This green paper is taken from the book *SharePoint 2010 Web Parts in Action* from Manning Publications. The author discusses Web Parts, a self-contained application that can be used within a SharePoint site. He explains how SharePoint’s wide set of out-of-the-box Web Parts can be used and customized to develop and deploy internal and external business solutions. For the table of contents, the author forum, and other resources, go to [http://manning.com/wilen/](http://manning.com/wilen/).

Microsoft SharePoint is making an impact on companies, from small businesses to large Fortune 500 companies. Users, developers, IT pros, and administrators have embraced it. SharePoint is one of Microsoft’s most important products and, ultimately, the fastest growing server product. A product like SharePoint is always a challenge to implement. It is a very broad platform, and it is sometimes hard to define what parts to use compared to specialized products. Developers have had mixed feelings about SharePoint throughout the years. From the beginning there was lack of extensibility, and, when the extensibility came, there was the lack of documentation. Each new version of SharePoint has introduced new challenges for developers, ranging from the developer toolset to the software features. Up until now, SharePoint developers have been rare, probably due to all of the mentioned obstacles, which made the threshold to become a great SharePoint developer quite high. But, thanks to excited developers and the strong online community, coupled with an engaged product team from Microsoft, these obstacles have been addressed with the new release of SharePoint 2010.

SharePoint is a platform for creating portals and web sites where the end-user without programming skills can create and build web sites and solutions using a standard web browser. Using Lists, they can store data in SharePoint or integrate with external line-of-business systems. Lists can be custom defined or predefined lists such as Contacts, Calendars, or Announcements.

A Web Part is a self-contained application that can be used within a SharePoint site. SharePoint comes with a wide set of out-of-the-box Web Parts, and you can download or buy even more from third-party vendors. Figure 1 shows you how a SharePoint site might look like using several different Web Parts.
Without Web Parts, SharePoint would not be such a successful and dynamic product. Web Parts can be small or standalone applications that are deployed into your SharePoint installation. With Web Parts, you can extend the functionality of your SharePoint investment and create new applications or even integrate with your existing applications to create a gateway from your SharePoint portal. Most of Microsoft’s products, such as Microsoft Dynamics AX and SQL Server Reporting Services also integrate with SharePoint using Web Parts.

Building solutions and applications using Microsoft SharePoint 2010 allows you to keep your focus on the business logic, solving the customer problem, and implementing the cool features. You do not have to worry about implementing security or authentication, building a framework for creating sites and pages, or scratching your head to create a solution for deploying and distributing your features onto several servers in a controlled manner. Monitoring of your servers, software, and solutions is another part that is important for large-scale scenarios, and you get this with SharePoint also.

To be a good or even a great SharePoint developer, you need to understand the when and why of Web Part usage. Especially, you need to understand what SharePoint offers for the end-users out of the box and what you as a developer can create with Web Parts. This will save you and your customer a lot of time and money.

Before we dig down deep into how to develop Web Parts for Microsoft SharePoint 2010 you should know what a Web Part is.

What is a Web Part?

As the name implies it is part of a web page. Web Parts are the building blocks for the construction of web pages. Web Parts are essentially ASP.NET web controls that inherit the web control base class in ASP.NET but have some special abilities compared to ordinary ASP.NET web controls. The special abilities given to Web Parts are focused on giving the end-user or administrator, not the developer, the possibility to customize, move, and edit the control. ASP.NET controls are normally configured and programmed on the server side with no or limited options for the end-user to affect its properties. Web Parts, on the other hand, are those building blocks that the end-user can pick and choose from and build their own web page. For example, the end-user may choose that a particular Web Part should show the contents of a document library with a specific set of columns, another Web Part should show a
graph generated from a data source, and so on. As the matter of fact, the most used Web Part in SharePoint is the Web Part that shows a list, such as a task list, a calendar, or a document library.

All Web Parts have a common set of properties, such as title, size, chrome layout, and other appearance settings. They also share a common set of methods, so that they can be initialized and rendered and the web page that contains the Web Part does not have to know exactly which Web Part it is going to show, just that it is a Web Part.

In a Web 2.0 standard fashion, the Web Part infrastructure also handles personalization of Web Parts and Web Part Pages, which means that every single end-user of the Web Part can have its own configured web page that looks different from the same page for other users.

