Special Report

2

UNDERSTANDING SHAREPOINT JOURNAL

Bjørn Furuknap

What's New for SharePoint 2010 Developers

UNDERSTANDING SHAREPOINT JOURNAL

What's New for SharePoint 2010 Developers

I dedicate this issue to my lovely wife Lena.

Rubina Ranasgt. 10
N-0190 Oslo, Norway
Phone +47 91 39 85 86

http://www.understandingsharepoint.com/journal
journal@understandingsharepoint.com

Credits

About the Author

Bjørn Christoffer Thorsmæhlum Furuknap is a senior solutions architect, published author of *Building the SharePoint User Experience*, speaker, and passionate SharePointaholic. He has been doing software development professionally since 1993 for small companies as well as multinational corporations. He has also been a teacher at a college-level school, teaching programming and development to aspiring students, a job that inspired him to begin teaching what he has learned and learns every day.

About *Understanding SharePoint Journal*

Understanding SharePoint Journal is a periodical published by UnderstandingSharePoint.com. The journal covers few topics in each issue, focusing to teach a deeper understanding of each topic while showing how to use SharePoint in real-life scenarios.

You can read more about *USP Journal*, as well as get other issues and sign up for regular updates, discounts, and previews of upcoming issues, at http://www.understandingsharepoint.com/journal.

Other Credits

A great big thanks to Kim Wimpsett for doing the copyedit. The quality of work in this issue is greatly attributed to her skill.

Table of Contents

Introduction

Asking the right question at the right time to the right people can trigger a landslide.

Welcome to this special report from *USP Journal* on what's new for SharePoint 2010 developers. This issue came about after a number of readers of "Beginning SharePoint Development" (volume 1, issue 5) of *USP Journal*, requested that I write an updated version for SharePoint 2010.

However, much of what you'd learn from developing in SharePoint 2007 also applies to SharePoint 2010. Further, I know a lot of those readers are eager to get started on SharePoint 2010 quickly, and writing a full issue such as "Beginning SharePoint Development" would take a long time.

Thus, I decided that to quell the most urgent needs for information about SharePoint 2010 development, I would write a special and briefer issue that would focus on what is new in SharePoint 2010 rather than starting from scratch.

Even if you haven't read "Beginning SharePoint Development," you'll find lots of valuable information in this report. And, of course, if you haven't read the aforementioned issue, your copy is waiting for you in the virtual bookshelf at http://www.beginningsharepointdevelopment.com/.

Finally, I would like to mention that a large portion of this issue is adapted and updated from the "Introducing SharePoint 2010" issue (volume 2, issue 1). However, even if you have that issue, you'll find that there is updated information and screenshots in those sections that have been repurposed for this report.

I hope you enjoy this report, and as always, I welcome your valuable feedback and comments to journal@understandingsharepoint.com.

.b

Chapter 1

What's New in SharePoint 2010?

A short history of nearly everything

In this chapter, I want to give you a brief overview of the changes in SharePoint 2010 in general. I'll also show you the new SharePoint Designer 2010, which has really taken huge steps forward toward being a natural choice for all SharePoint developers.

Perhaps the biggest surprise is that nothing has changed. Or rather, most of what worked in SharePoint 2007 also works in SharePoint 2010. You'll find the same components in 2010 as you did in 2007, such as features, solutions, lists, templates, content types, event receivers and so on.

Not just will you find the same features but you will also, for the most part, find that the syntax of CAML and .NET code is the same.

So, what has changed drastically?

First, CAML view schema is dead and has been replaced by XSL. Good riddance, if you ask me, but if you haven't worked extensively with views and custom field types, you may not have faced this beast.

CAML view schema was the XML language used to define views in list definitions as well as the rendering of columns in custom field types when displayed in these views.

Sadly, Microsoft never documented CAML view schema very well, and developers also lacked tool support. If you wanted to create your own custom views in a list definition, for example, the only assistance you'd get was, at best, some IntelliSense support in Visual Studio or similar XML editing tools.

XSL has now replaced CAML view schema. This means that rather than having to learn an XML language used only in one particular setting (CAML views), developers should now focus on learning XSL.

If you have already worked with the DataView web part (DVWP) in SharePoint 2007, you have already worked with XSL. DVWP uses XSL to transform the XML returned by various data sources into HTML for display in a browser.

Thus, even if you are not immediately planning to take the step up to SharePoint 2010, you will benefit from learning XSL even now.

> **Note**
>
> If you want to learn more about XSL and the DataView web part, I highly recommend Marc D. Anderson's ebook *Unlocking the Mysteries of the SharePoint Data View Web Part XSL Tags*.
>
> http://www.sympraxisconsulting.com/Unlocking-the-Mysteries-of-the-SharePoint-Data-View-Web-Part-XSL-Tags.aspx

Of the completely new features, I'd like to point out sandbox solutions, the user-deployed method of adding content, including binaries, to a SharePoint site. This opens up a whole new avenue of solution delivery.

In short, you are now able to build assembly-based solutions that do not require server administrator privileges to run so that site owners can deploy and activate these solutions in a safe and stable manner.

In addition, you'll likely appreciate the new client object model that finally allows for "normal" client application development on SharePoint. Previously, you were always relegated to using web services directly, but the client object model abstracts the plumbing required and gives you a .NET object model that you can use to build client-side applications.

You'll also want to notice the new service application model. Taking the place of SharePoint 2007's shared service provider model, Service applications offer a much more scalable and flexible way of sharing services across your farm or even to external farms.

Service applications are individual services such as search, user profiles, Excel services, and others. However, rather than being part of one monolithic application as was the case for shared services, the service applications are individual services that can be turned on and off as needed.

Further, these service applications can expose themselves as WCF services to any location anywhere. If you have no idea what that means, think of it like software as a service where you can expose services in your SharePoint farm to someone else either in your organization or on the Internet.

Companies may, for example, expose their user profile services to each other to allow for cross-company people search. Other organizations may want to build a giant Excel services rendering farm and allow other farms to utilize that service so they do not have to implement it themselves.

Developers can also create new service applications and harness the power of SharePoint to create completely new features either locally or globally. However, the complexity and number of moving pieces may limit widespread custom service applications.

The potential for service applications is enormous.

I'll cover most of the new features for developers in the next chapter of this report, including the three mentioned here.

SharePoint Designer 2010

SharePoint Designer, the offspring of FrontPage and common target for ridicule among programmers, has taken leaps ahead in the newest version.

Although SharePoint Designer 2007 bore a lot of resemblance to FrontPage, SharePoint Designer 2010 embraces the new Office Ribbon–style of user interface, as shown in Figure 1.

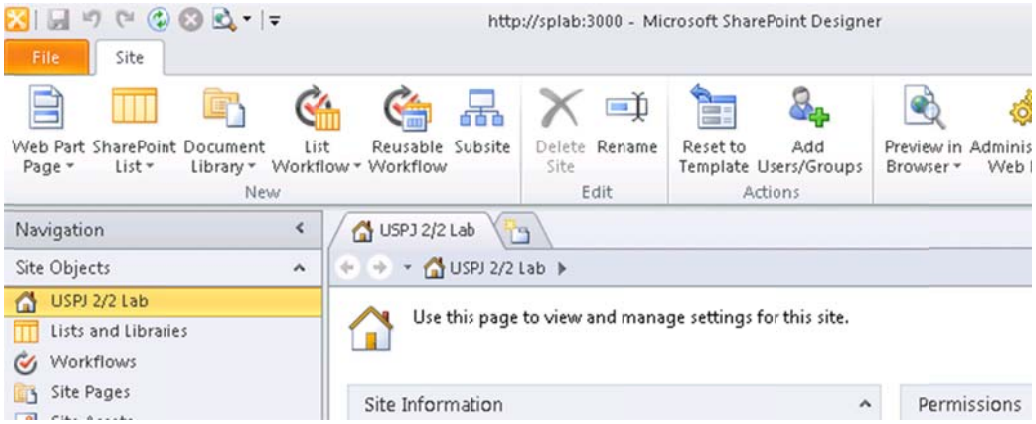

FIGURE 1. SHAREPOINT DESIGNER RIBBON

Besides the obvious user interface changes, you'll also find that you can do a lot more in SharePoint Designer than when you previously had to go to the web interface to do. Examples of this are site columns and content type management.

