

Module 5: Optimizing Performance for Web Content Management

Date published:

February 2009

Summary:

This paper presents measures that you can take to ensure that your authors, editors, developers, and administrators maintain an optimized system and teaches you how to use Internet Information Services (IIS) compression and caching to deliver content to users rapidly.

See [Web Content Management Training Modules](http://go.microsoft.com/fwlink/?LinkId=141931) (http://go.microsoft.com/fwlink/?LinkId=141931) for a complete list of the available downloads.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, JScript, SharePoint, Visual Basic, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

Module 5: Optimizing Performance for Web Content Management 1

Module 5 Overview 1

Introduction 1

Objectives 1

Lesson 1: Optimizing Content 1

Objectives 1

Microsoft Office SharePoint Server 2007 Core Files 1

The Purpose of Office SharePoint Server 2007 Core Files 1

Best Practices for Optimizing the Download of Core Files 2

Optimizing HTML 2

Optimizing CSS 4

Optimizing Custom Controls and Code 6

Considerations for Optimized Custom Code 6

Ensuring that Developers Optimize Their Code 7

Lesson 2: Network Security 7

IIS Compression 8

How IIS Compression Improves Performance 8

When to Use IIS Compression 8

Setting Compression Options 8

Compression Levels 9

Site-Specific Compression 10

Output Caching 10

Configuring Site Output Caching 11

Page Layout Caching 11

Post-Cache Substitution 12

Disk-based Caching for BLOBs 12

Office SharePoint Publishing Cache Counter 13

Other Caching Mechanisms 14

Retiming the IIS Object Cache 14

Web Part Caching 15

ASP.NET Cache Object 15

Web Part Cache in Memory or SQL Server 15

The Caching Application Block 16

Review of Module 5 16

Module 5 Overview

Introduction

It is critical to ensure that your Web site responds rapidly to user requests. If users are kept waiting for a page, they frequently abandon your site and look elsewhere, and you will lose potential business. Although Microsoft® Office SharePoint® Products and Technologies are optimized out-of-the-box, there are several measures that you can take to ensure that your authors, editors, developers, and administrators maintain an optimized system. This module describes those measures.

Objectives

After completing this module, you will be able to:

* Ensure that authors, editors, and developers create optimized content for a Web Content Management (WCM) system ([Lesson 1](#Lesson1)).
* Use Internet Information Services (IIS) compression and caching to deliver content to users rapidly ([Lesson 2](#Lesson2)).

Lesson 1: Optimizing Content

The content that authors and editors create can introduce delays into the page-rendering process. This lesson describes the issues you should guard against in your content development process. The lesson also highlights common issues in custom code that slow content delivery.

Objectives

After completing this lesson, you will be able to:

* [Describe the Office SharePoint core files and decide whether they delay content delivery](#SharePointServer2007CoreFiles)
* [Ensure that your authors create optimal HTML in their articles](#OptimizingHTML)
* [Ensure that your designers create style sheets with optimal performance](#OptimizingCSS)
* [Prevent custom code with slow performance from reaching your production Web farm](#OptimizingCustomControlsandCode)

Microsoft Office SharePoint Server 2007 Core Files

Microsoft® Office SharePoint® Server 2007 provides a selection of core files that are used as central stores for common reusable features of your sites. These core files can be large and can add approximately 500 kilobytes (KB) to the size of a page. This can pose a problem in that your users will experience increased download times. One of the most important factors for optimizing your deployment is to manage this page payload and reduce the overall download size for your pages. It is common for the load added by the core files to cause download times upward of 10 to 20 seconds on slower connections.

This section contains the following topics:

* [The Purpose of Office SharePoint Server 2007 Core Files](#PurposeofOfficeSharePointServer)
* [Best Practices for Optimizing the Download of Core Files](#BestPracticesforOptimizingDownload)

The Purpose of Office SharePoint Server 2007 Core Files

The core files provided with Office SharePoint Server 2007 provide many of the common features that you see across your sites. This includes the style sheets that affect the entire site and the JavaScript functions that are used across multiple sites.

The following table illustrates the main core files that are added to your page download by Office SharePoint Server 2007 when they are retrieved and provides a brief description of each.

|  |  |
| --- | --- |
| **Core File** | **Description** |
| core.js | A JavaScript file that provides Office SharePoint Server 2007 functions. For example, the Site Actions menu that authenticated users see is built by this file. |
| core.css | A cascading style sheet (CSS) file that provides the look and feel for the common Office SharePoint Server 2007 controls, such as the top horizontal navigation bar. |
| init.js | Internal users to Web server that hosts the Central Administration Web site |
| IE55up.js, IE50up.js, and Nonle.js | Web servers to Query servers (search requests) |

Best Practices for Optimizing the Download of Core Files

Because the core files that Office SharePoint Server 2007 adds to your pages can cause a performance hit on first use (known as PLT1, or the first time a page is downloaded by the user), you might want to optimize the download of these files. The following list provides guidance on possible optimizations that you should consider before you deploy your sites.