An important part of the Web Part infrastructure is the ability to connect Web Parts. Connected Web Parts can send or retrieve information from one Web Part to another Web Part. This concept allows for advanced web pages and dashboards, which not only can be created by a developer—the end-user can connect the Web Parts using the web interface. SharePoint contains a special kind of Web Parts that are called Filter Web Parts, they have one single purpose and that is to deliver a filter to other Web Parts so that their content can be filtered. For example, you might have a product catalog Web Part and customize a Filter Web Part so it enables you to select search results by product category.

You, as a developer, can create configurable applications that the end-user can adapt to their way of working. The end-user might also combine the Web Parts by connecting them together and from that create their own applications or mashups.

This concept is not unique to SharePoint—all major vendors and platforms have their version of Web Parts, more or less evolved: in the Java world, they are most commonly called Portlets; Google uses Gadgets for their iGoogle site. Web Parts can also be used to connect portals from different vendors and platforms using, for instance, the Web Services for Remote Portlets (WSRP), a standard that allows you to publish content from one portal to another or the upcoming Content Management Interoperability Specification (CMIS) standard.

If you are an ASP.NET developer, you might already be familiar with Web Parts, since that infrastructure exists in the ASP.NET 2.0 Framework. But SharePoint has extended this Web Part infrastructure and introduced a number of new and improved features. This is what this is all about—understanding the SharePoint 2010 Web Part infrastructure and utilizing all of its features so that you and your customer save time and money.

As previously stated, Web Parts are essentially ASP.NET WebControl objects, or to be more exact, it is an ASP.NET Panel object. So from a developer perspective the first things you need to learn is ASP.NET programming, in the language of your choice: C#, Visual Basic, or even F#, if you like. SharePoint 2010 is built upon the .NET Framework version 3.5 and ASP.NET 2.0, so anything in those frameworks can be used when developing Web Parts for SharePoint 2010.

**Web Parts history**

Microsoft announced the concept of Web Parts in the middle of 2000, which was before SharePoint existed as a product. It seems like ages ago. Web Parts was then a part of the, long forgotten, Digital Dashboard; an ASP-, XML-, and VBScript-based portal framework. The aim of Web Parts was the same then as it is now.

"In Web Parts, we are providing the building blocks for next-generation digital dashboard solutions..." – Bob Muglia, June 2000, now President, Server and Tools Business

When SharePoint Portal Server 2001, called Tahoe, was released, it was based on the Digital Dashboard and used Web Parts for creating the portal. This was the first real step to what SharePoint is now. SharePoint at that time was not based on Microsoft SQL Server but on the Microsoft Exchange Web Storage System (WSS) and the focus was more on document management than portals.

In 2003, Windows SharePoint Services 2.0 and SharePoint Portal Server 2003 were released. They were based on the relatively new Microsoft .NET Framework, which was first released in 2002. Web Parts was introduced as a class in the SharePoint API. You had to compile your code into assemblies instead of just writing a VBScript object.
This was the start of where Web Parts are today. With SharePoint 2003 you could build scalable, customizable, and personalizable web portals.

Microsoft .NET Framework 2.0 was released in 2005. By then, Microsoft had incorporated and enhanced the Web Part concept into the ASP.NET framework. The Web Part Manager was introduced to manage the Web Parts, which previously had been managed only through the Web Part zones. This new Web Part infrastructure allowed anyone to create their own Web Part-based site, using their own providers for storing customizations. For further reading on the internals of the ASP.NET 2.0 Web Part infrastructure, see ASP.NET Web Parts in Action by Darren Neimke.

SharePoint 2007, Windows SharePoint Services 3.0, and Microsoft Office SharePoint Server 2007 were a huge release. SharePoint had finally become a major player on the portal market. It was based on ASP.NET 2.0 and, therefore, on the ASP.NET built-in Web Part infrastructure. For backwards compatibility reasons, the Web Part class introduced in SharePoint 2003 was preserved and you could use both versions for your Web Parts, both with different abilities.

