then link manager uses the `targetHostName` attribute. If the `targetHostName` attribute has no value, and the `hostName` attribute has a value, and that value does not contain an asterisk character (“\*”) or a pipe character (“|”), then the link manager uses the `hostName` attribute.

Otherwise, the URL does not include a hostname or protocol.

**Warning**

The `Sitecore.Links.LinkManager.GetItemUrl()` method does not respect the `Rendering.SiteResolving` setting.<sup>3</sup>

<sup>3</sup> For an example custom link provider that always applies the `Rendering.SiteResolving` setting, see <http://trac.sitecore.net/LinkProvider>.

**Important**

All attributes are case-sensitive. The `hostName` attribute has an uppercase N.

**The LinkItemNotFound Setting**

The `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with name `LinkItemNotFoundUrl` controls the item to which Sitecore links when the item referenced by a link does not exist.

**2.1.2 How to Access the URL of a Content Item**

You can use the `Sitecore.Links.LinkManager.GetItemUrl()` method to access the URL of a content item.<sup>4</sup> For example, to access the URL of the context item:

```
Sitecore.Data.Items.Item item = Sitecore.Context.Item;
Sitecore.Links.UrlOptions urlOptions =
    (Sitecore.Links.UrlOptions) Sitecore.Links.UrlOptions.DefaultOptions.Clone();
urlOptions.SiteResolving = Sitecore.Configuration.Settings.Rendering.SiteResolving;
string url = Sitecore.Links.LinkManager.GetItemUrl(item, urlOptions);
```

**2.1.3 How to Access the URL of a Media Item**

You can use the `Sitecore.Resources.Media.MediaManager.GetMediaUrl()` method to access the URL of a media item. For example, to access the URL of the media item `/Sitecore/Media Library/Images/Sample` in the Master database:

```
Sitecore.Data.Database master = Sitecore.Configuration.Factory.GetDatabase("master");
Sitecore.Data.Items.Item sampleItem = master.GetItem(
    "/sitecore/media library/images/sample");
Sitecore.Data.Items.MediaItem sampleMedia = new Sitecore.Data.Items.MediaItem(sampleItem);
string url = Sitecore.StringUtil.EnsurePrefix(
    '/',
    Sitecore.Resources.Media.MediaManager.GetMediaUrl(sampleMedia));
```

**Warning**

Sitecore does not automatically include the leading slash character (“/”) in media URLs. This causes relative URLs for media items, which IIS resolves to the document root due to the tilde character (“~”). In solutions with very deep information architectures, relative media URLs can exceed limits imposed by the Web client or the Web server. Use the `Sitecore.StringUtil.EnsurePrefix()` method as shown in the previous example to ensure media URLs include the leading slash character.<sup>5</sup>

**Note**

There is no provider for media URLs.

You can use the `Sitecore.Resources.Media.MediaUrlOptions` class to specify media options when retrieving the URL of a media item. For example, to retrieve the URL of the thumbnail of the `/Sitecore/Media Library/Images/Sample` media item in the Master database:

```
Sitecore.Data.Database master = Sitecore.Configuration.Factory.GetDatabase("master");
Sitecore.Data.Items.Item sampleItem = master.GetItem(
    "/sitecore/media library/images/sample");
Sitecore.Data.Items.MediaItem sampleMedia =
    new Sitecore.Data.Items.MediaItem(sampleItem);
Sitecore.Resources.Media.MediaUrlOptions mediaOptions =
    new Sitecore.Resources.Media.MediaUrlOptions();
```

<sup>4</sup> For an example using the `Sitecore.Links.LinkManager.GetItemUrl()` method to access the URL of a content item, see the `Sitecore.Sharedsource.Data.Items.Item.GetUrlExtension` class described at <http://trac.sitecore.net/Library/>.

<sup>5</sup> For a solution to transform the prefix in media URLs consistently, see <http://trac.sitecore.net/LinkProvider/>.

```
mediaOptions.Thumbnail = true;  
string url = Sitecore.StringUtil.EnsurePrefix('/',  
    Sitecore.Resources.Media.MediaManager.GetMediaUrl(sampleMedia, mediaOptions));
```

## 2.2 Search Engine Optimized (SEO) URLs

You can improve search engine ranking by using Search Engine Optimized (SEO) URLs, such as by applying the following techniques:<sup>6</sup>

- Use hierarchies of words in the path to indicate topical categorization, such as `/humanresources/policies/` to represent a catalog of Human Resources policies.
- Avoid URL query string parameters.
- Avoid extensions such as `.aspx` and `.ashx` except for media, such as `.pdf` for PDF files.
- End URLs with a trailing slash character (“/”).
- Avoid multiple URLs for a single content or media item.

