# Implementing a Brand in a SharePoint Publishing Site

## [[1]](#footnote-2)#Introduction

Andrew Connell, [Andrew Connell Inc.](http://www.andrewconnell.com) (Microsoft MVP)

Month 2006

Applies to: Microsoft Office SharePoint Server 2007

Summary: Microsoft Office SharePoint Server 2007 Publishing sites are commonly used as the external face for companies and organizations. As the online presence, they should employ a consistent look and feel to make navigation and finding information as easy as possible for users. The challenge comes in the implementers of the site as there are various options, each with its own advantages and disadvantages, in the implementation of brand. This article provides guidance on selecting the right approach for implementing a brand in a SharePoint Publishing site. (XX printed pages)

Contents

<table of contents will go here>

Main Subhead#ProductVersionKeyword\_MainSubhead

## [[2]](#footnote-3)#Introduction to Branding SharePoint Publishing Sites

The most recent version of SharePoint, Microsoft Office SharePoint Server (MOSS) 2007, introduced the capability to host content-centric sites on the SharePoint platform. These sites, commonly referred to as Publishing sites for the SharePoint Publishing Features they employ, leverage a series of concepts and capabilities in MOSS 2007 called Web Content Management (WCM). Publishing sites are frequently used in an Internet facing scenario as the public face for a company or organization. Thus, it is very important for these sites to have a consistent look and feel. However developers and designers can quickly get confused with the various options available for how to implement and deploy the site's brand.

This article will explore things that a Publishing site project team should consider when planning how to implement a brand within SharePoint. While many of the concepts and issues covered in this article cover Windows SharePoint Services (WSS) 3.0 sites, the article is foxed on Publishing sites exclusively.

## SharePoint Sites are ASP.NET 2.0 Applications

Before diving into the details of branding a SharePoint site, it helps to level the playing field a bit. SharePoint projects are frequently overcomplicated because many think of SharePoint as much more than it really is. Starting with the third generation of SharePoint (WSS 3.0 & MOSS 2007), it is built natively on top of the .NET Framework and can be considered a pure ASP.NET 2.0 application. The SharePoint product team simply implemented a custom HTTP application, and custom HTTP handlers and modules to implement SharePoint and change how ASP.NET 2.0 works by default.

The implementation of the SharePoint architecture is detailed in Chapter 2: SharePoint Architecture of the [Inside Microsoft Windows SharePoint Services 3.0](http://www.microsoft.com/mspress/books/9692.aspx) book by [Microsoft Press](http://www.microsoft.com/learning/books/default.mspx). [The chapter is available in the WSS 3.0 Software Development Kit (SDK) as an excerpt](http://msdn.microsoft.com/en-us/library/bb892188.aspx).

While there are many similarities between SharePoint and ASP.NET 2.0, as expected, there are plenty of differences as well. Some of the more prominent similarities and differences are detailed in the following sections.

### ASP.NET 2.0 and SharePoint Site Similarities

Because SharePoint is built on top of ASP.NET 2.0, there are not only some striking similarities between the two technologies, but most aspects of ASP.NET 2.0 sites bleed through directly into SharePoint. SharePoint fully leverages ASP.NET 2.0 concepts like the navigation provider and membership provider models. There may some additional configuration to make SharePoint aware of certain aspects of ASP.NET 2.0 technologies, such as in the case of the membership provider model, but SharePoint still fully offloads that work to ASP.NET 2.0.

Specifically to the theme of this article, the technologies and techniques used to branding SharePoint sites is virtually the same as it is in ASP.NET 2.0. Images and cascading style sheets (CSS) are used to implement a custom brand within a traditional ASP.NET 2.0 site. SharePoint employs the exact same techniques. In addition, ASP.NET 2.0 sites offer master pages which simplify the global branding of a site. SharePoint also leverages master pages however in a somewhat different way than ASP.NET 2.0. This is covered in the next section.

### ASP.NET 2.0 and SharePoint Site Differences

While there are many similarities between ASP.NET 2.0 and SharePoint sites, there are also quite a few differences. One of these is the concept of customized and uncustomized files. In an ASP.NET 2.0 site it is relatively easy to determine where a file resides by inspecting the URL (when things such as URL rewriting are not employed). This is because ASP.NET 2.0 is configured out-of-the-box (OOTB) to retrieve files directly off the file system. This approach does not work in SharePoint because one ASPX page on the file system could be used as a template for many pages in one or more SharePoint site. To address this, the product team implemented a virtualized file system that is stored in the SharePoint content databases for each SharePoint site. In addition, files in a SharePoint site can be customized or uncustomized. An uncustomized file is that simply exists in the virtualized file system as a pointer to the template it is based on that resides on the file system. Customized files on the other hand are ones that have been modified and who's source is stored in the SharePoint content database. This enables power users to customize the same file on a site-by-site basis without affecting other sites that leverage the same template on the file system. For more information on this concept refer to the MSDN article [Understanding and Creating Customized and Uncustomized Files in Windows SharePoint Services 3.0](http://msdn.microsoft.com/en-us/library/cc406685.aspx).

