Sitecore CMS 6

Sitecore Dynamic Links

A Developer's Guide to Constructing URLs with Sitecore

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 © 2001-2015 Sitecore. All rights reserved.

Table of Contents

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 © 2001-2015 Sitecore. All rights reserved.

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 and other link management features.

This document contains the following chapters:

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

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 © 2001-2015 Sitecore. All rights reserved.

Chapter 2

Sitecore Dynamic Links

This chapter describes how IIS and ASP.NET process URLs, how Sitecore generates URLs dynamically, and Search Engine Optimization (SEO) techniques that you can use with Sitecore.

This chapter contains the following sections:

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 © 2001-2015 Sitecore. All rights reserved.

2.1 IIS and ASP.NET URLs

IIS responds to HTTP requests in one of three ways:

- IIS serves a file from disk.
- By invoking a process such as ASP.NET that may process a file from disk, respond with an error message, or return control to IIS.
- IIS responds 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 any of the following techniques to configure IIS to use ASP.NET to process additional requests:

- IIS HTTP 404 Page
- URL Rewriting ISAPI Filter
- IIS Wildcards

Note
Process additional requests using ASP.NET may consume additional machine resources.

2.1.1 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`. 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
You must 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

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 © 2001-2015 Sitecore. All rights reserved.

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 Start button, and then click Run. The Run dialog appears.
2. In the 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, or expand both the machine and Web sites, then right-click a Web site to apply the change to an individual Web site, and then click Properties. The Web Sites Properties dialog appears.
4. 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`.[1]

Note
On Windows Vista, you must install Internet Information Services/World Wide Web Servers/Common HTTP Features/HTTP Errors. On Windows 2008, you must install Web Server/Common HTTP Features/HTTP Errors.

1. In the Windows desktop, click the 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, or 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 Actions list in the IIS management console, click Edit Feature Settings. The Edit Error Pages Settings dialog appears.
8. In the Edit Error Pages Settings dialog, select Custom error pages.

2.1.2 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.[2]

[1] For more information about the defect in IIS that can corrupt `web.config`, see http://sdn5.sitecore.net/Products/Sitecore%20V5/Sitecore%20CMS%205,-d-,3/Installation/Installing%20Sitecore%20on%20Vista.aspx.
[2] For more information about wildcard extensions, see http://professionalaspnet.com/archive/2007/07/27/Configure-IIS-for-Wildcard-Extensions-in-ASP.NET.aspx.

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 © 2001-2015 Sitecore. All rights reserved.

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" />
  ...
```

Windows XP and Windows 2000 Server (IIS5)

To configure wildcard processing on II5 (Windows XP or Windows 2000):

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.
4. In the Add/Edit Application Extension Mapping dialog, copy the value of the Executable field 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, in the Executable field, paste the value from the Windows clipboard, or enter the equivalent of the following:

```
C:\windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll
```

7. In the Add/Edit Application Extension Mapping dialog, set Extension to a period character followed by a star character (“`.*`”).
8. In the Add/Edit Application Extension Mapping dialog, select All verbs.
9. In the Add/Edit Application Extension Mapping dialog, clear the Check that file exists checkbox, and then click OK.

Windows 2003 (IIS6)

To configure wildcard processing on II6 (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 the value of the Executable field to the Windows clipboard, and then click Cancel.

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 © 2001-2015 Sitecore. All rights reserved.

5. In the Application Configuration dialog, click Insert. The Add/Edit Application Extension Mapping dialog appears.
6. In the Add/Edit Application Extension Mapping dialog, in the Executable field, paste the value from the Windows clipboard, or enter the equivalent of the following:

```
C:\windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll
```

7. In the Add/Edit Application Extension Mapping dialog, clear the Check that file exists checkbox, and then click OK.

Windows Vista and Windows 2008 (IIS7)

To configure wildcard processing on IIS7 with a classic application pool:

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 the value of the Executable field to the Windows clipboard, and hten click Cancel.
3. Click Add Script Map… The Add Script Map dialog appears.
4. In the Add Script Map dialog, in the Executable field, paste the value from the Windows clibboard, or enter the equivalent of the following:

