Sitecore CMS 6.0 or later

Handling HTTP 404

A Developer's Guide to the HTTP 404 Page Not Found Condition with Sitecore

Table of Contents

Chapter 1

Introduction

This document provides information about how Microsoft Internet Information Server (IIS), the ASP.NET application server, and the Sitecore Web Content Management System (CMS) handle the HTTP 404 Page Not Found condition. Sitecore developers can use this information to handle the HTTP 404 Page Not Found condition consistently, or for specific purposes such as providing search results.

This document contains the following chapters:

- Chapter 1 – Introduction
- Chapter 2 – The HTTP 404 Page Not Found Condition

Chapter 2

The HTTP 404 Page Not Found Condition

This chapter provides information about how Microsoft Internet Information Server (IIS), the ASP.NET application server, and Sitecore handle the HTTP 404 Page Not Found condition.

This chapter contains the following sections:

- Overview of the HTTP 404 Page Not Found Condition
- Consistent HTTP 404 Page Not Found Management
- HTTP 404 Page Not Found Pipeline Processor

2.1 Overview of the HTTP 404 Page Not Found Condition

The HTTP 404 Page Not Found condition occurs when an HTTP request does not correspond to a resource on the Web server. The Web server responds to a successful HTTP request with the HTTP 200 Success status code and a content payload. The Web server responds to the HTTP 404 Page Not Found condition the HTTP 404 Page Not Found status code and markup providing a friendly error message known as the 404 page. Web solutions can handle the 404 condition and use the 404 page for a variety of purposes, such as to redirect or render search results instead of returning the HTTP 404 status code and the 404 page.

Important
In certain configurations, IIS does not automatically use ASP.NET to process all requests.[1] Sitecore cannot process requests that IIS does not process using ASP.NET.

2.1.1 IIS HTTP 404 Page Not Found Management

The IIS Web server responds to each HTTP request by either serving a file from disk or using a process such as the ASP.NET application server to process the request. If that process does not manage its own errors, error management reverts to IIS.

If IIS does not use ASP.NET to process a request, then IIS attempts to serve a file from disk. If the file does not exist, then IIS activates the 404 page defined through the IIS management console.

If IIS uses ASP.NET to process an HTTP request, then Sitecore intercepts the request and invokes the `httpRequestBegin` pipeline defined in `web.config`. If a processor aborts the `httpRequestBegin` pipeline, then ASP.NET processes the request as a request for a standalone ASP.NET page rather than as a request for a Sitecore item.

If IIS processes a request as a standalone ASP.NET page and no file corresponding to the URL exists, and the `mode` attribute of the `/configuration/system.web/customErrors/configuration` element in `web.config` is `On`, or the `mode` attribute is `RemoteOnly` and the request did not originate locally, then IIS redirects the user agent to the URL specified by the `redirect` attribute of the `/configuration/system.web/customErrors/error` element in `web.config` with `statusCode` of `404`. If no such element exists, then IIS redirects to Web client to the URL specified by the `defaultRedirect` attribute of the `/configuration/system.web/customErrors` element in `web.config`:

```
<customErrors mode="RemoteOnly" defaultRedirect="/errors/default.html">
  <error statusCode="404" redirect="/errors/404.html" />
</customErrors>
```

If the `defaultRedirect` attribute is absent, then IIS responds to the HTTP request with a hard-coded HTML error message.

2.1.2 Sitecore HTTP 404 Page Not Found Management

If IIS uses ASP.NET to process a request, and no processor aborts the `httpRequestBegin` pipeline, and the requested URL does not correspond to a Sitecore media item, a content item within the context site, or a file on disk, then Sitecore activates the URL specified by the `value` attribute of the

[1] For instructions to configure IIS to use ASP.NET to process additional types of requests, see http://sdn.sitecore.net/Reference/Sitecore%206/Dynamic%20Links.aspx.

`/configuration/sitecore/settings/setting` element in `web.config` with `name ItemNotFoundUrl`.

The LinkItemNotFound Setting

If a Rich Text Editor (RTE) field contains a link to an item that does not exist, Sitecore renders that link as a link to the URL specified by the `value` attribute of the `/configuration/sitecore/settings/setting` element in `web.config` with `name LinkItemNotFound`.

Analytics Features of the default NotFound.aspx

The Sitecore default page, `/sitecore/service/notfound.aspx`, is closely related to OMS analytics reports that are available through **Sitecore Analytics** -> **Reports** -> **Site Health**.

So, if you want to override the page, which is defined by the `ItemNotFoundUrl` and `LinkItemNotFound` options, please pay your attention to the following tips:

- The `Not Found Urls` report scans the analytics database for requests that have a URL such as `%/NotFound.aspx%`. So, if your own file has a different page name, modify the SQL query of this report:
    - Sitecore 6.2-6.4: Open the `\Website\sitecore\shell\Applications\Analytics\Reports\NotFound.mrt` file and replace `%/NotFound.aspx%` with your custom URL.
    - Sitecore 6.5 or later: Open the **Content Editor**, locate the item `/sitecore/system/Settings/Analytics/Reports SQL Queries/Not Found Urls` and replace the `%/NotFound.aspx%` with your custom URL in the SQL Server field.