Another major advance is what has happened to the SharePoint Designer workflow development. In SharePoint Designer 2007, the workflow development experience felt a bit awkward and not really like part of the product.

However, in 2010, the workflow designer is tightly integrated with everything else in the product and offers a much smoother experience, as shown in Figure 2.

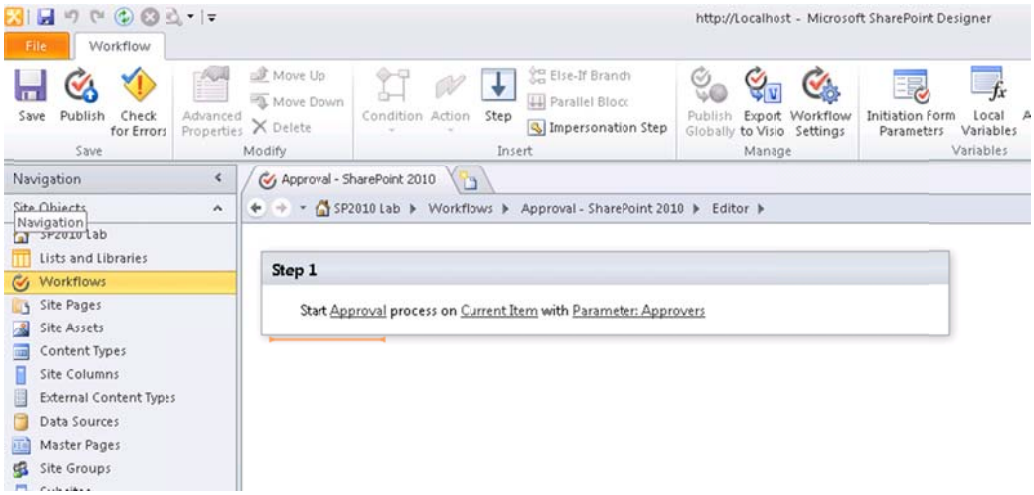

FIGURE 2. SHAREPOINT DESIGNER WORKFLOW SURFACE

As you may also notice in Figure 2, the out-of-the-box workflows that ship with SharePoint Server are now editable. In fact, not just are the workflows editable in SharePoint Designer, but you can even export and import them from Microsoft Visio and export them for editing in Visual Studio 2010 as well.

> **Note**
>
> Although you can export any SharePoint Designer workflow to Visual Studio 2010, you cannot reimport workflows from Visual Studio to SharePoint Designer.

Overall, SharePoint Designer 2010 feels much more like a natural SharePoint management tool now than in the previous version. Even though I am a hard-core programmer by nature, I find that the speed and ease with which I can develop functionality and content in SharePoint Designer is thrilling.

If you want to get the full treatise on SharePoint Designer 2010 workflows, you want to get *USP Journal* volume 2, issue 2, simply titled "SharePoint Designer 2010 Workflows."

http://sharepointdesigner2010workflows.com/

Chapter

2

New Developer Features

What's new, Doc?

In this chapter, I will introduce you to the new features you need to learn as a SharePoint 2010 developer. I'll show you some code, usage examples, and screenshots, as well as demonstrate important new concepts.

SharePoint 2010 New Development Features

SharePoint 2010 promises to be a game changer for business application development. Why? Well, let me tell you about some of the major new development features you can utilize.

Sandbox Solutions

SharePoint 2010 introduces a new feature called *sandbox solutions*. This feature, sometimes called *user solutions,* allows end users and site owners to upload WSP solutions and run them as part of their sites.

As a developer, however, you need to know what these solutions really are. You need to know what sandbox solutions can do, but more important, you need to know what they cannot do.

First, know that sandbox solutions offer only a subset of features that full farm solutions do. For example, sandbox solutions cannot access the file system; they live only in the database. So, the new fancy visual web part in Visual Studio cannot be deployed as a sandbox solution.

> **Note**
>
> There are workarounds for deploying a visual web part as a sandbox solution, by downloading the Visual Studio SharePoint Powertools:
>
> http://visualstudiogallery.msdn.microsoft.com/en-us/8e602a8c-6714-4549-9e95-f3700344b0d9

The following list contains all the element types that sandbox solutions can contain:

- ContentType
- Field
- CustomAction
- Modules
- ListInstance
- ListTemplate
- Receivers
- WebTemplate
- WorkflowAssociation
- PropertyBag
- WorkflowActions

Some of these elements are new to SharePoint 2010, and we'll explore some of them later in this chapter.

Another important aspect of sandbox solutions is that you can control their resource use. Any assemblies in a sandbox solution run in a special process that monitors the solution's use of CPU, memory, network, database queries, and other factors.

Using the resources counts toward a point system, and if a site uses more than an allocated amount of such resources, no sandbox solutions will run within that site for the next 24 hours.

Note

When you are developing sandbox solutions, you must remember to attach to the special user code process and not the regular w3wp.exe process.

My good friend Sahil Malik has written a comprehensive blog series on sandbox solutions, starting at http://blah.winsmarts.com/2009-12-SharePoint_2010_Sandboxed_Solutions__The_Definitive_Guide.aspx.

Developer Dashboard

Speaking of resource management, have you heard about the *developer dashboard*? If not, this will certainly excite you.

The developer dashboard is a debugging mechanism, giving you detailed information about the resources used during the execution of a page. Take a look at Figure 3, which shows the developer dashboard for the default Central Administration front page.

FIGURE 3. DEVELOPER DASHBOARD

Although the text may be too small to read the details in Figure 3, I can see that the page request took 136.67 milliseconds, and I can also see a breakdown of which part of the page life cycle took what amount of those 136.67 milliseconds.

Not just that, but you get an overview of all the database calls performed by the page. Clicking any of the queries opens a detail view, shown in Figure 4, with the details of the query, including the SQL query and the stack trace.