The development tools for SharePoint and Web Parts, compared to other .NET based projects, were lousy in the beginning and had to rely on the community. The lack of a visual editor of Web Parts made the Web Part development threshold high. The community around SharePoint has been extremely interesting, professional, and entertaining since the release of SharePoint 2007. The early community understood the lack of proper development tools and a number of excellent tools were born such as WSPBuilder by Carsten Keutmann (http://wspbuilder.codeplex.com) and STSDev by Ted Pattison et al. (http://stsdev.codeplex.com).

Today we have SharePoint Foundation 2010 and SharePoint Server 2010. They are based on Microsoft .NET Framework 3.5 Service Pack 1, which essentially is Microsoft .NET Framework 2.0 with a set of extensions. So the baseline is still the same as in SharePoint 2007; we can still use both versions of the Web Parts, but the new one is the Visual Web Parts, which is based on the ASP.NET 2.0 Web Parts. SharePoint 2010, together with Visual Studio 2010, will most certainly affect the lives of us developers. We can now just as easy as in any other .NET project, design, build, and debug our solutions.

“Keep me going strong…”

SharePoint 2010 has the version number of 14 and SharePoint 2007 has version number 12. Why Microsoft skipped version 13 has no clear answer but one of the "tales" says that the reason for skipping number 13 is superstition, just like you have hard to find floor number 13 in any hotels. SharePoint 2003 had version 11, which was chosen to align the versioning number with the Microsoft Office versioning.

With SharePoint 2010 Microsoft has dropped the name Office from the server product. Microsoft Office SharePoint Server 2007 has become Microsoft SharePoint Server 2010. The main reason is said to be that users confused the SharePoint Server product to much with the Office client suite and too many believed that you needed to have the Office client to work with SharePoint, even though the integration works best using the Microsoft Office suite compared to other suites.

Why use Web Parts?

Web Parts are just one, but crucial, part of the huge SharePoint product; without Web Parts, we would not have flexibility in the platform. It is the Web Parts that makes the SharePoint, or other Web Part based portals, interesting for the editors and end-users. Building a new web page or site can be done without programming skills. The editors create their pages and choose Web Parts from the gallery. Then, they customize them and save the page—all done from the browser. The configuration options of the Web Part are defined by the developer who created the Web Part.

Web Parts are easy to use when creating custom applications for SharePoint since they can be self-contained or grouped together. The business logic code and user interface are based on .NET and ASP.NET development, which helps you to get started fast if you are familiar with .NET development.

Since Web Parts share a lot of common features and they are managed in the same way, it is very easy for the end-user to get started with a new Web Part-based application. The end-user knows how to find, add, and edit a
Web Part of a page. This makes the adaptation of new features or applications in your SharePoint environment faster and easier.

Deployments of Web Parts are, thanks to the SharePoint infrastructure, easy and controlled. You can deploy the Web Parts to the site collections of your SharePoint environment that you want your Web Part to be able to run in and even control who can use the Web Parts using the built-in permissions in SharePoint.

**What’s in the box?**

SharePoint 2010 comes with a number of Web Parts that you can use. Which Web Parts that are available in your SharePoint installation, depends on which version of SharePoint you have installed. These Web Parts comes in a couple of different types. The simple Web Parts shows the users tasks, the content of a list, or graph. Then, we have the Web Parts that needs customization, such as the Content Editor Web Part or Forms Web Parts. For more advanced usage there is, for example, the new XsltListView Web Part or the DataForm Web Part, which need to be configured using SharePoint Designer, which can be used to make really advanced applications.

The growing Web Part market offers complete suites or standalone Web Parts that can be downloaded for free or fee. In many cases, getting one of these off-the-shelf products can solve your problem for a much lower price than it would cost to develop a Web Part. But there are always custom designed applications that need to be done, perhaps, in combination with these standard products.

**Customizing Web Parts**

When do we need to create our own Web Parts then? Why do not we just use the DataForm Web Part or create an external list using the Business Connectivity Services? The DataForm Web Part is great for connecting to an external data source or web service. All you have to do is customize the view of the data. SharePoint 2010 also introduces a new way to connect to external data using the External Lists in SharePoint. The External Lists, in combination with the out-of-the-box Web Parts, can be used to create an interface for your custom database in SharePoint using just configuration techniques.

**Business Connectivity Services and External Lists**

Business Connectivity Services (BCS) is the evolution of the SharePoint 2007 Business Data Catalog (BDC) which allows you to connect to external data sources such as databases or web services. These can be visualized in an External List; this list looks and acts as a standard SharePoint List for the end-user but the data still resides in the connected system. SharePoint Designer 2010 is the tool to use when creating external entities and mapping them to the external data sources.