---

<sup>6</sup> For an example that applies these techniques, see <http://trac.sitecore.net/LinkProvider/>.

## 2.3 IIS and ASP.NET URLs

IIS responds to HTTP requests in one of three ways:

- IIS serves a file from disk.
- IIS invokes a process such as ASP.NET that may process a file from disk, respond with an error message, or return control to IIS.
- IIS handles an error by redirecting, displaying the contents of a file, or displaying a hard-coded error message.

By default, both IIS6 and IIS7 with a classic ASP.NET pipeline only use ASP.NET to process requests with file paths that end with specific extensions such as `.aspx` and `.ashx`. For requests that end with other extensions or no extension, IIS attempts to serve files from the document root or a subdirectory of the IIS Web site.

Because Sitecore is an ASP.NET application, when IIS does not use ASP.NET to process a request, Sitecore cannot process the request. This can lead to apparent inconsistencies, such as when IIS, ASP.NET, and Sitecore handle the HTTP 404 Page Not Found condition differently.

You can use one of the following techniques to configure IIS to use ASP.NET to process additional requests:

- IIS7 or Later Integrated Mode ASP.NET Managed Pipeline
- IIS HTTP 404 Page
- URL Rewriting ISAPI Filter
- IIS Wildcards

### Note

Configuring IIS to use ASP.NET to process additional requests can increase the attack surface and consume additional machine resources.

### 2.3.1 IIS7 or Later Integrated Mode ASP.NET Managed Pipeline

If you use IIS7 or later (Windows Vista, Windows 2008, Windows 7, or later) with Sitecore 6.2 or later, then you can configure the integrated mode of the ASP.NET managed pipeline to cause IIS to use ASP.NET to process all requests.

To configure integrated mode managed pipeline for the application pool associated with a Web site:

1. In the IIS management console, select the Web site, and then click **Basic Settings...** The **Edit Site** dialog appears.
1. In the **Edit Site** dialog, note the value of **Application pool**, and then click **Cancel**.
2. In the IIS management console, select **Application Pools**, and then double-click the application pool noted in the previous step. The **Edit Application Pool** dialog appears.
3. In the **Edit Application Pool** dialog, set **Managed pipeline mode** to **Integrated**, and then click **OK**.

### Note

If you use the classic mode of the ASP.NET managed pipeline, then see the section IIS7 or Later Classic Managed Pipeline Mode (Windows Vista, Windows 2008, and Windows 7).

### 2.3.2 IIS HTTP 404 Page

You can configure IIS to use ASP.NET to process additional requests that do not correspond to files by configuring the IIS HTTP 404 page to a URL that includes an ASP.NET extension such as `.aspx`,

such as `/default.aspx`.<sup>7</sup> If the URL of the IIS 404 page ends with an extension that would cause IIS to process the request with ASP.NET, IIS invokes ASP.NET to handle the request, whether or not the file exists.

#### Note

Update the parameters passed to the `FilterUrlExtensions` processor in the `preprocessRequest` pipeline defined in `web.config` to allow Sitecore to process requests with specific extensions. For example, to allow Sitecore to process requests with the `.htm` and `.html` extensions:

```
<processor
  type="Sitecore.Pipelines.PreprocessRequest.FilterUrlExtensions, Sitecore.Kernel">
  <param desc="Allowed extensions (comma separated)">aspx, ashx, asmx, htm,
html</param>
  ...
```

To allow Sitecore to process requests with any extension:

```
<processor
  type="Sitecore.Pipelines.PreprocessRequest.FilterUrlExtensions, Sitecore.Kernel">
  <param desc="Allowed extensions (comma separated)">*</param>
  <param desc="Blocked extensions (comma separated)"></param>
```

#### Note

Sitecore removes the extension from the URL before attempting to determine the context item. If you configure IIS to process requests with additional extensions or no extensions, whether the path in the URL is `/item.aspx`, `/item`, `/item/`, or `/item.html`, if the `/Sitecore/Content/Home/Item` item exists, Sitecore sets that item as the context item for this request.