* Delay the load of the core.js file.

Loads in the background for anonymous users.

* Load the core.js file only in cases when it is required.

Loads conditionally for authenticated users.

Note: For more information about loading the core.js file in the background, see [How to create a detached page that downloads the Core.js file but that does not reference the Core.js file on a SharePoint Server 2007 site](http://go.microsoft.com/fwlink/?LinkId=140157&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140157&clcid=0x409).

The core.js file provides many aspects of an Office SharePoint Server 2007 site’s functionality. In many Internet-facing sites, you might not require such features; however, the authoring and administration modes might require some of the provided features. In this situation, you might use one of the above best practices to prevent the load of the core.js file unless the user is authenticated for access to the required functionality.

Optimizing HTML

There are many standards and guidelines to which you should adhere when you develop your HTML content. Your Web developers must be aware of these standards because they enable you to perform optimization of your HTML code.

The following table illustrates techniques that your Web developers can use to optimize the HTML that they produce for your sites.

|  |  |
| --- | --- |
| **Best Practice** | **Guidance** |
| Documents must validate as HTML. | The World Wide Web Consortium (W3C) develops standards and guidelines for numerous interoperable technologies. You can view the HTML specification created for HTML on the [W3C site](http://go.microsoft.com/fwlink/?LinkId=140159&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140159&clcid=0x409).  When you have created your HTML, you can also validate it by using the W3C [Markup Validation Service](http://go.microsoft.com/fwlink/?LinkId=140161&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140161&clcid=0x409).  Note: Extensible Hypertext Markup Language (XHTML) is not supported by default in Office SharePoint Server 2007. |
| Documents must have a Document Type Definition (DTD). | A DTD contains the description of a page’s format. Browsers use the DTD to determine the type of HTML that is used on the page. This will be valuable when you want to validate your HTML because the validator will use the DTD to determine which type of HTML to validate against. By including a DTD with your pages, you enable browsers to determine the type of HTML that you use and therefore understand and display the page at a faster rate with fewer unexpected problems. |
| Documents must have character encoding. | You should use an encoding that covers as many of the characters in your document as possible. There might be characters that are not part of this encoding, but you can still represent them by using their character references. Most Western European languages use **"Latin-1."** |
| Documents must define a language. | You should declare the language in which the page is written, for example, “**en”** for English. This will optimize your pages for globalization. For example, specifying the language in HTML is useful for language-specific searches, aural browsers, translation tools, and print style sheets. You should take this into consideration as you design the pages for your site. |
| Attribute values should be delimited with quotation marks. | Although this might not have a discernible effect on performance, you should do this as a matter of good practice. |
| Attribute and Element names should use a consistent case. | The use of a consistent case enhances the readability of your code. You should ensure your Web developers follow the casing rules set out for the project. Improving the readability of the code will reduce the time it takes for modifications to be made to your HTML at a later date, and will reduce confusion during development time.  Note: If you require compatibility with XHTML, you use lowercase characters. |
| Images, image maps, and scripted content must have the alt-attribute. | Use the **alt** attribute to provide alternate text that provides content for browsers that do not support forms, people who are visually impaired, and users of speech synthesizers. To provide a high level of accessibility and prevent user frustration, the following should be observed:   * Do not use alternate text for Images intended for page formatting. * Do not use meaningless alternate text.   Following these two rules will reduce the time required for clients to process the page and thus improve performance. |
| Images should specify width and height. | These attributes should be included in your HTML, because when the width and height of an image have been defined, the client is aware of the required space for that image and can reserve the space while it retrieves the image data and renders the rest of the document. This improves load times for the document and reduces user wait times. |
| Some special characters must be replaced by their character entities or numeric character references. | The W3C Architecture domain recommends that special characters such as **“**, **<**, **>**, and **&** be replaced by character entities or numeric character references (NCRs). This is especially true when you use these characters within attributes such as **href** and **src**.  If you use these characters in your HTML, it is advised that you use **"&quot"**, **"&lt"**, **"&gt"**, and **"&amp"** respectively.  By using **"&lt"** and **"&gt"** you will reduce the chances of confusion because the **<** or **>** can get mistaken for the start or end of a tag. |
| Characters not included in the character set must be replaced. | The W3C Architecture domain recommends that characters not included in a character set, such as the € character, must be replaced. There is no recommendation to replace language-specific characters such as é. The purpose of this replacement is to provide readable code. |
| Script and style tags must have a **type** attribute. | The W3C states that script and style tags must have a **type** attribute. This specifies the language of the contents and overrides the default scripting language. The benefit of this is that the client knows in advance that the content is of a given type, for example "text/javascript." This can reduce load times and errors. |
| ECMAScript code should be written and included in files external to the HTML. | ECMAScript such as JavaScript or Microsoft JScript® should be written externally to the main HTML page. This enables you to provide reusable scripts that can be obtained and cached to reduce the number of hits on a server for the given functionality of the script. This also reduces the amount of redundant code located within the HTML page. |
| Use and include central, nonredundant files that can be cached by the browser. | By creating scripts as external files, you can remove redundancy from your code and provide reusable files that can be downloaded and stored in the cache of the browser. This will improve overall request time and performance. |