Custom Web Part development takes all of these approaches even one step further, you have full control of what is going to be shown and you can create an interface that lets your user interact with the Web Part in a more advanced and sophisticated way than with the standard Web Parts. Using agile development methods you can even involve the end-users early in the development process and iterate until you have a satisfied customer. The developer has the power to make the Web Part configurable or personalizable for the end-user. Compare that to the DataForm Web Part, where the administrator configures the Web Part and then all the end-user can do is watch and use the result. Almost all configuration of the DataForm Web Part has to be done using the SharePoint Designer application or requires skills in XSLT, which most users do not have access to or do not know.

**Agile development methods**

Agile development has become very popular during the recent years. It is a set of methodologies focusing on some simple principles. These principles include an iterative approach and early and continuous delivery. One of the most popular agile methods is Scrum. Other popular principle for agile methods is Test Driven Development, TDD, and Continuous integration.

For Source Code, Sample Chapters, the Author Forum and other resources, go to [http://www.manning.com/wilen/](http://www.manning.com/wilen/)
As a Web Part developer, you have the full control of the user interface. This is especially important when creating Web Parts for publishing sites, where you might need to control the exact output of the Web Part or if you are using a dynamic AJAX based interface so you need to run client-side scripts.

Building custom Web Parts gives you access to the complete object model of SharePoint and can be used to modify or show content that the user normally would not have seen by using elevated privileges. For example, you might store data in a SharePoint list that is protected by permissions so that the end-users are not allowed to see the list. You can then create a Web Part that uses the System Account context and reads data from the list and presents to the user or lets the user update the data.

Web Parts are mobile

SharePoint 2010 has been updated to support mobile devices better than its successor and it is not just about the way it shows lists and libraries. It now has support to convert your Web Part pages into mobile views and even the Web Parts can be converted to mobile Web Parts using special mobile adapters. If you compare that to a non-code solution built using custom JavaScript, which relies on a powerful browser and perhaps not is adapted to a smaller screen, you will find that building Web Parts and mobile adapters will make your users experience better and useful.

Another interesting aspect of Web Part development is that you can enhance an out-of-the-box Web Part or third-party product. As long as the Web Part is not sealed then you can subclass that Web Part and enhance your version of it and override its default logic.

Web Parts can be powerful, which is not always good. Since you can run whatever .NET code you like in your Web Part, there is a risk that there can be malicious or bad written code that in worst case deletes your information. SharePoint has several mechanisms that help you control those scenarios through policies applied during deployment. With the new Sandboxed solutions, you can even allow Web Parts from less trusted sources run in your SharePoint environment in a special protected process.

Introducing the Web Part infrastructure

Before we go further we need to know some basic facts about web pages and Web Parts in SharePoint. If you have been working with SharePoint or ASP.NET before, this might be old news to you, but nevertheless, it is always good to refresh your memory.

The Web Part Page

A Web Part Page is an ASP.NET page that is inheriting from or using a specific SharePoint page class, the WebPartPage. This class is, in turn, inherited from the standard ASP.NET Page class. The responsibilities for the Web Part Page are:

- Connection to the SharePoint context
- User authentication
- Hosting the Web Part manager, which is defined in the master page
- Hosting the Web Part zones
- Hosting the special zones for Web Part properties and Web Part Gallery

To be able to use Web Parts on a web page, that page needs to be equipped by one Web Part Manager and optionally Web Part Zones. The Web Part manager in SharePoint is defined in the master page, which makes it available in all pages. Figure 2 shows how the building blocks of a Web Part Page are organized. Web Parts can live inside or outside a zone. When a Web Part is outside a zone, that Web Part is called a static Web Part and cannot be customized by the end-user using the web user interface. If a Web Part exists in a zone, then it can be customized and the customization can be persisted so that the customization is retained. It is then called a dynamic Web Part. The Web Part Zones can also allow the end-user to add, remove, or move Web Parts that the user either can upload or find in the Web Part Gallery.
Figure 2 The figure shows the building blocks of a Web Part Page and how they are located in relation to each other.