The previous section covered master pages as a similarity between ASP.NET 2.0 and SharePoint site. While the same technology is used in both environments, the implementation is a little different. In an ASP.NET 2.0 site, content pages reference a specific master page and this defined by the site designer or developer at design time. This model does not work for SharePoint site because when developing ASPX pages, one cannot always be certain which SharePoint site the page will be used in. Therefore the SharePoint team elected to allow site owners select a master page for use across all pages in a site. Therefore content pages, ASPX files, in a SharePoint site should not be configured to point to a specific master page file. Instead, dynamic tokens are used to tell SharePoint, at runtime, which master page should be used. SharePoint replaces this token with the URL of one of the two master pages that had previously been selected by the site owner for use in the current SharePoint site. This is detailed in the WSS 3.0 SDK: [Customizing Master pages in Windows SharePoint Services](http://msdn.microsoft.com/en-us/library/ms476046.aspx).

Another big difference between ASP.NET 2.0 and SharePoint is the code and custom file deployment story. In ASP.NET 2.0 sites, custom code and files including DLL's, ASPX's, ASCX's, GIF's, JPG's, CSS and JS's to name just a few are typically deployed as loose files to production systems. While Visual Studio 2008 does have deployment / publish capability, developers do not usually have write access to a production server. Another deployment option in ASP.NET 2.0 sites is to create an installer (MSI) or package all the files into a ZIP for an administrator to use in the deployment on a production server. Finally, a management application such as [Microsoft Systems Management Server](http://www.microsoft.com/smserver) or a 3rd Party deployment application can be used for the deployment of an ASP.NET 2.0 application.

On the other hand, SharePoint includes a robust custom code and file deployment mechanism. The Windows SharePoint Services Solution Package framework (aka: solution framework, SharePoint solutions or just WSP's) is SharePoint's internal deployment vehicle for custom code and files. Developers package up all custom code and files into a Microsoft Cabinet (\*.CAB) file with the \*.WSP file extension, add a special manifest file (manifest.xml) to the root of the WSP and then add the solution to the SharePoint farm's solution store. The manifest file lets SharePoint know about all the files in the solution, telling it what each file is used for and where it should be deployed on the server. Then, either on demand or at a scheduled time, SharePoint deploys the files to the other SharePoint servers based on what is defined in the manifest and selected in the deployment settings such as which Web application the solution is deployed for. If SharePoint is running in a load balanced environment, the solution framework automatically determines which servers to deploy the solution to and does deploys to all servers are the same time. Finally, the solution framework also supports retracting, or "undo-ing", the deployment. For more information on the solution framework, refer to the official documentation in the WSS 3.0 SDK on MSDN: <http://msdn.microsoft.com/en-us/library/aa543214.aspx>.

## Site Content is not Branding

Before covering branding SharePoint Publishing sites, it's important to understand the difference between content and branding assets. Quite simply, branding is anything that is used to implement the user interface of the site. This could be images or CSS files… thinks that implement the corporate brand or theme. The brand of a SharePoint Publishing site is typically not very dynamic and doesn't change that often. Rather organizations typically undergo a rebranding campaign, either slightly customizing or completely revamping the look and feel of the company. At any rate, the files associated with the brand do not change on a frequent basis (every few days or weeks). In addition, the branding files are usually owned and controlled by the site developers and designers… the team responsible for creating and maintaining the site.

Content, on the other hand, is much more dynamic. Content consists of the text and media, such as images and collateral, that makeup the reason for the site. These are the press releases, product information pages and images of the products. Content is owned and controlled by the content owners and authors rather than the production team responsible for the site. Content also usually follows a publishing process complete with workflow, permissions limiting who can create, approve or publish the content and versioned. Another difference between content and branding is that content is usually updated more frequently than branding files. Some pages might be updated daily or every few days, other pages may be updated much less frequently. Regardless, the time between content updates is usually much shorter than the time between branding updates.