Optimizing CSS

When you design the content for your sites, you must consider the styles that are applied to the content. This is managed by using cascading style sheets (CSS). When you design CSSs, you will often discover inefficiencies that affect performance. It is possible to optimize your CSS files by following a series of best practices. The following table provides best practices that your CSS developers should follow in order to produce optimized style sheets.

|  |  |
| --- | --- |
| **Best Practice** | **Guidance** |
| CSS must validate. | The W3C provides specifications for CSS. To view these specifications, see [CSS SPECIFICATIONS](http://go.microsoft.com/fwlink/?LinkId=140162&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140162&clcid=0x409).  To help optimize your sites, you should follow the guidance provided by the W3C and then perform validations to ensure that your CSS complies with the guidelines.  Note: For more information see the [CSS Validation Service](http://go.microsoft.com/fwlink/?LinkId=140166&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140166&clcid=0x409). |
| Site elements should be positioned with CSS 2. | The W3C has created three CSS levels: 1, 2, and 3. CSS level 2, or CSS 2, adds absolute positioning for elements. For more information about CSS 2, see [Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification](http://go.microsoft.com/fwlink/?LinkId=140168&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140168&clcid=0x409). |
| Decorative site elements should be rendered with CSS 2. | Site elements such as lines, borders, background colors and images, and underlines are covered in the W3C specification of CSS 2. You should use this specification for developing a CSS that uses a decorative site element. |
| Text should be formatted with CSS 1. | CSSs that specify the style of your text content should be created by using CSS level 1, or CSS 1. For details of this specification, see [Cascading Style Sheets, level 1](http://go.microsoft.com/fwlink/?LinkId=140171&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140171&clcid=0x409). The CSS level 1 specification provides guidance about the creation of style sheets for the definition of font, margin, and color styles. |
| Styles should be included in external CSS files. | This enables you to create CSSs that you will apply to multiple pages. By removing redundant code from the HTML files that you create, you can have a single reusable and maintainable CSS file with a one-to-many relationship to pages. This CSS file can be cached, which improves overall performance of your sites. |
| External CSS should be sent with the correct Multipurpose Internet Mail Extensions (MIME) type. | The W3C specification suggests that CSS files should have a MIME type of **text/css**. The MIME type is a standard that enables the publication of binary data on the Internet. The MIME type is stored in the header for a file and informs browsers that they cannot process the data as normal text. By following this guidance, you can hope to reduce browser compatibility issues. |
| CSS classes and IDs must be valid. | It is important that CSS classes and IDs are valid. The following provides guidance on the aspects of CSS class and ID design that your Web developers must take into consideration:   * **Classes** * Must be generic. * Must be descriptive, such as describing functionality over appearance. * **IDs** * Must begin with a letter. * Cannot begin with a number. * Cannot begin with an underscore (\_). |

Optimizing Custom Controls and Code

You can add custom controls and other custom code to your WCM solution in many ways. For example, you could create custom Web Parts, workflows, or ASP.NET controls. You will see in Module 6 that you can customize the content deployment system and the editing tools for pages. If these custom objects are created carelessly, you might find that they slow responses to users significantly. If this happens, all of your HTML and CSS optimization is rendered pointless.

As an architect, you should ensure that developers realize the need for code optimization and common pitfalls. You should also ensure that they follow good working practices.