```
%windir%\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll
```

5. In the Add Script Map dialog, in the Request path field, enter a star character (“`*`”).
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 clear the Invoke handler only if the request is mapped to checkbox.
9. In the Request Restrictions dialog, click the Verbs tab, then select All Verbs, and then click OK.

Important
To use this solution on Windows 2008, you must install Web Server/Application Development/ISAPI Extensions and Web Server/Application Development/ISAPI Filters. To use this solution on Windows Vista, you must install World Wide Web Services/Application Development Features/ISAPI Extensions.

2.1.3 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.[3]

[3] For more information about using ISAPI filters to rewrite URLs, see http://sdn.sitecore.net/Scrapbook/Friendlier%20Marketing%20URLs.aspx.

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 © 2001-2015 Sitecore. All rights reserved.

2.2 Sitecore Dynamic Link Management

Sitecore URLs do not correspond to files on disk, but to items in a Sitecore database. Items contain layout details, which specify the presentation components used to service requests for the item from different types of devices.

URLs that correspond to data items provide numerous advantages over URLs that correspond to files, such as simplifying the ability to share content between multiple devices, translate content into multiple languages, and reuse, update, and change presentation components at any time. With dynamic URLs, you can ihclude information into the path that might otherwise require a query string parameter. For example, you can specify a content language using the path prefix `/en` instead or the URL query string parameter `sc_lang=en`.

2.2.1 Dynamic Link Configuration

You can configure the following attributes of the `/configuration/sitecore/providers/add` element in `web.config` with `name sitecore` to control how Sitecore generates URLs:

- **`type`**: You can override the link provider by updating the `type` attribute (.NET class signature).
- **`addAspxExtension`**: Whether to include the `.aspx` extension in URLs (`true` or `false`). If you set `addAspxExtension` to `false`, you muse 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.
- **`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.

Important
Consider the impact of URL configuration changes on other applications that rely on URLs, such as Web analytics solutions.

Note
You may see other URL formats in content management user interfaces, such as the raw value of a Rich Text Editor (RTE) fields. Presentation constructs transform these links to friendly URLs before transmitting markup to user agents.

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

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 © 2001-2015 Sitecore. All rights reserved.

The Rendering.SiteResolving Setting

A single instance of Sitecore supports multiple logical Web sites. By default, administrators configure multiple logical Web sites within the `/configuration/sitecore/sites` element in `web.config`. Each `/configuration/sitecore/sites/site` element specifies a start item for a Web site, which represents the home page for that site.

The `/configuration/sitecore/settings/setting` element in `web.config` with `name Rendering.SiteResolving` controls whether the renderField pipeline and XSL constructs determine the hostname to include in URLs by matching the path to the linked item with the logical site definitions. The `Sitecore.Links.LinkManager.GetItemUrl()` method does not respect the value of the `Rendering.SiteResolving` setting.

If the `alwaysIncludeServerUrl` attribute of the `/configuration/sitecore/providers/add` element in `web.config` with `name sitecore` is True, then the link manager determines the hostname from the first defined logical Web site with attributes that match the path to the linked item.

If the `Rendering.SiteResolving` setting is False, if the logical Web site associated with the linked item is undefined, or if that logical Web site is the context site, then the hostname in the friendly URL is the hostname in the current HTTP request.

If the `Rendering.SiteResolving` setting is `true`, and the dynamic link manager can determine a logical Web site for the item, and that site is not the context site, and the `targetHostName` attribute of that site has a value, then the hostname in the friendly URL is the `targetHostName` attribute of that site. If the `targetHostName` attribute has no value, and the `hostName` attribute has a value, and that value does not contain a star ("`*`") or pipe character ("`|`"), then the hostname in the friendly URL is the value of that `hostName` attribute. Otherwise, the URL does not include a hostname or protocol.

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

You can implement code such as the following to apply the `Rendering.SiteResolving` setting:[4]

```
        Sitecore.Data.Database master = Sitecore.Configuration.Factory.GetDatabase("master");
        Sitecore.Data.Items.Item home = master.GetItem("/sitecore/content/home");
        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(home,urlOptions);
```

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.

2.2.2 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.[5] For example, to access the URL of the context item:

```
        Sitecore.Data.Items.Item item = Sitecore.Context.Item;
        Sitecore.Links.UrlOptions urlOptions =
```

[4] For a solution that applies the `Rendering.SiteResolving` setting consistently, see https://marketplace.sitecore.net/Modules/Link_Provider.aspx.
[5] 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 https://marketplace.sitecore.net.

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 © 2001-2015 Sitecore. All rights reserved.

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

2.2.3 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.Item 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.[6]

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

[6] For a solution to transform the prefix in media URLs consistently, see https://marketplace.sitecore.net/Modules/Link_Provider.aspx.

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 © 2001-2015 Sitecore. All rights reserved.

2.3 Search Engine Optimized (SEO) URLs

You can iimprove search engine ranking by using Search Engine Optimized (SEO) URLs, such as by applying the following techniques:[7]

- 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.

[7] For an example that applies these techniques, see https://marketplace.sitecore.net/Modules/Link_Provider.aspx.

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 © 2001-2015 Sitecore. All rights reserved.