To configure the IIS HTTP 404 page on Windows XP or Windows 2003:

1. In the Windows desktop, click the Windows Start button, and then click **Run**. The Windows **Run** dialog appears.
2. In the Windows **Run** dialog, enter `inetmgr`, and then click **OK**. The IIS management console appears.
3. In the IIS management console, right-click the machine to apply the change to all Web sites. Alternatively, expand both the machine name and **Web sites**, then right-click a Web site to apply the change to an individual Web site. Then click **Properties**. The **Web Sites Properties** dialog appears.
4. In the **Web Sites Properties** dialog, click the **Custom Errors** tab. For each of the 404 errors, click **Edit**, and then set **Message type** to `URL` and **URL** to `/default.aspx`.

To configure the IIS HTTP 404 page on Windows Vista or Windows 2008:

#### Warning

Changes through the IIS management console on some versions of Windows Vista may remove text values from `web.config`.<sup>8</sup>

#### Note

To use the HTTP 404 page on Windows Vista, install **Internet Information Services/World Wide Web Servers/Common HTTP Features/HTTP Errors**. To use the HTTP 404 page on Windows 2008, install **Web Server/Common HTTP Features/HTTP Errors**. To use the HTTP 404 page on Windows 7, install **World Wide Web Services/Common HTTP Features/HTTP Errors**.

<sup>7</sup> For more information about handling the HTTP 404 Page Not Found condition with Sitecore, see the guide to handling the HTTP 404 Page Not Found condition with Sitecore at <http://sdn.sitecore.net/Reference/Sitecore%206/Handling%20HTTP%20404.aspx>.

<sup>8</sup> For more information about the defect in IIS that can corrupt `web.config`, see <http://sdn.sitecore.net/Products/Sitecore%20V5/Sitecore%20CMS%205,-d-,3/Installation/Installing%20Sitecore%20on%20Vista.aspx>.

1. In the Windows desktop, click the Windows Start button. The Windows Start menu appears.
2. In the text field on the Windows Start menu, type `inetmgr`, and then press the `Enter` key. The IIS management console appears.
3. In the IIS management console, in the **Connections** tree, select the machine to apply the change to all Web sites. Alternatively, expand the machine and **Sites**, and then click an individual Web site to apply the change to an individual Web site.
4. In the IIS management console, under **IIS**, double-click **Error Pages**. A list of error codes appears.
5. In the list of error codes, double-click the **404** entry. The **Edit Custom Error Page** dialog appears.
6. In the **Edit Custom Error Page** dialog, select **Execute URL** or **Execute a URL on this site**, enter `/default.aspx` as the **Path** or **URL**, and then click **OK**.
7. In the IIS management console, in the **Actions** list, click **Edit Feature Settings**. The **Edit Error Pages Settings** dialog appears.
8. In the **Edit Error Pages Settings** dialog, select **Custom error pages**.

### 2.3.3 IIS Wildcards

Except in integrated mode, IIS configuration maps specific extensions, such as `.aspx`, `.ashx`, and `.asmx`, to an ISAPI filter that implements ASP.NET. You can configure IIS to process all requests with the ASP.NET ISAPI filter by configuring wildcard extensions.<sup>9</sup>

#### Important

Configuring IIS to use ASP.NET to process additional requests could have performance and security implications.

#### Important

For each file name extension that you use for Sitecore items including media items, it may be necessary to disable the option in the IIS management console that requires that the file exist.

#### Important

You may need to configure the `StaticFileHandler` in `web.config` for each file name extension that you use for both files and Sitecore items. For example, to process requests with the `.htm` extension as either Sitecore items or static files:

```
<httpHandlers>
  <add verb="GET,HEAD" path="*.htm" type="System.Web.StaticFileHandler, System.Web,
    Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
  ...
```

### IIS 5.1 or Earlier (Windows XP)

To configure wildcard extension processing on IIS (Windows XP):

1. In the IIS management console, right-click the Web site, and then click **Properties**. The **Web Site Properties** dialog appears.
2. In the **Web Site Properties** dialog, click the **Home Directory** tab, and then click **Configuration**. The **Application Configuration Dialog** appears.
3. In the **Application Configuration** dialog, select the `.aspx` entry, and then click **Edit**. The **Add/Edit Application Extension Mapping** dialog appears.

<sup>9</sup> For more information about wildcard extensions, see <http://professionalaspnet.com/archive/2007/07/27/Configure-IIS-for-Wildcard-Extensions-in-ASP.NET.aspx>.