This section contains the following topics:

* [Considerations for Optimized Custom Code](#ConsiderationsforOptimizedCustomCode)
* [Ensuring that Developers Optimize Their Code](#EnsuringDevelopersOptimizeCode)

Considerations for Optimized Custom Code

Developers should consider the following issues when they write code:

* **Database server round-trips**: Each query against a database takes time, particularly if the database is stored on a separate server, which is frequently the case. Developers should ensure that the smallest number of database queries is used.
* **Foreach loops and recursive functions**: Developers might test their control against small data sets and observe no delay in rendering as the code loops through entries in a database or tags in an XML file. However, when the control is deployed to a production server, the data sets can be much larger, and the control can stop the page rendering as it loops through the entries.
* **CPU utilization**: Custom code that makes unnecessary use of the CPU for calculations can delay the rendering of the current page and also all other pages that concurrent users might have requested.
* **Page download size**: Ensure that your controls do not add unnecessary data to the rendered page. If your controls do this excessively, the page takes a long time to download to the browser.
* **Client-side code efficiency**: JavaScript and other code that runs on the client can improve efficiency by removing the need to reload the page from the server. However, you must ensure that this code executes rapidly, even on client computers with low performance specs.
* **AJAX callbacks**: Asynchronous JavaScript and XML (AJAX) is a Web programming technique that improves responsiveness by exchanging small amounts of data with the server when users click controls, instead of reloading the entire page. You can make use of AJAX by using the ASP.NET AJAX framework. The classes in this framework enable developers to add AJAX-based controls to their ASP.NET pages, such as Office SharePoint page layouts. ASP.NET AJAX will be integrated into the Microsoft .NET Framework 3.5.

Ensuring that Developers Optimize Their Code

You must emphasize the need for efficient code in your solution design documents. However, you can best ensure efficient code by requiring that developers and testers review the project at several stages:

1. **At the code design stage**: When the developers create detailed software designs, some inefficiencies can be eliminated if the documentation is detailed.
2. **In the source code**: If you ensure that source code must be reviewed and approved before each component is considered for deployment, problems such as long loops or poor error handling can be eliminated.
3. **At the staging stage**: One of the purposes of staging content and custom code is to ensure that it executes efficiently against realistic databases and content.
4. **After deployment**: Page rendering on the production Web servers should be monitored by using Web logs and other techniques. This will detect any delays that arise as the use of content changes over time, for example, as the size of the content database grows.

Lesson 2: Network Security

WCM is designed to publish information to an audience by using Web technologies. It is essential to provide the best user experience by providing fast access to content. In this lesson, you will see how you can enhance WCM performance by modifying caching and compression options in Office SharePoint Server 2007 and Internet Information Services (IIS) 6.0.

Note: You can find supplemental information on many of these subjects in the Configuring Performance Options webcast, accessible through the table in [Advanced lectures and white papers](http://go.microsoft.com/fwlink/?LinkId=140172&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140172&clcid=0x409).

### Objectives

After completing this lesson, you will be able to:

* [Explain how to manage static and dynamic content compression in IIS](#IISCompression)
* [Describe how to configure the output cache for site collections, sites, and page layouts](#OutputCaching)
* [Describe how to configure disk-based caching for binary large objects (BLOBs)](#DiskbasedCachingforBLOBs)
* [Describe the ways in which other mechanisms, such as the object cache, can affect performance](#OtherCachingMechanisms)

IIS Compression

You can enable IIS Hypertext Transfer Protocol (HTTP) compression to use bandwidth more effectively. This provides faster transmission time between compression-enabled browsers and IIS for static files (such as .htm, .html, and .txt files), dynamic application response files (such as asp, .dll, and .exe files), or both types. If you enable IIS compression from IIS Manager, you must enable compression globally. IIS compression is enabled by default for Office SharePoint Server 2007, in contrast to previous versions. You can use compression for individual sites, but you will need to do this programmatically. You can use the adsutil.vbs administrative script for IIS 6.0 to achieve site compression.

This section contains the following topics:

* [How IIS Compression Improves Performance](#HowIISCompressionImprovesPerformance)
* [When to Use IIS Compression](#WhentoUseIISCompression)
* [Setting Compression Options](#SettingCompressionOptions)
* [Compression Levels](#CompressionLevels)
* [Site-Specific Compression](#SiteSpecificCompression)

How IIS Compression Improves Performance

When IIS receives a request, it checks whether the browser that sent the request has sent a header announcing that it is a compression-enabled browser. When the request is for static content, IIS searches the IIS Temporary Compressed Files folder for a compressed version of the file. If the file does not exist, an uncompressed version is sent and a background thread runs a compression job and stores the file in the folder. Subsequent requests for that file are serviced directly from the compression directory.