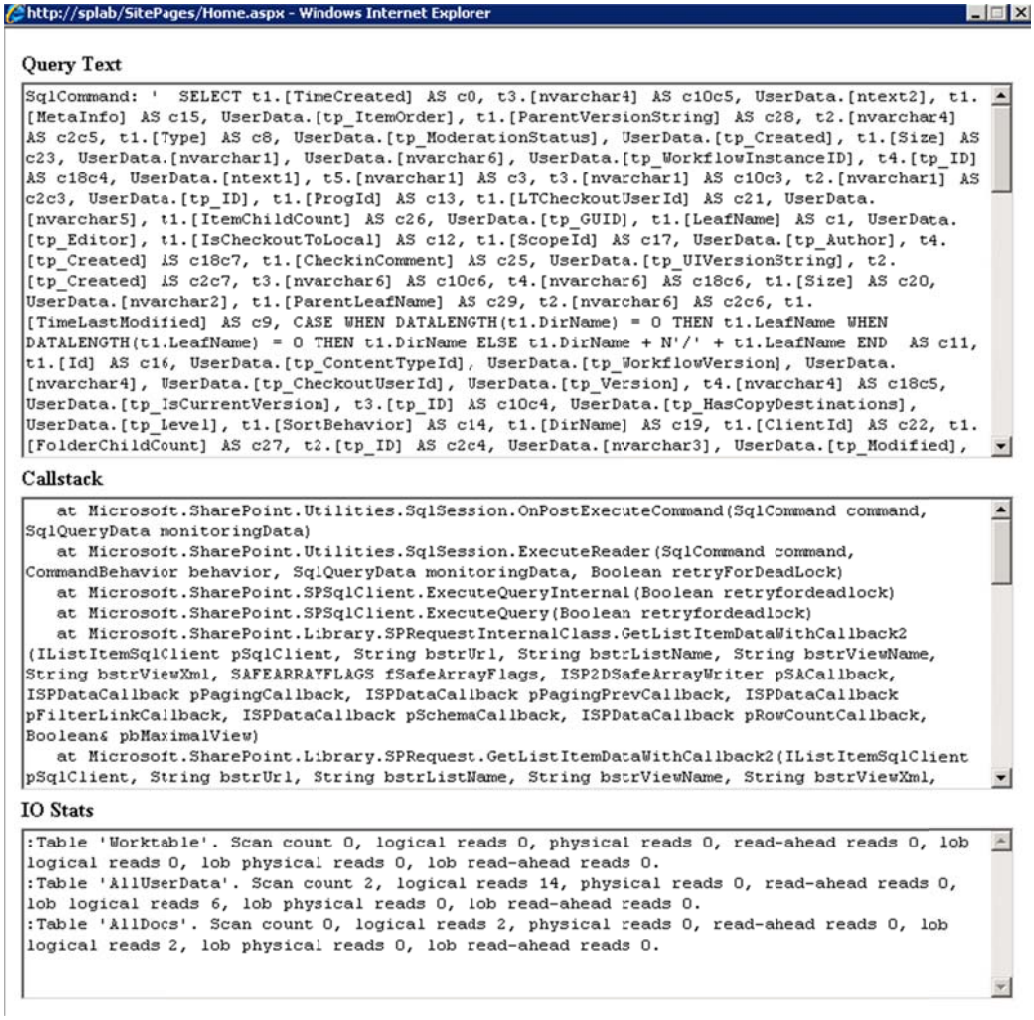

FIGURE 4. DASHBOARD QUERY DETAILS

OK, the design may not impress you if you are a designer, but as a developer, I love the plain-text output in these textboxes.

Enable the Developer Dashboard

The developer dashboard has three display modes. The "on" mode will display the developer dashboard for every page, "ondemand" will add an icon next to the user menu, and "off" will prevent display completely.

You can enable or disable the developer dashboard using different methods, for example using PowerShell, custom code, or even the now quickly aging stsadm.exe command:

```
stsadm -o setproperty -pn developer-dashboard -pv on
```

The corresponding PowerShell commands would be as follows:

```
$dashboard =
[Microsoft.SharePoint.Administration.SPWebService]::ContentService.DeveloperD
ashboardSettings;

$dashboard.DisplayLevel = 'OnDemand';

$dashboard.Update();
```

After setting this property, every page will display the developer dashboard at the bottom of the page. However, this may not be practical, so I recommend leaving the setting as "ondemand" so that you can turn it on only when you need it and then turn it to off once your system enters production.

Client Object Model

Here's a brilliant new and major feature of SharePoint 2010. The *client object model* is made for creating, well, client applications. This includes console applications, JavaScript web pages, and Silverlight applications.

The client object model offers only a subset of the full server object model. For example, there are no objects above the site collection object (SPSite, or Site as it is called in the client object model).

In addition, the client object model works somewhat differently than the server object model. The most important changes are how you access data as well as how you update data.

In a client object model application, you start with a ClientContext object, which links your code to a SharePoint site. Then, you create an object representing what kind of object you want to get back, either a site or a web.

> **Note**
>
> Most client object model objects have the same name as the server object model equivalent, without the *SP* in front. So, the client object model equivalent of an SPWeb is a Web.

You then call the Load method, sending the method the object you want. Finally, you call the ExecuteQuery method, which is actually the first time the application goes out to the SharePoint site.

Once the ExecuteQuery method finishes executing, your object is ready for use. Here is a simple example of how you would use the client object model:

```
ClientContext context = new ClientContext("http://sp2010lab-02:2000/");
Web contextWeb = context.Web;
context.Load(contextWeb);
context.ExecuteQuery();
Console.WriteLine(contextWeb.Title);
Console.ReadLine();
```

Oh, but how about getting objects such as lists? Well, you need to wrap your head around the way the client object model works with its context, loads specific objects, and executes queries.

To get the Announcement list from a standard Team Site, for example, you would need to specify which list you want and create client objects representing the list. Then, you send the List object into the ClientContext.Load method instead.

```
ClientContext context = new ClientContext("http://sp2010lab-02:2000/");
Web contextWeb = context.Web;
List announcementList = contextWeb.Lists.GetByTitle("Announcements");

context.Load(announcementList);
context.ExecuteQuery();

Console.WriteLine("The Announcements list contains " + ↵
announcementList.ItemCount + " item(s)");
```

This code, however, does not give you access to any properties of any parent objects, such as the list collection or the web. If you want that, you need to load those objects to the ClientContext prior to calling ExecuteQuery as well. Luckily, you can add multiple objects to the ClientContext and call the ExecuteQuery method only once:

```
ClientContext context = new ClientContext("http://sp2010lab-02:2000/");
Web contextWeb = context.Web;
List announcementList = contextWeb.Lists.GetByTitle("Announcements");

context.Load(contextWeb); // Load web as well
context.Load(announcementList);
context.ExecuteQuery();

Console.WriteLine("The Announcements list of " + contextWeb.Title + ↵
    " contains " + announcementList.ItemCount + " item(s)");
```

Note

There is one more option in the client object model, though, and that is to pass in a LINQ query rather than specifying the objects you want.

However, confusingly, the LINQ queries you pass in are not LINQ to SharePoint queries but rather LINQ to Objects queries. LINQ to SharePoint is available for the server object model only.

I'll tell you more about LINQ to SharePoint later in this chapter.

The good news is that, besides the somewhat complex method of getting your desired objects, working with the client object model is very similar to working with the regular server object model. You will find most of the objects you normally use, such as lists (List), items (ListItem), content types (ContentType), and so on.

You can also work with these objects almost the same way as you do in the server object model. However, you must remember to call ClientContext.ExecuteQuery after you have made changes to any objects, as shown here:

```
announcementList.Title = "Modified announcements list";
announcementList.Update();

context.ExecuteQuery();
```

The code will modify the list only when the ExecuteQuery method runs, resulting in a modified list, as shown in Figure 5.

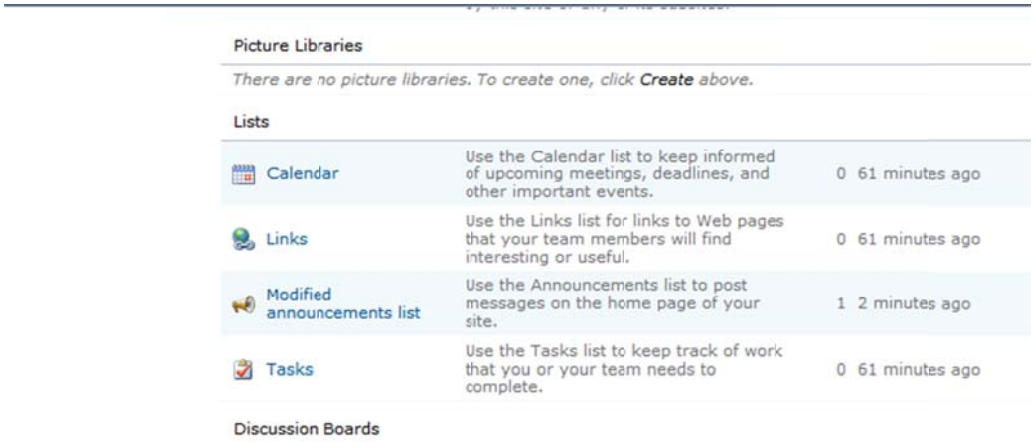

FIGURE 5. MODIFIED ANNOUNCEMENTS LIST

If you have made multiple changes to objects loaded in the ClientContext, all changes are updated with just a single call to the ExecuteQuery method.

It may take a bit of getting used to, but the client object model promises great things for the SharePoint 2010 developer, especially for those who work with the user experience.

REST and WCF Data Services

The ADO.NET data services, in the future known as WCF Data Services, are another new data access feature. What this brings to the table is a fabulous new data manipulation format, namely, REST.

REST is a URL-accessed, feeds-based data retrieval protocol that allows you to create queries and update data using standard HTTP requests and posts. What's even better is that these services are now included and supported fully for all SharePoint lists.

REST uses ATOM feeds for data delivery and manipulation. Since ATOM is an open standard, this provides a great way for third-party developers to access and manipulate list data using well-known protocols.