The Web Part Manager and Web Part Zone objects are defined in the ASP.NET object model but SharePoint uses its own implementation of them. Microsoft .NET Framework has a provider model that allows developers to create their own back-ends for the Web Part storage. The SharePoint product team has created providers for SharePoint and enhanced them for SharePoint usage. The customization and personalization of the Web Parts are stored in the SharePoint content databases using the SharePoint implementation of the provider model.

**The Web Part Manager**

The Web Part Manager is the control that is responsible for handling the different Web Parts on a page. It de-serializes and serializes Web Parts and their customizations from the content database, correctly places them in the Web Part Zones, and optionally connects them. During the Web Part Page lifecycle, the Web Part Manager makes sure to fire and forward all of the events on the different Web Parts. The Web Part Manager is responsible for:

- Persisting properties and customizations of the Web Parts
- Managing the Web Part connections
- Managing the Web Parts in the Web Part zones
- Wiring up the events to the Web Parts
- Managing standard and mobile views

**The Web Part Zones**

The Web Part zone is acting together with the Web Part manager to render out the Web Parts onto the Web Part page and is responsible for creating the chrome, which is made up by borders or styles around the Web Part, as well as rendering the necessary scripts to make the Web Parts react on drag and drop actions. A Web Part zone contains a set of properties that defines the layout of the zone and its responsibilities are to:

- Define areas where the Web Parts can be added
- Define the appearance (i.e., header, footer, and frames) and behavior of the zone
- Define if the Web Parts within the zone can be customized and/or personalized
- Render the Web Parts in the zone

In SharePoint 2010, the Web Part zones and the Wiki content controls have had a major overhaul since the 2007 version and now the Wiki content areas can be used as Web Part zones. Yes, you heard it right! In a Wiki page, you can add your Web Parts into the Wiki content between paragraphs of text or even in the middle of a word (see Figure 3).

For Source Code, Sample Chapters, the Author Forum and other resources, go to [http://www.manning.com/wilen/](http://www.manning.com/wilen/)
Figure 3 The default SharePoint 2010 Team Web is based on Wiki pages. Wiki pages in SharePoint allow you to add Web Parts directly into the Wiki content.

The Web Part

The Web Part itself consists of several different parts, as seen in Figure 4. First of all, it has the content area, which you as developer mostly are focusing on. The content area is where the Web Part is rendering out the content and controls. The Title of the Web Part can be configured through a property of the Web Part, but the layout of it is controlled by mainly the zone.

Figure 4 A Web Part consists of a content area surrounded by the chrome. All Web Parts also have a title and a menu containing a set of verbs. The title and the menu may be configured so they are not shown.

A Web Part has an options menu, typically located in the upper right corner, and this menu contains actions called Verbs. They are an action that belongs to the Web Part type. The action that is executed when chosen is controlled by the developer in the Web Part object. This options menu also contains some other standard Web Part verbs, such as Minimize, Close, Delete, and Edit. Which actions are shown depends on the permissions the user has on the page and how the Web Part is configured.
Web Part Gallery
In SharePoint, each Site Collection has a set of galleries containing different artifacts that are used to build the sites in the site collection. One of these galleries is the Web Part Gallery. This gallery contains all the Web Parts that are available for the editors in that Site Collection. The editors can page pick and choose a Web Part from this gallery and add it to the page. Administrators can add new Web Parts to the gallery through the web interface.

Each Web Part in the gallery has a set of metadata attached to it, such as a group, for grouping Web Parts, default values for the Web Part properties, and permissions.

SharePoint 2010 Pages
SharePoint has two basic page types serving different purposes: application pages and content pages. These page types are based on a single ASP.NET master page which allows SharePoint to have a consistent interface framework.

The pages exist in SharePoint Sites. A SharePoint Site represents a set of lists, libraries, groups, features, and sub-sites. These Sites are grouped into Site Collections, where every Site Collection has a top-level site, which may have one or more sub-sites.

Sites in SharePoint 2010 have a special library called SitePages, which normally contains the content pages of the site, similar to the Pages library that publishing sites use to store the pages. This SitePages library is new to SharePoint 2010. In SharePoint 2007 you either had to create a library on your own or the more commonly used scenario where you put your pages directly into the site. The advantage of the SitePages library is that you can turn on versioning and/or workflows on all your content pages.