The IIS Temporary Compressed Files folder is at the following location by default:

C:\Windows\IIS Temporary Compressed Files

If the file contains dynamic content, IIS compresses the response as it is generated and sends the compressed response to the browser. No copy of the file is cached by the Web server. This means that if a dynamic file is requested often, the IIS server must compress the file each time. You must decide whether this processing overhead is worthwhile because it provides a reduced demand on network bandwidth.

When to Use IIS Compression

To identify whether your server has poor bandwidth, review the performance counter Network Interface\Bytes Sent/sec. This counter displays the rate at which bytes are transmitted over a TCP/IP connection by monitoring the network adapter. Always check that your Web server processor is not overloaded before you enable IIS compression for dynamic file types. If the % Processor Time counter is already 80 percent or higher, enabling HTTP compression is not recommended.

IIS compression is particularly helpful when there is a slow connection such as a dial-up line between the IIS server and the user’s computer.

To determine whether IIS compression will help, you should establish a baseline before you make configuration changes. Review the current performance counters and then make the configuration changes. You can then monitor the performance counters and assess the difference that the change has made.

Setting Compression Options

You can configure whether to enable IIS compression by using IIS Manager. This will only enable you to start static or dynamic compression and set the cache directory for static content. IIS 6.0 holds many of its configuration options in the metabase. This is an XML file (metabase.xml) located at c:\windows\system32\inetsrv. You can set file types to compress, compression schemes, compression level, and the compression folder by editing this file. Microsoft provides a Microsoft® Visual Basic®, Scripting Edition (VBScript) utility to edit the file safely, called adsutil.vbs, which you can use in conjunction with the cscript.exe command. You should only edit this file if the default compression settings do not work for your organization. The default settings are shown in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Default Settings IIS Compression** | **File Type** | **Default Configuration** | **Configuration Method** |
| File types compressed | Static | .txt, .htm, and .html | Metabase |
|  | Dynamic | .exe, .dll, and .asp | Metabase |
| Compression schemes | Static | Both gzip and deflate | Metabase |
|  | Dynamic | Both gzip and deflate | Metabase |
| Compression level | Static | 10 | Metabase |
|  | Dynamic | 0 | Metabase |
| Compression directory | Static | Size: 95 megabytes (MB)  Location: %Windir%\IIS Temporary Compressed Files | Metabase or IIS Manager |
|  | Dynamic | No directory | N/A |

The syntax of the adsutil.vbs script to include .css files, in the list of static files for compression, is as follows.

cscript.exe c:\inetpub\adminscripts\adsutil.vbs set w3svc/Filters/Compression/DEFLATE/HcFileExtensions "txt" "css"

You can add other extensions in quotation marks with each type separated by a space. Consider carefully the file types that you add to the list. For example, file types that have built-in compression, such as JPEGs and MP3s, should not be added.

You can use the following command to display help information about CScript syntax.

cscript.exe c:\inetpub\adminscripts\adsutil.vbs help

Be careful when editing metabase settings to avoid hurting user response time. For example, if you disable a compression scheme (gzip or deflate), you must be sure that your client systems do not use that scheme. If client systems use the disabled scheme, all files will be transmitted uncompressed.

Compression Levels

Higher compression levels result in smaller files and therefore less network traffic between client and server. However, higher levels require more CPU and memory resources on the server. A compression level of 10 is recommended for static files (this is the default in Office SharePoint Server). A compression level of 9 or 10 is recommended for dynamic files (in Office SharePoint Server 2007, the default compression level for dynamic files is 0). You should monitor to ensure that these settings do not overload the CPU or memory.

Site-Specific Compression

You can use adsutil.vbs to use compression for individual sites.

► Enable compression on a single site

To enable IIS compression for a directory at, for example, http://www.litwareinc.com/MySite/CompressibleFiles—where the IIS web site identifier for the www.litwareinc.com Web application zone is 1—the procedure is as follows:

1. Disable global static compression by executing the following command at a command prompt:

cscript.exe c:\inetpub\adminscripts\adsutil.vbs set w3svc/filters/compression/parameters/HcDoStaticCompression false

1. Enable static compression at this directory by executing the following command at a command prompt:

cscript.exe c:\inetpub\adminscripts\adsutil.vbs set w3svc/1/root/MySite/CompressibleFiles/DoStaticCompression true

► Disable compression on a single site

To disable IIS compression for a directory at http://www.litwareportal.com/MySite/CompressibleFiles, where the IIS Web site identifier for the www.litwareportal.com web application zone is 1, the procedure is as follows:

1. Enable global static compression by executing the following command at a command prompt:

cscript.exe c:\inetpub\adminscripts\adsutil.vbs set w3svc/filters/compression/parameters/HcDoStaticCompression true

1. Disable static compression at this directory by executing the following command at a command prompt:

cscript.exe c:\inetpub\adminscripts\adsutil.vbs set w3svc/1/root/MySite/CompressibleFiles/DoStaticCompression false

Output Caching

If a site is enabled for output caching, a setting that is disabled by default, the HTML markup generated at run time by each Web page is cached in the ASP.NET output cache. The use of the cache is based on the specified cache profile. Output caching enables the server to cache dynamic pages and user controls for subsequent requests without the need to recall them each time. Cached pages can be served to site visitors without processing by the ASP.NET pipeline. This is only the case for calls where user privilege is sufficient, so security is not compromised. Checking authorization does increase the overhead on a Web server, so it is probably better to implement caches on servers where security is common to all users; for example, Internet servers designed for anonymous access. You should also implement caching on servers with static content because it is far more likely that you will receive multiple requests for the same page in this scenario.

Office SharePoint Server 2007 creates a separate space in the output cache for each group with unique security settings, to ensure that cached pages are not served to those who do not have permission to view them. Therefore, for an efficient output cache, you should minimize the number of such groups or design your taxonomy and site structure to facilitate output caching.

You can use output caching on a site or site collection only if you have activated the Office SharePoint Server 2007 Publishing Infrastructure feature for the site collection and the Office SharePoint Server 2007 Publishing feature for the site.

The behavior of output caching can be specified at the following levels:

* Site collection
* Site
* Page layout

This section contains the following topics:

* [Configuring Site Output Caching](#ConfiguringSiteOutputCaching)
* [Page Layout Caching](#PageLayoutCaching)
* [Post-Cache Substitution](#PostCacheSubstitution)

Configuring Site Output Caching

You can configure the output cache on a per-site collection and site basis. Typically a subsite inherits its caching profile from the site collection, but you can choose to specify a different cache profile, depending on the function of the site. There are four default caching profiles that you can configure from the Site Collection Output Cache Settings page (top-level site only – click **Site Settings** and then **Site Collection Output Cache**) and the Publishing Site Output Cache Settings page (all sites with publishing features enabled – click **Site Settings** and then **Site Output Cache**):

* Disabled
* Public Internet (Purely Anonymous)
* Extranet (Published Site)
* Intranet (Collaboration Site)

Note: You should enable and verify Output Cache Debug statements to ensure that output caching is functioning properly.

Page Layout Caching

If you have a home page for site entry, this will almost certainly be the most-used page on your site. Consider making the information on this page static so that you can apply a special cache profile to that unique page layout and override the caching behavior of the entire site collection.

► Configure page output cache settings for page layouts

1. On the **Site Actions** menu, point to **Site Settings**, and then click **Modify All Site Settings**.
2. In the **Galleries section**, click **Master pages and page layouts**.
3. On the Master Page Gallery page, point to the name of the page layout to which you want to apply a cache profile, click the down arrow that appears, and then click **Check Out**.
4. Point to the page layout again, click the down arrow that appears, and then click **Edit Properties**.
5. If you want to edit the authenticated cache profile, in the **Authenticated Cache Profile** drop-down list click the authenticated cache profile that you want to apply to the page layout.
6. If you want to edit the anonymous cache profile, in the **Anonymous Cache Profile** drop-down list click the anonymous cache profile that you want to apply to the page layout, and then click **OK**.
7. Point to the name of the page layout that you want to check in, click the down arrow that appears, and then click **Check In**. You might also need to approve the page layout.

Post-Cache Substitution

Consider a page that has a large quantity of static content, with a small amount of dynamic content, such as an About Us page with an advertisement rotator. Most of the information on the page does not change, so you want to cache it, but if you cache all the content, users see only one advertisement repeatedly instead of a different one on each request.

To avoid this problem and use caching, you can use post-cache substitution. The developer places the dynamic content within an ASP.NET Substitution control. This marks the content as exempt from caching. At run time, the Substitution control calls a method that replaces its contents.

Disk-based Caching for BLOBs

You can cache BLOBs, which are primarily image, sound, and video files (or other large, relatively static files), to avoid repeated database input/output (I/O) and to minimize network traffic. BLOB caching is most appropriate for large files that change rarely. This setting is configured in the Web.config file for the Web application. In BLOB caching, the front-end Web servers can actually cache and store local copies of certain file types, specified in the Web.config file for the Office SharePoint Server 2007 Web application, and will retrieve the files from the front-end Web server’s local file system rather than making the round trip to and from the SQL Server database.