I think the easiest way to explain REST is to look at some examples.

> **Note**
>
> As of this writing, you need to install the ADO.NET Data Services. If you do not, you will get a 404 error message when you try accessing the following URLs.
>
> You can get a working copy for download from http://www.microsoft.com/downloads/details.aspx?FamilyID=a71060eb-454e-4475-81a6-e9552b1034fc&displaylang=en, but keep an eye out for any updates, because this is a community technology preview (CTP).

You can get the default list data from ListData.svc, located in the _vti_bin virtual folder of each site. In my example, this means the URL is http://sp2010lab-02:2000/_vti_bin/ListData.svc/.

Opening this URL gives you an overview of all the lists in your site, in ATOM feed format, as shown in Figure 6.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <service xml:base="http://sp2010lab-02:2000/_vti_bin/ListData.svc/"
    xmlns:atom="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
    xmlns="http://www.w3.org/2007/app">
  - <workspace>
      <atom:title>Default</atom:title>
    - <collection href="Attachments">
        <atom:title>Attachments</atom:title>
      </collection>
    - <collection href="Calendar">
        <atom:title>Calendar</atom:title>
      </collection>
    - <collection href="CalendarCategory">
        <atom:title>CalendarCategory</atom:title>
      </collection>
    - <collection href="ConvertedForms">
        <atom:title>ConvertedForms</atom:title>
      </collection>
    - <collection href="CustomizedReports">
        <atom:title>CustomizedReports</atom:title>
```

FIGURE 6. LISTDATA.SVC DEFAULT OUTPUT

You can then access data in any of the lists using a refined URL syntax. For example, if I want to take a look at the Modified Announcements List (remember, I modified it earlier in this chapter), you simply add the list name to the URL as such:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList

This will produce a consumable feed of content in the list, as shown in Figure 7.

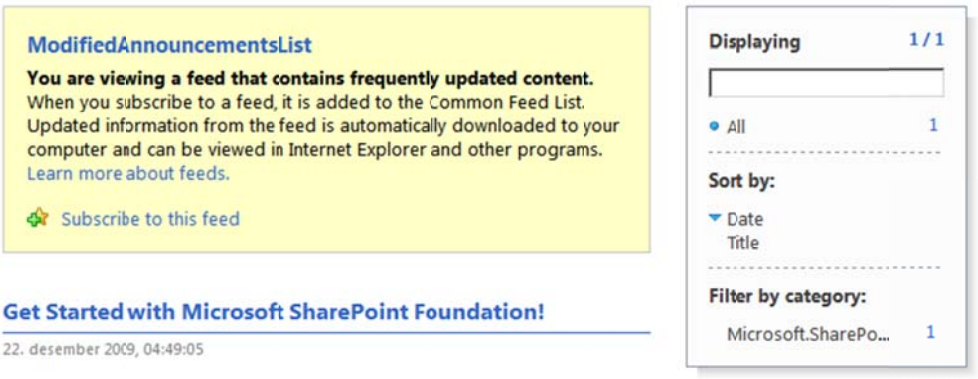

FIGURE 7. MODIFIED ANNOUNCEMENTS LIST WEB FEED

If you are using IE, you can turn off the web feed reader view and display the underlying XML data by going to Tools→Internet Options, clicking the Content tab, and then clicking the Settings button in the Feeds and Web Slices section. Once open, deselect the "Turn on feed reading view" checkbox, as shown in Figure 8.

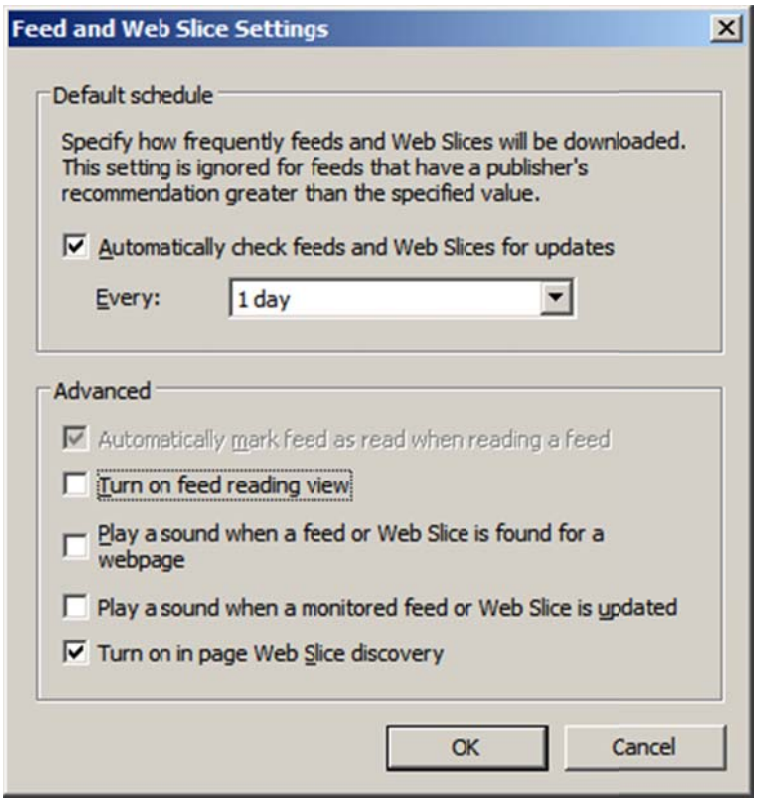

FIGURE 8. DISABLING FEED READING VIEW

Hit OK until you are back in IE, and then restart IE before reopening the URL.

You can drill deeper down into the content, still using the URL, by addressing individual items in the list. The URL format is as such:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList(1)

Note the parentheses at the end, used for accessing the first item in the list. The list, by the way, uses a one-based index, so there is no item at (0).

You can even get individual properties from an item using the following format:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList(1)/Title

But Wait, There's More!

There's more to REST-based queries, though. Way more.

You can filter items using the URL, filtering on both properties and number of items. These filters use query string options, which are commands prefixed by $. For example, the query string option for returning a given number of items is $top, so if you wanted to retrieve only the first five items of a list, you would do something like this:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList?$top=5

Ordering is done in the same manner, using the $orderby query string option, as such:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList?$top=5&$orderby=Modified

This query would return the top five items ordered by the modified column.

You can also skip items in a list using the $skip query string option. This allows for easy paging, for example using a query string like this:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList?$top=5&$orderby=Modified&$skip=5

This query would select the top five announcements on page 2 of the list sorted by the Modified column. In other words, it would select items 6–10 of the list.

Finally, and here's what will blow your mind, you can query related lists in the same query, using the $expand query string option. Let's say our announcements list has a lookup column named Category, pointing to a list of categories, and that each of the categories has a manager.

Let's further say that you want to retrieve the announcement titles along with the name of the manager for the category in which the announcement is posted.

In the "old days" of SharePoint 2007, this means you would need to create two queries with a lot of lookups. With REST in SharePoint 2010, however, you can now simply say something like this:

http://sp2010lab-02:2000/_vti_bin/ListData.svc/ModifiedAnnouncementsList?$expand=Category

This will return a result containing both the announcements and, for each announcement, the corresponding category, including the name of the manager, as shown in Figure 9.

FIGURE 9. RECURSIVE REST QUERY RESULTS

> **Note**
>
> If you want to learn a lot more about REST and WCF Data Services, take a look at the following MSDN article:
>
> http://msdn.microsoft.com/en-us/library/cc907912.aspx

LINQ to SharePoint

Since we are talking about data retrieval, it makes sense to look at another great advance in SharePoint 2010.

LINQ (Language Integrated Query, if you need to full name) is a query language used to get SQL-like queries inside a .NET programming language. It takes a bit of getting used to, but it is incredibly powerful for retrieving and filtering data.

I'm not going to teach you anything fancy in LINQ but, rather, give you a brief overview of how LINQ in SharePoint will work and tell you what kinds of problems that LINQ to SharePoint will work.

One problem facing SharePoint developers is the inherent lack of strong typing of data. For example, if you retrieve an SPListItem object, you can only access the columns of that objects using an indexer such as item["Title"].

This approach is error-prone and complex. Consider what happens if you misspell the column name. Also, you cannot easily see what kind of data resides in the column, so you need to manually check the field type if you want to treat the data in a strongly typed manner.

Of course, you can circumvent the problem to some extent by using custom classes and abstract the SPListItem access. However, since SPListItem is sealed, you cannot inherit from that class, so in effect you end up wrapping all the methods and properties you want exposed and lose the rest.

Then there is the problem of querying data. The only language SharePoint understands is CAML, and since CAML tool support is rather limited, you end up with static queries and no strong typed objects, again making your code error prone and complex to manage.

LINQ to SharePoint solves these two problems by providing strongly typed access to SharePoint data and by giving you an abstraction from CAML queries. In addition, the LINQ syntax is less verbose, and your code is thus easier to read, write, and maintain.

> **Note**
>
> Behind the scenes, LINQ to SharePoint still creates CAML, but you don't need to see any of it.

So, where you would previously write something like this:

```
// Using the object model
Console.WriteLine("Using Object Model:");
using (SPSite site = new SPSite("http://sp2010lab-02:2000/"))
{
    using (SPWeb web = site.OpenWeb())
    {
        SPList announcementsList =
            web.Lists["Modified Announcements List"];
        SPQuery query = new SPQuery();
        query.Query = @"
<Where>
<Eq>
<FieldRef Name='Category' />
<Value Type='Lookup'>Organization</Value>
</Eq>
</Where>                    ";
        SPListItemCollection announcements =
            announcementsList.GetItems(query);

        foreach (SPListItem announcement in announcements)
        {
            // Hope we spelled the column names right!
```

```
            Console.WriteLine(announcement["Title"]);
            Console.WriteLine();
            Console.WriteLine(announcement["Body"]);
            Console.WriteLine("----------------");

        }
    }
}
```

in LINQ to SharePoint, you would write something like this:

```
// Using LINQ
Console.WriteLine("Using Linq:");

USPJLabDataContext context = new USPJLabDataContext("http://sp2010lab-
02:2000/");

EntityList<ModifiedAnnouncementsListAnnouncement> announcementsLinq =
context.GetList<ModifiedAnnouncementsListAnnouncement>("Modified
Announcements List");

var organizationAnnouncements =
    from announcement in announcementsLinq
    where announcement.Category.Title == "Organization"
    select announcement;

foreach (var announcement in organizationAnnouncements)
{
    Console.WriteLine(announcement.Title);
    Console.WriteLine();
    Console.WriteLine(announcement.Body);
    Console.WriteLine("--------------------");
}
```

OK, the lines may be a bit longer, but the code is a lot easier to understand.

The results of these two code pieces are the same, as shown in Figure 10.

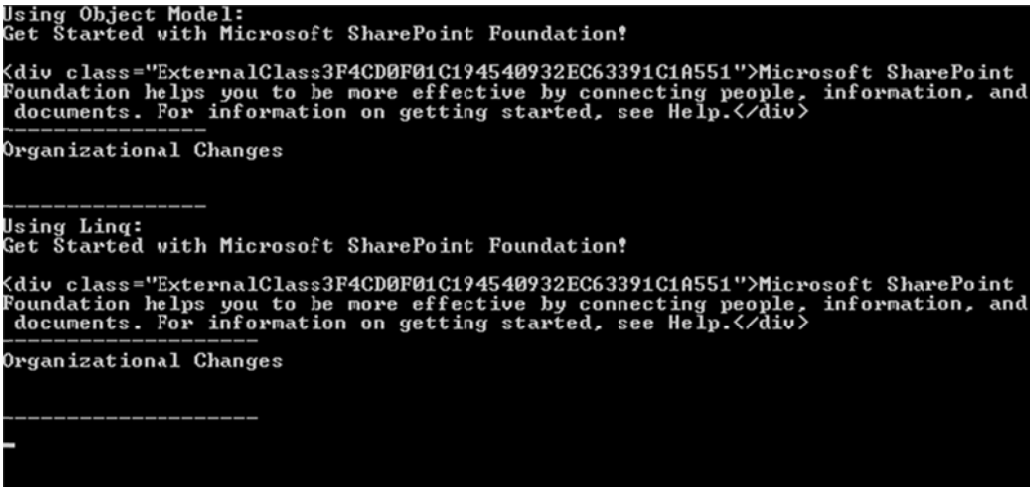

FIGURE 10. OBJECT MODEL VS. LINQ TO SHAREPOINT

Notice the filtering I'm using in the LINQ query:

```
where announcement.Category.Title == "Organization"
```

As you can see, I'm here actually getting a recursive query to the Category object to which the announcement is linked. You can even get multiple levels of recursion, so that if there was a second lookup column from the Categories list to a Department list, you could write something like this:

```
var organizationAnnouncements =
    from announcement in announcementsLinq
    where announcement.Category.Department.Title == "Some Department"
    select announcement;
```

However, the true beauty of LINQ comes from the strong typing and subsequent IntelliSense support for querying data. Look at Figure 11 to see what happens while I type the LINQ query.

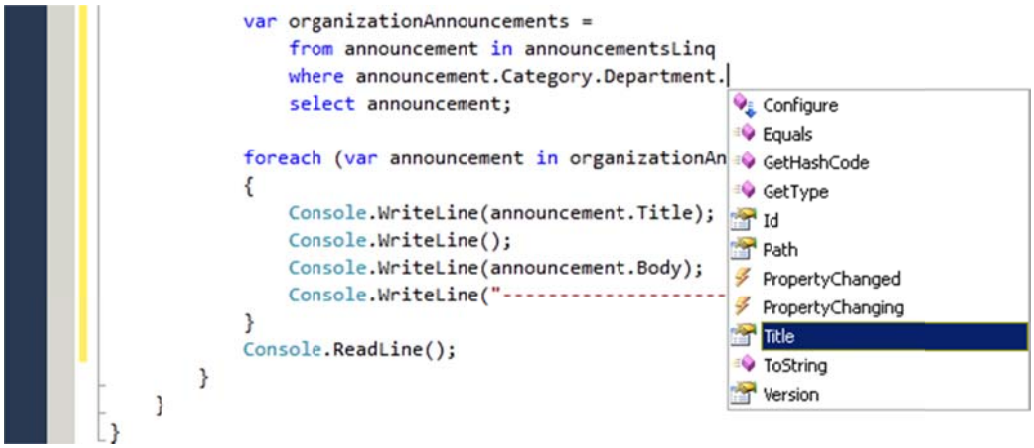

FIGURE 11. INTELLISENSE FOR LINQ

⌘ Tip

Note

LINQ to SharePoint is available only for server-side code. You may find that odd, since the client object model includes options for loading data using LINQ. However, the LINQ used in the client object model is LINQ to Objects, not LINQ to SharePoint.

This is an important distinction.

For more information about LINQ to SharePoint, check out MSDN at http://msdn.microsoft.com/en-us/library/ee537339(office.14).aspx.

PowerShell

PowerShell has sadly become mainly an administrator's tool. However, even developers, in all our code-centricity, should sacrifice considerable time learning this powerful tool.

PowerShell, in case you don't know, is the command prompt on steroids. However, whereas the command prompt was simply a tool to run other programs, PowerShell allows you access to the entire API of virtually anything you want, as long as it is .NET.

This includes SharePoint, and that is why PowerShell is now also for developers' fun and profit.

PowerShell sports an somewhat intelligent way of accessing APIs method calls and storing variables, more or less exactly what you would do in a console application. And once you get used to the syntax, PowerShell becomes a good alternative to writing one-off console applications to solve trivial tasks, as illustrated in Figure 12.

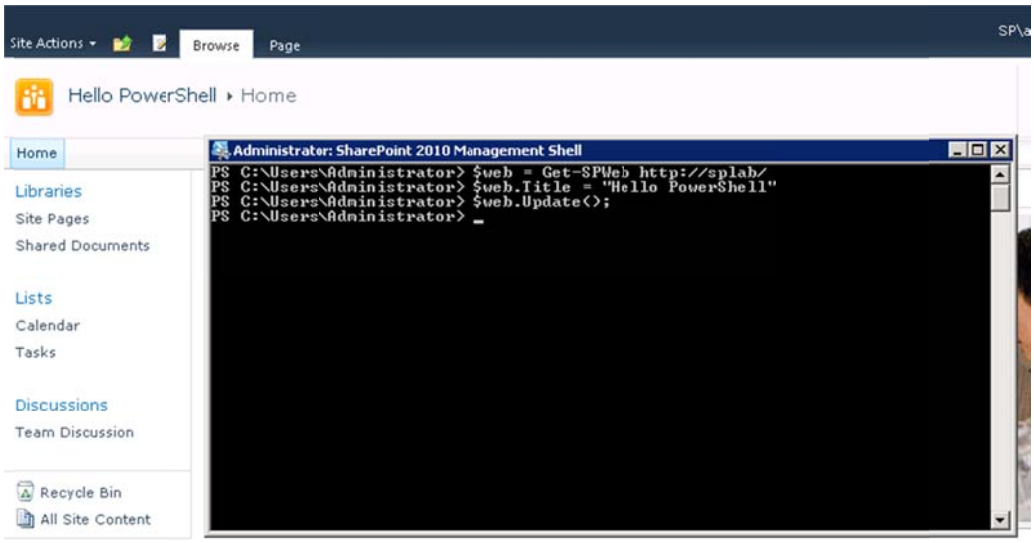

FIGURE 12. USING POWERSHELL

Further, you can create PowerShell scripts so you don't need to type out these commands yourself. In essence, this would mimic creating a console application but without the need to build and run the actual application.

However, where PowerShell really shines for us as developers is when we start looking at PowerShell extensibility. PowerShell operates by running what are known as commandlets, or cmdlets. In the previous example, Get-SPWeb is such a cmdlet.

These cmdlets are built using .NET and are essentially just .NET classes. As developers, we can extend PowerShell by creating custom cmdlets to suit our needs. A good scenario is when we need to create setup scripts for administrators, or we need to provide specialized functionality for the solutions we build.

A specific example is a cmdlet I built for a client a few months back that allowed for importing of user profile fields into the User Profile service application from an Excel sheet—a feature that isn't delivered out-of-the-box.

> **Note**
>
> Sadly, I can't share the code for that cmdlet.

Visual Development

Even while we developers normally leave any visual development to designers, it pays to know what is possible.

SharePoint 2010 Ribbon

The first thing you should learn is the Ribbon. If you are an existing SharePoint developer, you will be pleased to know that the Ribbon consists of CustomAction elements.

Let's jump in and take a look at some examples.

The basic principle of a Ribbon element is an augmented CustomAction, so let's start there.