Application Pages
Application pages are also known as ".layouts" pages since they exists in the IIS ".layouts" virtual directory. These files are stored in the TEMPLATES\LAYOUTS sub-folder of the SharePoint Root and virtualized as the ".layouts" sub-folder of each SharePoint site. Application pages are most often used as configuration pages for features or applications. The Application Pages cannot host Web Part Zones and, therefore, dynamic Web Parts. Static Web Parts, Web Parts living outside of a zone, can be used. These pages cannot be controlled or edited by the users and should only be used for custom applications. They behave very much like a standard ASPX page. Adding or editing these application pages is done directly on the file in the file system. This is important to be aware of, since in a SharePoint farm, each server has a copy of these file and a changes and additions must be replicated to all servers.

Content Pages
Content pages are the pages in SharePoint that are created while a site is provisioned, by the user or by different triggers. Typical content pages are the default or home pages created in a site. The allitems.aspx list view page and the editform.aspx form page are used to edit list items. These pages can be customized or modified by the end-user using the browser, SharePoint Designer, or using other tools or techniques. There are a few different types of content pages. Content pages used with Web Parts are called Web Part Pages and Wiki Pages. For publishing sites, there is a special type of page called Publishing Page that inherits from the Web Part Page and is specifically designed to have a flexible layout. You can also create your own types of content pages by inheriting from or customizing any of the existing types.

WEB PART PAGES
Web Part Pages are the simplest kind of content page. They are based on the master page, just as all other pages in SharePoint, and are equipped with a set of Web Part Zones. The Web Part Page is used for all form and view pages for lists and libraries. These pages are limited to changes in the Web Part zone and in SharePoint 2010 the use of Wiki Pages should be considered when creating pages.

Wiki Pages
In SharePoint 2010 the default page becomes Wiki pages which have several benefits, compared to previous version where the default page type was a Web Part page. First of all the editing of text and content is easier and more user friendly with the WYSIWYG (what you see is what you get) interface. Secondly the Wiki content areas have been hugely improved and now even acts as Web Part zones. In SharePoint 2007 the Wiki content areas was
just an area where you could have content such as text, links and images with very limited editing possibilities. SharePoint 2010 Wiki content areas can contain text, links, images, even Web Parts and have far better editing tools. The page editor can add and remove Web Parts in the Wiki content and drag the Web Parts around and place them in appropriate position as in Figure 5.

![Wiki pages in SharePoint 2010 can contain Web Parts within the Wiki content.](image)

Figure 5 Wiki pages in SharePoint 2010 can contain Web Parts within the Wiki content.

The Wiki pages, inheriting from the `WikiEditPage` class, always contain a hidden Web Part zone, which is used for storing the Web Parts that exist in the Wiki content areas. During the rendering, the Wiki content is rendered into a buffer and the Wiki content contains special tags where the Web Parts are placed. When the SharePoint finds one of these tags, it renders the Web Part, which it locates in the hidden Web Part zone, into the buffer, and finally this buffer is flushed out to the page.

**PUBLISHING PAGES**

Publishing Pages are most often used in enterprise portals or public-facing web sites. Publishing pages are a Web Part page that uses specific field controls to render the publishing content. The publishing page inherits from the `PublishingLayoutPage`, which inherits from the `WebPartPage` class. In many cases, publishing pages also contains Web Part zones that the page editor may use to add extra information or functionality to the pages. Since SharePoint 2010 now is versioning the customization and personalization state of the Web Part, the usage of Web Parts in publishing pages and sites is enhanced compared to previous SharePoint releases.

**CUSTOM CONTENT PAGES**

Content pages are normally based on one of two classes, both defined in the `Microsoft.SharePoint.WebPartPages` namespace.

- `WebPartPage` - A content page containing Web Part Zones.
- `WikiEditPage` - The new type of content page, introduced with SharePoint 2010, which contains the new Wiki field which can host Web Parts. In the SharePoint 2010, these pages are sometimes called Content Pages (see Figure 6).
Figure 6 The new Silverlight-based Create dialog form can be used to search for and add new items to your site.