The reason for this distinction is that the two are stored in very different locations and should be kept separate from each other. There are multiple reasons for this separation. First, the individuals who own the content and branding are typically different people or groups. By putting everything in one location, the permissions can get mixed. Second, as outlined above, content typically follows a publishing process with robust workflow and business rules within production whereas branding is validated in a development or test environment and ultimately rolled out into production.

With an understanding of the differences between content and branding assets, it is now time to look at the various deployment options.

## Deployment Options for Branding Files

As covered in the previous section, branding files are different, and should be treated differently, from content. First, consider where content should go within a Publishing site. Every Publishing site within a Publishing site collection contains three special libraries: Pages, Images and Documents. The Pages library is where the content pages, also known as the Web pages, should be stored for each site. The Images and Documents library is where content collateral should be stored that is referenced from pages in that same site. Each site has these libraries to facilitate a certain level of control over what sections of the site certain users can manage. For instance, maybe contributors to the Press Releases section of the site should not have the same rights or access to the materials in the Products section. While SharePoint allows them to have rights in both, it does support keeping things isolated if the business requirements demand that.

There are two special libraries in the top level site of the site collection: Site Collection Images and Site Collection Documents. The reason for these two libraries is that regardless of what site a content owner or author has access to, there are some assets that everyone should have access to. For instance, the company logo or privacy policy should be available to all content authors. Instead of having multiple copies floating around the various Images libraries throughout the sites, they should only reside in one place as well implemented content sites are all about content reuse and not content duplication.

When it comes to branding files, they should be kept separate from the content files for reasons outlined in the previous section. Therefore branding files should not go in the Images, Documents, Site Collection Images or Site Collection Documents libraries throughout the Publishing site collection. So where should branding files be deployed?

As with most things in SharePoint, there are multiple options and each has advantages and disadvantages. The following sections cover three options on where to deploy branding files as well as the advantages and disadvantages of each.

### Deploying Branding Files to the Site's Web Root Folder

One obvious choice, especially to those with an ASP.NET 2.0 development background, is to simply deploy the branding files to the root folder (or a subfolder within it) of the Web application hosting the SharePoint Publishing site collection. This works just fine and is familiar to traditional ASP.NET 2.0 developers and designers as it follows a similar model to traditional ASP.NET 2.0 Web sites. However this approach brings with it many disadvantages.

First and foremost, there is no OOTB automated deployment option when deploying files to the Web root. SharePoint solutions, as previously covered in this article, cannot deploy files to the Web root of a Web application. They can only copy files to a location within the SharePoint "12" folder (usually c:\Program Files\Common Files\Microsoft Shared\web server extentions\12), modify code access security policies, deploy assemblies to the server's global assembly cache or \bin folder in the Web root, and make minor changes to the Web application's web.config file. This means that not only can the deployment of the brand not be scheduled using the same OOTB solution framework vehicle, but that if a SharePoint farm employs multiple load balanced Web front end (WFE) servers, the deployment must be coordinated for all servers. This challenge can be mitigated using a deployment management package as previously covered.

Building off this point, when a new SharePoint site is added to the farm, if the SharePoint Web services are started, SharePoint will automatically deploy all previously deployed solutions to the new server. However because the branding files were not deployed using SharePoint solutions, they will not be automatically deployed. Thus, farm administrators must be aware of what manual deployment steps must be implemented on the new server.

One aspect to this approach is the files are stored within the SharePoint site. This means that the files cannot be versioned as they could be within a publishing site, nor could they be scheduled. For instance, maybe a few files that makeup branding changes, like CSS or a new company logo, should not be used before a specific date and time. If the assets were in a Publishing site, they could be scheduled for when they would go live. This also means that the files are not tightly coupled to the site and thus have to be backed up separately from the site.

Another concern with this approach is backup/restore and disaster recovery. When the Publishing site is backed up, either using SQL Server database backups or using SharePoint’s backup capabilities, only the content within the site is backed up; none of the files on the file system are backed up.

In addition, if multiple Publishing site collections were hosted in the same Web application, they would all have access to the same branding files. This is because all files in the web applications root folder are accessible by all site collections in the Web application. Granted this may or may not be an issue in the business requirements, but if so it is something to be aware of. However, this may in fact be considered an advantage of this approach!

### Deploying Branding Files to the \_layouts Directory

Another option is to deploy the files into SharePoint’s \_layouts virtual directory. This folder maps to the [..]\12\TEMPLATE\LAYOUTS folder. This approach mitigates a big disadvantage of the previous technique as administrators can deploy files using the SharePoint solution framework. This means the deployment of branding files can not only scheduled but they are also automatically deployed to all SharePoint WFE’s in the farm when the solution is deployed.