```
<CustomAction Id="USPJRibbonDemo" Location="CommandUI.Ribbon"
RegistrationType="List" RegistrationId="101"   Title="USPJ Demo">
</CustomAction>
```

Note especially the Location value. This must be CommandUI.Ribbon. Beyond that, RegistrationType and RegistrationId maps the Ribbon element to the right list.

The next bit inside the CustomAction element is a CommandUIExtension element, telling SharePoint we want to manipulate the user interface.

A CommandUIExtension can contain two child elements, CommandUIDefinitions and CommandUIHandlers, which in turn contain one or more child elements.

The CommandUIHandlers is the simpler of the two and handles the interaction with the user in the form of CommandUIHandler elements. Here is one example:

```
<CommandUIHandlers>
  <CommandUIHandler
    Command="USPJRibbonDemo"
    CommandAction="javascript:alert('Introducing SharePoint 2010');" />
</CommandUIHandlers>
```

The command is the name you use to hook your buttons to specific actions.

Now, in the CommandUIDefinitions element is where things get complex. The only legal child element, CommandUIDefinition, can contain a highly complex mixture of ribbons, tabs, groups, and buttons, and even the buttons come in a wide variety of flavors. You can have a regular button, checkboxes, menus, comboboxes, flyout anchor buttons, galleries, split buttons, and a whole range of other types.

To show you one example, here is a simple button to call the previous CommandUIHandler from the New section of the Ribbon's Documents tab of a document library:

```
<CommandUIDefinition
  Location="Ribbon.Documents.New.Controls._children">
  <Button
    Id="Ribbon.Documents.New.Controls.USPJRibbonDemo"
    Description="Some Description"
    Command="USPJRibbonDemo"
    LabelText="USPJ Test"
    Sequence="1"
    Image32by32="/_layouts/images/placeholder32x32.png"
    Image16by16="/_layouts/images/placeholder16x16.png"
    ToolTipDescription="Tool tip!"
    TemplateAlias="o1" />
</CommandUIDefinition>
```

Figure 13 shows the result, including the JavaScript pop-up.

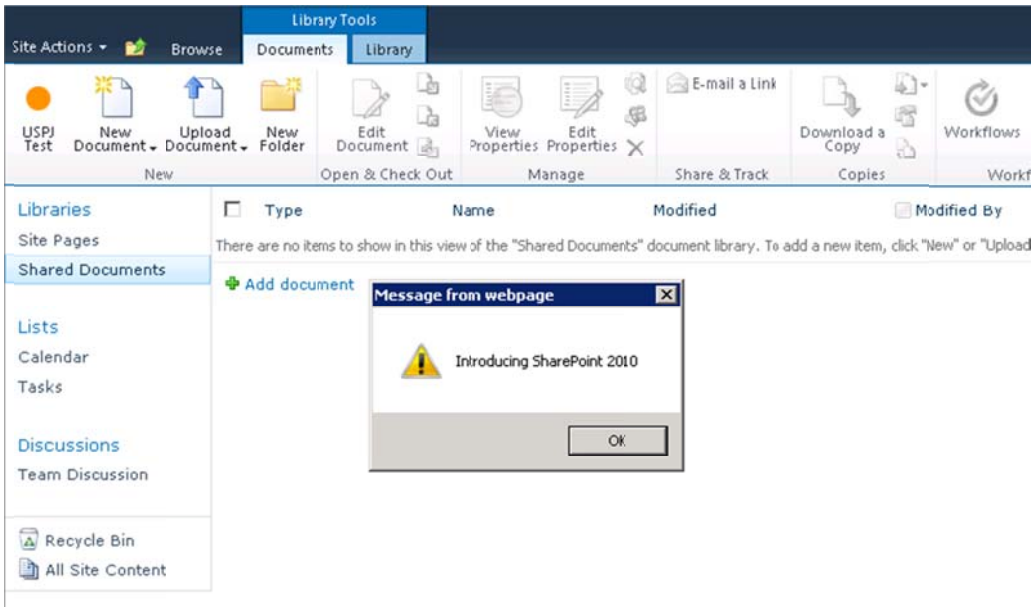

FIGURE 13. CUSTOM RIBBON BUTTON

> **Note**
>
> To read more about Ribbon customization, take a look at this MSDN article:
>
> http://msdn.microsoft.com/en-us/library/ee534970(office.14).aspx

User Feedback

You should know one more thing about the visual development options—or, rather, three things, but they are closely related.

SharePoint 2010 ships with a notification function that pops up a small configurable box at the right of the screen, at least by default. You create these notifications with a simple JavaScript method call, for example inside a Ribbon button.