These Content Pages cannot by default contain inline server-side code, such as controls and event handlers. Assume that you create a new Content Page in a document library and want to add a submit button that should run some code when clicked. This cannot be done using inline code, without changing the security settings of your SharePoint web application. You can lower the security, but it is not recommended and it exposes your web application for vulnerable code. Instead, you have to two basic options that work equally well but have different purposes:

1. Create a Web Part.
   This Web Part will show the button and handle the click event for you.

2. Subclass the Content page.
   Your new class will have the code that handles the click event and the Content Page ASPX code will define the button.

Using a Web Part for your custom code is the preferred way since you can reuse the logic much easier. If you add the Web Part as static Web Part, which is outside a zone, you can control on which pages the Web Part is available. The easiest way to do this customization is by using SharePoint Designer. If you add it to the page using the web-interface to a zone, then the editors of that page can move, customize, or even delete it, but you will let the page stay in its uncustomized state (if it was uncustomized before).

**PAGE CUSTOMIZATION**

The Content Pages are stored, just like the Web Part customizations, in the Content Database of your SharePoint instance. Compared to the application pages, you do not have to care about replicating changes between the SharePoint servers since the file resides in the database.

This is true in most cases. SharePoint has a mechanism called *page customization*, previously called *Ghosting*. This technique allows the template of the content page to be stored on disk (“ghosted”) and the database keeps a reference to the file on the disk instead. Only when you customize the page (“unghost”) the file, then data will be added to the content database and the reference points to the database instead of the file system. This page customization makes SharePoint perform much better, since it does not have to keep a copy in the database for all provisioned pages, only the ones that the user explicit customized. For example, the editform.aspx for a contacts list uses the version stored on disk for all contacts lists in the farm, unless customized, and saves you a lot of database space and performance. Any updates on the file in the file system will immediately be visible for all uncustomized Contacts list edit forms, for example, when you need to do some changes or when a Service Pack of
SharePoint is applied. As a good practice you should not modify a content page residing in the file system of the SharePoint front-end web servers, since it is hard to find out if and where this file has been customized.

**Master Pages**

A Master Page is the foundation of any page in SharePoint sites or ASP.NET sites. The master page defines the basic layout, structure, and functionality of the page and contains place holders where the pages can host their content. Each time a page is requested by a user, the page is created by merging the master page with the requested page by replacing the place holders in the master page with the content from the page.

SharePoint 2010 mainly uses one master page, called v4.master, compared to the previous version of SharePoint where there was a separate master page for application pages. You can edit the master pages or provide your own—but that is a completely different topic. The default master page is located in the SharePoint root under the `TEMPLATE\GLOBAL` folder.

**Use the SharePoint 2007 user interface in SharePoint 2010**

SharePoint 2010 contains a completely new master page to support the new user interface and web browser compatibility. If you have invested heavily in the SharePoint 2007 UI and want to upgrade your site, it is good to know that the old master pages are still provided with the SharePoint 2010 upgrade and you can at any time select which interface you want to use.

For a Web Part developer, there are a couple of things to be aware of when working with master pages. The default master page delivered with SharePoint 2010 contains the important Web Part Manager object (`SPWebPartManager`), so there is no need to define your own in your content or application pages. It is defined close to the body tag of the page and has the id of `m`.

You cannot add Web Part zones to the master page but you can add Web Parts. These Web Parts will be available on all pages using the master page and they will be static, since they do not exist in a zone. This also means that the Web Parts cannot be customized using the user interface, for instance, when editing a page. You have to use SharePoint Designer 2010 to customize the Web Parts.

**Summary**

This article introduced the world of Web Parts in SharePoint. SharePoint is a platform that can be used to develop and deploy custom applications. For a SharePoint Web Part developer, there are two important tools: Visual Studio 2010 and SharePoint Designer 2010. The first is used to design and build the Web Parts and the latter for configuring Web Parts, pages, and sites.

You do not always have to develop your own Web Parts; there are plenty of configurable Web Parts that can be used to create your solutions. But there are so many situations when you need to develop your own custom Web Part to provide that unique functionality that your customer is requesting or to have full control of your application. Developing Web Parts is not just about writing a piece of code that gets injected into a SharePoint environment; you need to understand how SharePoint uses the Web Parts and how you leverage the features of SharePoint.

For Source Code, Sample Chapters, the Author Forum and other resources, go to [http://www.manning.com/wilen/](http://www.manning.com/wilen/)