4. In the **Add/Edit Application Extension Mapping** dialog, copy **Executable** to the Windows clipboard, and then click **Cancel**.
5. In the **Application Configuration** dialog, click **Add**. The **Add/Edit Application Extension Mapping** dialog appears.
6. In the **Add/Edit Application Extension Mapping** dialog, for **Executable**, paste the value from the Windows clipboard.
7. In the **Add/Edit Application Extension Mapping** dialog, for **Extension**, enter a dot character followed by a star character (“ . \* ”).
8. In the **Add/Edit Application Extension Mapping** dialog, select **Limit to**, and enter **GET, POST**.
9. In the **Add/Edit Application Extension Mapping** dialog, disable **Check that file exists**, and then click **OK**.

## IIS6 (Windows 2003)

To configure wildcard extension processing on IIS6 (Windows 2003):

1. In the IIS management console, right-click the Web site, and then click **Properties**. The **Web Site Properties** dialog appears.
2. In the **Web Site Properties** dialog, click the **Home Directory** tab, and then click **Configuration**. The **Application Configuration** dialog appears.
3. In the **Application Configuration** dialog, click the **Mappings** tab, select the **.aspx** entry, and then click **Edit**. The **Add/Edit Application Extension Mapping** dialog appears.
4. In the **Add/Edit Application Extension Mapping** dialog, copy **Executable** to the Windows clipboard, and then click **Cancel**.
5. In the **Application Configuration** dialog, click **Add**. The **Add/Edit Application Extension Mapping** dialog appears.
6. In the **Add/Edit Application Extension Mapping** dialog, for **Executable**, paste the value from the Windows clipboard.
7. In the **Add/Edit Application Extension Mapping** dialog, for **Extension**, enter an asterisk character (“\*”).
8. In the **Add/Edit Application Extension Mapping** dialog, select **Limit to**, and enter **GET, POST**.
9. In the **Add/Edit Application Extension Mapping** dialog, disable **Verify that file exists**, and then click **OK**.

## IIS7 or Later Classic Managed Pipeline Mode (Windows Vista, Windows 2008, and Windows 7)

To configure wildcard extension processing on IIS7 or later using an application pool with the classic managed pipeline mode:

### Important

To use IIS wildcards on Microsoft Windows Vista, install **World Wide Web Services/Application Development Features/ISAPI Extensions**. To use IIS wildcards on Microsoft Windows 2008, install **Web Server/Application Development/ISAPI Extensions** and **Web Server/Application Development/ISAPI Filters**. To use IIS wildcards on Windows 7, install **Internet Information Services/World Wide Web Services/Application Development Features/ISAPI Extensions** and **Internet Information Services/World Wide Web Services/Application Development Features/ISAPI Filters**.

1. In the IIS management console, click the Web site, then double-click **Handler Mappings**, then select the `.aspx` entry, and then click **Edit**. The **Edit Script Map** dialog appears.
2. In the **Edit Script Map** dialog, copy **Executable** to the Windows clipboard, and then click **Cancel**.
3. Click **Add Script Map...** The **Add Script Map** dialog appears.
4. In the **Add Script Map** dialog, for **Request path**, enter an asterisk character (“\*”).
5. In the **Add Script Map** dialog, for **Executable**, paste the value from the Windows clipboard.
6. In the **Add Script Map** dialog, in the **Name** field, enter a name for the handler mapping, such as `Wildcard ASP.NET ISAPI`.
7. In the **Add Script Map** dialog, click **Request Restrictions**. The **Request Restrictions** dialog appears.
8. In the **Request Restrictions** dialog, click the **Mapping** tab, and then disable **Invoke handler only if the request is mapped to**.
9. In the **Request Restrictions** dialog, click the **Verbs** tab, then select **One of the following verbs**, then enter `GET, POST`, and then click **OK**.

### 2.3.4 URL Rewriting ISAPI Filter

You can configure IIS to use ASP.NET to process additional requests by implementing an ISAPI filter to rewrite URLs before IIS determines how to process them.<sup>10</sup>

**Note**

This document does not describe URL rewriting with ISAPI filters.

---

<sup>10</sup> For more information about using ISAPI filters to rewrite URLs, see <http://sdn.sitecore.net/Scrapbook/Friendlier%20Marketing%20URLs.aspx>.