```
<CommandUIHandler
  Command="USPJNotificationDemo"
  CommandAction="javascript:SP.UI.Notify.addNotification('Hello
Notification', false, 'Tool top', null);" />
```

In addition, you have a status bar located right below the Ribbon, and you can create messages for the user using yet another JavaScript call. For example:

```
<CommandUIHandler
  Command="USPJStatusbarDemo"
  CommandAction="javascript:SP.UI.Status.addStatus('Hello Status Bar',
'You may want to &lt;a href=&quot;#&quot;&gt;click&lt;/a&gt; me',
false);" />
```

> **Note**
>
> Since this is XML, you need to escape all quotes used in scripts as well as the < and > characters.

Figure 14 shows the notification box and the status bar.

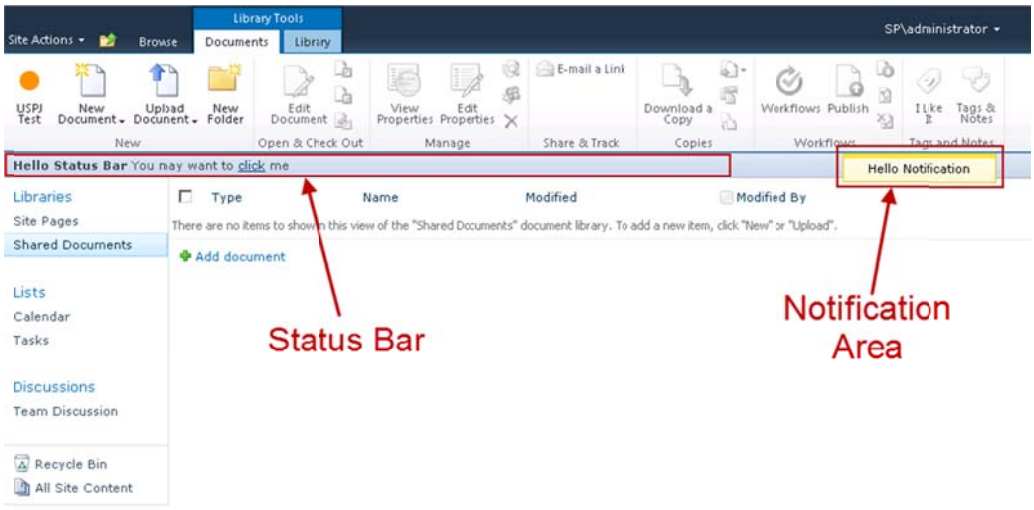

FIGURE 14. NOTIFICATION AND STATUS BAR

The final thing in user feedback that you need to learn is the dialog framework. The dialog framework is responsible for those fancy modal pop-up dialog thingies in SharePoint 2010. You can create these dialog boxes yourself using the dialog framework.

Creating a dialog box is simple enough but is slightly different from notifications and status updates. Rather than sending multiple parameters into the method, you must create an options variable and pass that into the method as such:

```
<CommandUIHandler
  Command="USPJDialogDemo"
  CommandAction="javascript:
  var options = {
  url: &amp;quot;http://www.understandingsharepoint.com/journal&amp;quot;
  };
  SP.UI.ModalDialog.showModalDialog(options);" />
</CommandUIHandlers>
```

The options available allow you to specify the size, the title, whether the box should be closable, the callback method, and so on.

I won't go into more details here since I'm running out of space, but check out the companion solution USPJRibbonDemo.zip, which contains a Visual Studio 2010 project with the code for these Ribbon and user feedback examples, and feel free to experiment with them.

Business Connectivity Services

Have you heard of the Business Data Catalog (BDC)? If not, you can safely count your blessings, for they are many.

BDC is a feature of SharePoint 2007 that allows you to spend weeks of development time to show data that you could just as easily display using a web part that would take 10 minutes to write.

Of course, in a web part, you could enable read, write, and updates of the data in another 30 minutes, whereas with BDC you would fall flat on your face. And, of course, with a web part, you could just connect to the data, where BDC forces you to create a static mapping between entities in your data source and something somewhat fuzzy in SharePoint.

After coming to its senses somewhere between 2007 and now, Microsoft has decided that BDC may not be the best idea since sliced bread. Thank all known deities that BDC dies in SharePoint 2010, and may it be a slow and painful death, followed by eons of suffering and eternal damnation.

Perhaps I'm exaggerating a bit here, but you get the point: SharePoint 2007 BDC is just plain awful.

But you may still want to use your outside data source in SharePoint, and with SharePoint 2010, this becomes a whole lot easier.

> **Note**
>
> Curing cancer with a box of matches, a pack of chewing gum, and a dented license plate is easier than configuring BDC.

In SharePoint 2010, you get a new feature called Business Connectivity Services, which is much better. Here are some of the main reasons and features of BCS.

First, BCS is part of SharePoint Foundation 2010 and does not require SharePoint Server 2010. That means you don't need to shell out the big bucks for the paid version of SharePoint, at least not to get this feature.

Second, BCS offers two-way data access, so you can use your SharePoint interface to work with data in external data sources. BCS supports the full CRUD set of operations (create, read, update, and delete).

Third, you can create BCS connections using SharePoint Designer 2010, so you don't need to write all the code yourself, at least for relatively simple systems.

Fourth, you can connect BCS data to regular SharePoint data and show both types of data in a single view. Leave your customer database in an external database, but show each SharePoint-hosted project with that customer data included.

Yeah, I'm sounding like a marketing engine, but it's really that good.

You should know, though, that despite the awesome advancement over the previous version, I'm not going to walk you through getting BCS up and running. Because of the complexity of all the moving parts, the only realistically feasible way of creating a BCS application is using Visual Studio.

Service Application Model

Another very complex feature, as least from a development perspective, is the new SharePoint 2010 service application model. Again, this replaces a MOSS 2007 feature, and again, this is included in the free version of SharePoint.

The service application model replaces the Shared Service Provider and improves on that model as well. In SharePoint 2010, shared applications now live their own merry lives without dependencies on other services.

I've previously explained the details of why the service application model is so much better, so refer to Chapter 1 if you need your memory refreshed. As I mentioned in that chapter, developers can create custom service applications.

Before you go all crazy with imagination, you need to know that creating service applications from scratch is very tedious. The amount of moving parts is enormous, and although the individual pieces may not be daunting, the fact that there are so many of them makes the task seemingly hopeless.

Don't worry, though, because despite what many people believe, custom service applications are really fairly straightforward once you understand how they work.

> **Note**
>
> There will be an upcoming journal issue on building custom service applications. Sign up for the mailing list, and I'll tell you when. ☺
>
> http://www.understandingsharepoint.com/journal/uspj-mailing-list

Upgrading Code

In Chapter 1, I mentioned that upgrading code is very easy. In general, you need to swap the referenced DLL from version 12 to version 14, rebuild your WSP, and you're off.

Of course, this does not utilize any of the new features of SharePoint 2010, but that may not be what you need. I mean, if your custom code solves your business needs today, it is unlikely that a new version of SharePoint will change your business.

With a proper methodology then, you can keep your current solutions with a bit of rebuild and then keep extending your solutions as you need new features.

However, despite the apparent ease of migrating code up a version, there are pitfalls. You need to test your code thoroughly, and you need to verify whether any object model features you use have been deprecated.

For example, take a look at the previous model for creating a new profile property for the User Profile shared service in MOSS:

```
UserProfileManager profileManager = new UserProfileManager();
Property newProp = profileManager.Properties.Create(false);
newProp.Name = "MyProp";
newProp.DisplayName = "My Property";
newProp.Type = "Boolean";
newProp.IsVisibleOnEditor = true;
newProp.IsUserEditable = true;
newProp.Commit();
```

Neat and intuitive, right? Well, in the SharePoint 2010 User Profile service, you can no longer manipulate the properties of the UserProfileManager. Instead, you need to use a property manager. Or rather, you need to use three various property managers.

In SharePoint 2010, the user profiles aren't really user profiles, but general profiles, because you can have people, organization, and group profiles. In addition, people profiles are divided into various subtypes, for example Employee, Intern, Consultant, and so on.

So, you really have three levels of profiles rather than the old style of only a single level. Rather than working with a UserProfileManager, you need to work with the following:

- A Core Property Manager
- A Profile Type Property Manager
- A Profile Subtype Property Manager

To make matters worse, each of these levels defines various properties of, well, properties. For example, the Name property is set in the core property. The IsVisibleOnEditor is set for the profile type property, while the IsUserEditable property is set on the profile subtype property.

Confused yet? No? Well, take a look at the code needed to do the previous property adding, this time using SharePoint 2010 code:

```
SPSite currentSite = new SPSite("http://mosslab-01:2000/");
SPServiceContext context =
    SPServiceContext.GetContext(currentSite);
UserProfileConfigManager upConfigManager =
    new UserProfileConfigManager(context);
ProfilePropertyManager profilePropManager =
    upConfigManager.ProfilePropertyManager;