One of the downsides to this approach is that the files still reside on the file system and thus most of the same concerns mentioned in the previous approach carry over to this one as well. This includes the lack of versioning or scheduling branding files.

In addition, another potential concern here is that files deployed to the \_layouts virtual directory are available to all SharePoint sites within the farm… not just Publishing sites. This is due to the fact that all SharePoint WFE’s in a farm should be virtual mirrors of each other each has a single [..]\12\TEMPLATE\LAYOUTS folder. While each SharePoint Web application has a separate \_layouts virtual directory, they all point to the same folder on each server. This means that every site will have access to the same files. Again, maybe this is not a concern in a specific implementation, but if it is, administrators should be aware of it. However, just like the previous approach, this may be an advantage rather than a disadvantage of this approach. It all depends on the business culture, deployment concerns and business requirements.

### Deploying Branding Files to the Site Collection's Content Database

This approach is very different from the previous two approaches. Files do not have to live on the file system. Instead, they can be deployed to the Publishing site collection and thus, be included in the site’s virtual file system. This is how the OOTB Publishing Portal site template implements its brand.

The biggest advantage here is that all the branding is included in the same SharePoint content database that contains the content of the site. This means that either SQL Server database backups or SharePoint backups will also contain the branding files. In addition, because the files live in the content database they could be versioned or even scheduled.

Simply saying “they go in the site collection” or “they go in the content database” does not say much. Where exactly should they go? Recall the discussion previously in this article about the differences between content and branding and how they should be kept separate. Therefore images related to the brand should not be stored in any of the site’s Images libraries or the top level site’s Site Collection Images library. Instead, all branding files should be put in the Style Library SharePoint library found in the top level site of the site collection. All authenticated users in the Publishing site are automatically granted access to the Style Library which is a good thing considering the brand will be persisted across virtually all pages in the Publishing site! The same cannot be said for the Images or Site Collection Images libraries. While it may not be terribly common, there are plenty of Publishing sites where users are not granted access to the root of the Publishing site, rather they only have rights within a subsection of the site. Even though the Style Library is in the top level site of the site collection, all authenticated users in a site, even those who only have access to a subsection of the site collection, have access to the Style Library.

This approach may be a bit more comfortable to developers and designers as they can add content directly to the site’s virtual file system using tools such as Office SharePoint Designer 2007. However recall the previous discussion about customized and uncustomized files, especially the details of it. There can be challenges to maintaining the files that live exclusively in the virtual file system (ala content database). However these can be mitigated using the technique of provisioning files using Features. Thus, the brand can be packaged up in a SharePoint solution for a clean and repeatable deployment story and relatively easy updates. The technique of provisioning files using Features is covered in the [Understanding and Creating Customized and Uncustomized Files in Windows SharePoint Services 3.0](http://msdn.microsoft.com/en-us/library/cc406685.aspx) MSDN article.

## Summary

This article has explored the various options available those implementing a custom brand in a Publishing site. As explained for the start, the goal here is not to define the right or wrong approach to implementing a brand in a Publishing site. Rather, the goal is to provide guidance and education around the various options for implementing a brand. Consider all the options, the advantages and disadvantages of each and the specifics of each Publishing site project to adopt the right approach. It likely does not make sense for all projects to adopt the same process and technique.

## Acknowledgements

I would like to acknowledge [Randy Drisgill](http://blog.drisgill.com) and [Heather Solomon](http://www.heathersolomon.com) (SharePoint MVP) for their feedback in discussions that contributed to this article.

## Additional Resources

For more information, see the following resources:

* Microsoft Windows SharePoint Services Developer Center
* Microsoft Office Developer Centerhttp://msdn.microsoft.com/office/
* [Understanding and Creating Customized and Uncustomized Files in Windows SharePoint Services 3.0](http://msdn.microsoft.com/en-us/library/cc406685.aspx)
* [MSDN Book Excerpts: Inside Windows SharePoint Services 3.0 - Chapter 2: SharePoint Architecture](http://msdn2.microsoft.com/en-us/library/bb892188.aspx)
* [MSDN Book Excerpts: Inside Windows SharePoint Services 3.0 - Chapter 3: Pages and Design](http://msdn2.microsoft.com/en-us/library/bb964678.aspx)

1. # ProductVersionKeyword\_Introduction [↑](#footnote-ref-2)
2. # ProductVersionKeyword\_MainSubhead [↑](#footnote-ref-3)