This option is disabled by default, and you must modify the following line in the Web.config file of the Web application for each front-end Web server in the server farm.

<BlobCache location="C:\blobCache" path="\.(gif|jpg|png|css|js)$" maxSize="10" max-age="86400" enabled="false"/>

The attributes of the XML are as follows:

* **Location** — The directory where the cached files will be stored.
* **Path** — Which files are cached based on the file name extension.
* **maxSize** — The maximum allowable size of the disk-based cache in gigabytes (GB).
* **max-age** — The maximum amount of time in seconds that the client browser caches BLOBs that are downloaded to the client computer.
* **Enabled** — A Boolean value that disables or enables the cache.

Of these settings, the **max-age** attribute can make the greatest impact on performance and should be carefully tuned. Longer ages reduce the number of times the file must be downloaded, but if the file changes regularly the user might not see the latest version.

► Enable disk-based BLOB caching in Web.config

1. Click **Start**, point to **Administrative Tools**, and then click **Internet Information Services (IIS) Manager**.
2. In Internet Information Services (IIS) Manager, click the plus sign (+) next to the server name that contains the Web application, and then click the plus sign next to **Web Sites** to view the Web application(s) that have been created.
3. Click the name of the Web application for which you want to configure the disk-based cache, and then double-click the Web.config file.
4. In the **Web.config Properties** dialog box, on the **ASP.NET** tab, select the file location, and copy it.
5. Click **Start**, and then open Windows® Explorer.
6. Click in the **Address** box, and paste the location of the Web.config file into the box.
7. Click **Go** to open the Web.config file.

The file will open in the associated editor, such as the Microsoft® Visual Studio® 2005 development system. If you do not have an associated editor, you can use Notepad.

Note: You should make a backup copy of Web.config before you make any changes to the file.

The Master Page gallery and Style Library do not work by default for anonymous users; for BLOB caching to work for these lists, you must go to these lists and explicitly switch on anonymous access for caching.

Note: You can flush the BLOB cache at any time in the Office SharePoint user interface. To do this, click **Site Settings** and **Site Collection Object Cache**. Then click **Force this server to reset its disk based cache** and click **OK**.

Office SharePoint Publishing Cache Counter

IIS provides a number of counters with which you can monitor the performance of your BLOB caching. These counters are found in the IIS Global object and are detailed in the following table.

|  |  |
| --- | --- |
| **Counter** | **Description** |
| Current BLOBs Cached | The BLOB information blocks currently in the cache. |
| Total BLOBs Cached | The number of BLOB information blocks that have been added to the cache. |
| BLOB Cache Hits | The number of successful lookups in the BLOB cache. |
| BLOB Cache Misses | The number of unsuccessful lookups in the BLOB cache. |
| BLOB Cache Hits % | The ratio of BLOB Cache Hits to the total number of cache requests. |
| BLOB Cache Flushes | The number of BLOB cache flushes that have occurred since the service started. |
| Total Flushed BLOBs | The number of BLOB information blocks that have been removed from the cache since the service started. |

Other Caching Mechanisms

Office SharePoint Server 2007 supports caching of some page items such as navigation data through the object cache. This is configured on a per-site basis, but is always enabled by default. Caching page items reduces the need to retrieve field data from the database each time a page is rendered. The caching system also caches complete field data for a page, except for data from Web Parts.

The default object cache size is 100 MB, but you can adjust it to ensure that your cache is working optimally. To check the cache, you can use the Cache Hit Ratio counter. This should be about 90 percent. You should not make any cache too big because this can waste memory. Because your front-end Web servers might also be configured to use output caching and store the index file, you might create contention for memory if you use too large an object cache.

The object cache is also used to cache items that are retrieved as part of cross-list queries, which are primarily used in search.

This section contains the following topics:

* [Retiming the IIS Object Cache](#RetimingtheIISObjectCache)
* [Web Part Caching](#WebPartCaching)
* [ASP.NET Cache Object](#ASPNETCacheObject)
* [Web Part Cache in Memory or SQL Server](#WebPartCacheinMemoryorSQLServer)
* [The Caching Application Block](#CachingApplicationBlock)

Retiming the IIS Object Cache

You can change the period of time that an unused object remains in the cache by adding the ObjectCacheTTL entry to the registry.