  For example, if your file is named `My404.aspx`, you should replace `%/NotFound.aspx%` with `%/My404.aspx%`.
- The 'Latest Failures', 'Common Mistakes' and 'Pages that report errors' reports scan the analytics database for the specific default analytics page events – `PageNotFound`. The default `notfound.aspx` triggers this event, so your own custom page should trigger this event too to update the analytics data.

For Sitecore 6.2 to 6.4, use the following code in the code behind of your page to call the `PageNotFound` events:

```
        using Sitecore.Analytics.Extensions.AnalyticsPageExtensions;
        …
        protected override void OnPreRender(EventArgs e)
{
         base.OnPreRender(e);
         base.Response.StatusCode = 0x194;
         base.Response.StatusDescription = "Not Found";
         Sitecore.Analytics.AnalyticsTracker current =
Sitecore.Analytics.AnalyticsTracker.Current;
         string str = Sitecore.MainUtil.DecodeName(Sitecore.StringUtil.GetString(new string[]
{base.Request.QueryString["item"], "[unknown]" }));
         if (current != null)
         {
           current.CurrentPage.PageNotFound(Sitecore.Web.WebUtil.SafeEncode(str));
         }
}
```

For Sitecore 6.5 or later + DMS 2 or later, use the following code in the code behind of your page to call the `PageNotFound` events:

```
        using Sitecore.Analytics.Extensions;
        ...
        protected override void OnPreRender(EventArgs e)
{
         base.OnPreRender(e);
         base.Response.StatusCode = 0x194;
         base.Response.StatusDescription = "Not Found";
         string str = Sitecore.MainUtil.DecodeName(Sitecore.StringUtil.GetString(new string[]
{base.Request.QueryString["item"], "[unknown]" }));
         if (Sitecore.Analytics.Configuration.AnalyticsSettings.Enabled)
         {
         Sitecore.Analytics.Tracker.CurrentPage.RegisterPageNotFound(Sitecore.Web.WebUtil.SafeEnco
de(str));
         }
}
```

2.2 Consistent HTTP 404 Page Not Found Management

To manage the HTTP 404 Page Not Found condition consistently, you can set all of the following to the same value:

- The IIS 404 page set through the IIS management console (optionally set to an alternate value such as `/default.aspx` to cause IIS to process all requests with ASP.NET and hence Sitecore).
- The ASP.NET 404 page in the `/configuration/system.web/customErrors` section of `web.config`.
- The `ItemNotFoundUrl` setting in `web.config`.
- The `LinkItemNotFound` setting in `web.config`.

Tip
For search engine optimization, you should return the HTTP 404 status code for any invalid URL by setting the `Status` property of the current `System.Web.HttpResponse`. To set the status code in the HTTP response from the code-behind file of a layout or sublayout, `this.Response.StatusCode`. To set the status code in the HTTP response from a Web control, set `this.Page.Response.StatusCode`. You can also set the `Status` and `StatusDescription` properties of this `System.Web.HttpResponse` object.[2]

To configure Sitecore to handle the HTTP 404 Page Not Found condition consistently when Sitecore is configured not to use extensions for requested pages, that is when the LinkManager is configured with the `addAspxExtension` attribute set to `false`, follow the previous instructions for configuring your `customErrors` settings and then refer to Section 2.3 of the Sitecore Dynamic Links[3] reference for instructions on how to configure IIS.

[2] For more information about the `System.Web.HttpResponse` class, see http://msdn.microsoft.com/en-us/library/system.web.httpresponse.aspx.
[3] For more information about configuring IIS 404 page, see http://sdn.sitecore.net/Reference/Sitecore%206/Dynamic%20Links.aspx.

2.3 HTTP 404 Page Not Found Pipeline Processor

Instead of using a 404 page to handle the HTTP 404 Page Not Found condition, you can use a `httpRequestBegin` pipeline processor. For example, instead of returning a 404 page, you can log the 404 request and the referring page and redirect from an old URL to new URLs, or provide search results instead of the 404 page. By handling the HTTP 404 Page Not Found condition in the `httpRequestBegin` pipeline, you can often avoid the overhead of the additional HTTP request and response generated by a redirect.[4]

To implement an HTTP 404 Page Not Found `httpRequstBegin` pipeline processor:

1. Create a class that inherits from Sitecore.Pipelines.HttpRequest.HttpRequestProcessor.
2. Implement the `Process()` method (in most cases, the processor should not take action if the context item is known or either the context site or database is unknown):

```
namespace Namespace.Pipelines.HttpRequest
{
  public class NotFoundProcessor :
    Sitecore.Pipelines.HttpRequest.HttpRequestProcessor
  {
    public override void Process(
      Sitecore.Pipelines.HttpRequest.HttpRequestArgs args)
    {
      if (Sitecore.Context.Item != null
        || Sitecore.Context.Site == null
        || Sitecore.Context.Database == null)
      {
        return;
      }

      // TODO: logic, such as to set Sitecore.Context.Item
      // based on Sitecore.Context.Request.FilePath
    }
  }
}
```

3. Insert the processor after the default item resolver `<processor>` in the `/configuration/sitecore/pipelines/httpRequestBegin` pipeline in web.config:

```
<processor type="Sitecore.Pipelines.HttpRequest.ItemResolver, Sitecore.Kernel" />
<processor type="Namespace.Pipelines.HttpRequest.NotFoundProcessor, Assembly" />
...
```

[4] For example pipeline processors that handle the HTTP 404 Page Not Found condition, see http://trac.sitecore.net/PageNotFound/.