Note: You should not edit the registry unless you are experienced in registry administration and have saved a backup of your registry.

The value of ObjectCacheTTL controls the static file cache by specifying the Time to Live (TTL). This sets the length of time that objects are held in cached memory. If an object in the memory cache has not been referenced for the defined period, that object is flushed out of the cache.

The default value for ObjectCacheTTL is 30 seconds. Usually, a change is implemented only in extreme circumstances. Limited memory is more likely to be mitigated by installing more memory. However, if your site content is dynamic, you can use a lower TTL to prevent system memory from being used to cache a large number of volatile objects.

If you have static data, you can set the ObjectCacheTTL entry to unlimited. This disables the object-cache scavenger and allows cached objects to remain in the cache until the cached object changes.

► Reset the time that an unused object remains in the object cache

You can reset the time by adding the ObjectCacheTLL entry to the registry:

1. On the **Start** menu, click **Run**, type **regedit.exe** and then click **OK**.
2. In the Registry Editor, navigate to the following subkey:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters

1. Right-click the **Parameters** subkey, point to **New**, and then click **DWORD Value**.
2. Rename **New Value** to **ObjectCacheTTL**
3. Right-click **ObjectCacheTTL**, and then click **Modify**.
4. In the **Base section**, select **Decimal**.
5. In the **Value Data** box, type the number of seconds that you want an unused object to remain in the cache, and then click **OK**.

The default value is 30 (seconds). You can enter any value from zero, to disable caching, through 4,294,967,295 (unlimited), to disable the object-cache scavenger and allow cached objects to remain in the cache until the cached object changes.

Web Part Caching

There are a number of options for Web Part caching. These options are not specific to Office SharePoint Server 2007, but can be used to develop configurable caching scenarios.

ASP.NET Cache Object

A Web Part is essentially an ASP.NET server control. This enables you to implement caching in the control. You can cache objects used by the control or the HTML rendered by the control. Multiple instances of your control can share the same cache. The cached Web Part and data are available to any user with the appropriate permissions. You can set an expiration policy based on time or dependencies on other objects for cached objects. Items are removed from the cache when they expire or if a dependent file on disk changes. If memory pressures on the machine mean that items must be purged from the cache, ASP.NET will then remove items based on their priority, even if they have not expired.

Web Part Cache in Memory or SQL Server

You can configure the Web Part cache to cache items on a per-user or per-Web Part basis. This does not enable multiple instances of a Web Part to share the same cache. When items are added to the cache, the Web Part framework adds text to the cache item key to make the key unique for either the user or the Web Part instance.

You can also use SQL Server to cache serialized Web Part objects into the database, with other Web Part properties. All Web Parts that are used are cached. You must mark the serialized object, otherwise the Web Part framework will not attempt to write it into the database. Items can be cached for each user or for each Web Part instance. Cached content is available to all of the Web servers in the farm, so closing the application does not remove the data from the cache. You also usually will have more database storage capacity than memory, so more data can be cached without affecting system performance, although a disk-based cache is slower then a memory-based cache.

The Caching Application Block

The Caching Application Block is developed at Microsoft and is specifically designed to work in any .NET application. In the caching application block, cached items are held in a table. This is dynamic, so the table entries are lost if the Web application is closed. You can implement a backing store of the table, which is held in a SQL Server database table. The SQL Server table repopulates the cache table when the Web application is restarted.

The caching application block offers developers a number of configuration options such as providing multiple caches in one application with different settings. For more information, see [Caching Application Block](http://go.microsoft.com/fwlink/?LinkId=140179&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140179&clcid=0x409).

Review of Module 5

* Optimizing content
* Optimizing servers

References

The following videos provide supplemental information to these modules. There is no one-to-one correspondence between the modules and the videos.

* [Video 1](http://go.microsoft.com/fwlink/?LinkId=140097&clcid=0x409) (http://go.microsoft.com/fwlink/?LinkId=140097&clcid=0x409)
* [Video 2](http://officecpub/Teams/itpro/Documents/White%20paper%20library/WCM%20modules/Video%202) (http://go.microsoft.com/fwlink/?LinkID=140063&clcid=0x409)
* [Video 3](http://officecpub/Teams/itpro/Documents/White%20paper%20library/WCM%20modules/Video%203) (http://go.microsoft.com/fwlink/?LinkID=140068&clcid=0x409)
* [Video 4](http://officecpub/Teams/itpro/Documents/White%20paper%20library/WCM%20modules/Video%204) (http://go.microsoft.com/fwlink/?LinkId=140101&clcid=0x409)