// Get all the various levels of property managers
CorePropertyManager corePropManager =
    profilePropManager.GetCoreProperties();
ProfileTypePropertyManager profileTypePropManager =
    profilePropManager.GetProfileTypeProperties(ProfileType.User);
ProfileSubtypePropertyManager profileSubtypePropManager =
    profilePropManager.GetProfileSubtypeProperties("Employee");

// Now, to create the property
// First the core property
CoreProperty coreProp = corePropManager.Create(false);
coreProp.Name = "MyProp";
coreProp.Type = "Boolean";
coreProp.DisplayName = "My Property";
coreProp.Commit();
```

```
// then the profile type property
ProfileTypeProperty profileTypeProp =
    profileTypePropManager.Create(coreProp);
profileTypeProp.IsVisibleOnEditor = true;
profileTypeProp.Commit();

// and finally, the profile subtype property
ProfileSubtypeProperty profileSubtypeProp =
    profileSubtypePropManager.Create(profileTypeProp);
profileSubtypeProp.IsUserEditable = true;
profileSubtypeProp.Commit();
```

And no, you can stick to the old ways.

Happily, there aren't too many such deprecated object model features. Microsoft has started a special MSDN Code section to track deprecated code, which sadly isn't very up-to-date at this point:

http://code.msdn.microsoft.com/sps2010deprecated

New Feature Elements in SharePoint 2010

In the current beta, there are only a few new feature elements, but these may save you a lot of time from traditional SharePoint 2007 development techniques.

WebTemplate

A really major and exiting new thing in SharePoint 2010 is that site definitions are now available as features, which is a huge leap forward. You create site definitions using the WebTemplate element and treatthe definition more or less like a regular SharePoint 2007–style site definition.

The even better news, at least initially, is that when you export a site as a template, you no longer get an STP template, but rather a WSP solution, containing all the features required to re-create your site. One of those features will contain a WebTemplate element.

The slightly bad news is that WSP site templates still suffer from many of the same problems that STP templates do. For instance, you always get far too much information exported from the WSP to import the whole thing into any meaningful Visual Studio solution.

Another problem is that you still don't get any .NET code exported, so you still need to re-create any workflows, event handlers, and such manually or at least ensure that you export the assemblies at the same time as you extract your site template.

PropertyBag

A very common need for developers of slightly more complex solutions is to store custom properties in SharePoint. Traditionally, this had to be done using .NET code, but now there is a dedicated feature element you can use.

The PropertyBag element can add properties to various types of content, specifically, webs, folder, files, and list items. Properties are added using a key and value, as you would do with .NET code–based property bag manipulation.

WorkflowActions

If you work with workflows in SharePoint Designer, you may know that you can extend the list of available workflow actions. However, doing so in SharePoint 2007 meant a lot of details and horrible deployment methods.

Well, now you can deploy your custom workflow actions as features, meaning the development, deployment, and management will be a lot easier. You have tons of options with the WorkflowActions element, many of which are directly mapped to the previous model.

Other Feature Elements

Even with only a few major new elements available in SharePoint 2010, there are still other items of interest. Although space does not permit a deep dive into any of these, I want to mention some features I find interesting.

> **Note**
>
> I highly recommend looking to the SDK for more details, including the now downloadable version available at http://www.microsoft.com/downloads/details.aspx?FamilyID=f0c9daf3-4c54-45ed-9bde-7b4d83a8f26f.

Columns and Content Types

One major change to content type elements in SharePoint 2010 is that you can now have web-scoped content types deployed as features. That means you can also have web-scoped columns.

Another important new feature of columns is also that you can now enforce unique values in the columns and, of course, that you can now enforce data integrity by setting relationships between lists.

Lists

Lists get a huge overhaul in SharePoint 2010, and as a developer, you need to be aware of the implications.

Right at the top of great new features is that lists now use, or at least should use, XSLT for view rendering rather than the dreaded CAML View schema. XSLT rendering means you get much better development support, including using external tools and visual authoring support.

You can also have basic list validation, based on much the same syntax as you have in calculated columns. Oh, and as I briefly mentioned earlier, lists now support referential integrity, so you can have cascading deletes or prevent deletion of referenced list items. That's one step toward a decent data model in SharePoint.

Let's hope that SharePoint 2013 brings the remaining steps in this marathon.

Chapter

3

Welcome to Visual Studio 2010

Introducing the SharePoint developer's tool for the future

It is time to start exploring the new SharePoint developer's favorite tool, Visual Studio 2010.

In this chapter, and in this issue, I will help you explore the new features of Visual Studio 2010 because you will likely use it for SharePoint development. I'll try to stay away from the more general Visual Studio changes, simply to make room for more SharePoint-related stuff.

I'll show you some of the new feature types and walk you through the new tools.

> **Note**
>
> USPJA Labs allows you to play around with SharePoint 2010 without having to install anything, on machines that have adequate performance.
>
> You still need to install Visual Studio 2010, though, but this is available as a free download from Microsoft.
>
> http://uspjalabs.com/

Prerequisites

I'm going to assume that you have some experience working with Visual Studio in previous versions.

Before you dive in, however, you should prepare your lab environment. There are some new things to consider even if you have a running SharePoint 2010 lab available.

First, expect to increase your RAM requirements. A normal SharePoint 2010 lab machine may work well with as little as 3GB of RAM, but trying to run an SP2010 lab with Visual Studio means an absolute lower limit of 4GB. Even that will feel like moving through mud, so if you can, get more RAM.

Second, if you have any plans to do Office client integration testing, prepare to add RAM to the previous numbers. A rule of thumb is to add 500MB of RAM per Office application you want to run in a development situation. Thus, if you plan to develop Word, Outlook, and Excel add-ons, you should expect your minimum memory requirements to increase by 1500MB.

Finally, remove any service applications you do not need at the time. Unless you are actively working with search, for example, remove that service application for the time being to improve performance. You can always install it later; the time it takes to reinstall is usually less than the added time you'll wait for a slowly building solution.

How much RAM do you need? The answer is usually "more." My personal labs run on 8GB of RAM. Because I always use virtual machines for my labs, that means I need more RAM than that for my physical machine, not to mention the needs if I want to have another machine running Active Directory or Exchange.

My laptop has 16GB of RAM and feels fairly good as long as I don't overload it with multiple Visual Studio sessions or install services beyond those I absolutely need.

> **Note**
>
> Did I mention that USPJA Labs has adequate performance? All lab machines have at least 8GB of RAM.
>
> USPJA Labs is available to the public, not just students at the academy.
>
> http://uspjalabs.com/

To follow along with the exercises in this chapter, you should have your SharePoint 2010 environment up and running and have at least one site collection with which you can play around.

WSPBuilder and Visual Studio 2010

The bulk of this chapter focuses on the Visual Studio SharePoint tools. These tools, an evolution of Microsoft's much-hated Visual Studio extensions for WSS, have gained popularity among SharePoint 2010 developers.

However, I still find myself using the good ol' WSPBuilder method of development. That probably has more to do with how I like working and how I have worked for a long time.

What you should know is that you can keep using the WSPBuilder method if you prefer. In fact, I've already worked on several production solutions for SharePoint 2010 using WSPBuilder.

Carsten Keutmann is busy creating the next version of WSPBuilder, and if history is any judge, his efforts will result in an absolutely brilliant developer experience.

You need to download the correct version of WSPBuilder, though, because the normal version will not support SharePoint 2010. To get the correct version, go to the WSPBuilder CodePlex site and then to the Downloads page where you should find the correct version for SharePoint 2010.

http://wspbuilder.codeplex.com/

Installing Visual Studio 2010

Installing Visual Studio 2010 is easy. In fact, the default spouse mode almost works flawlessly.

> **Note**
>
> Spouse mode, in case you don't know, is clicking Yes, Yes, Yes, Yes, Yes, and OK until you're done.

Visual Studio 2010 First Look

When you start Visual Studio 2010 for the first time, you are greeted by a fresh, new interface, shown in Figure 15.

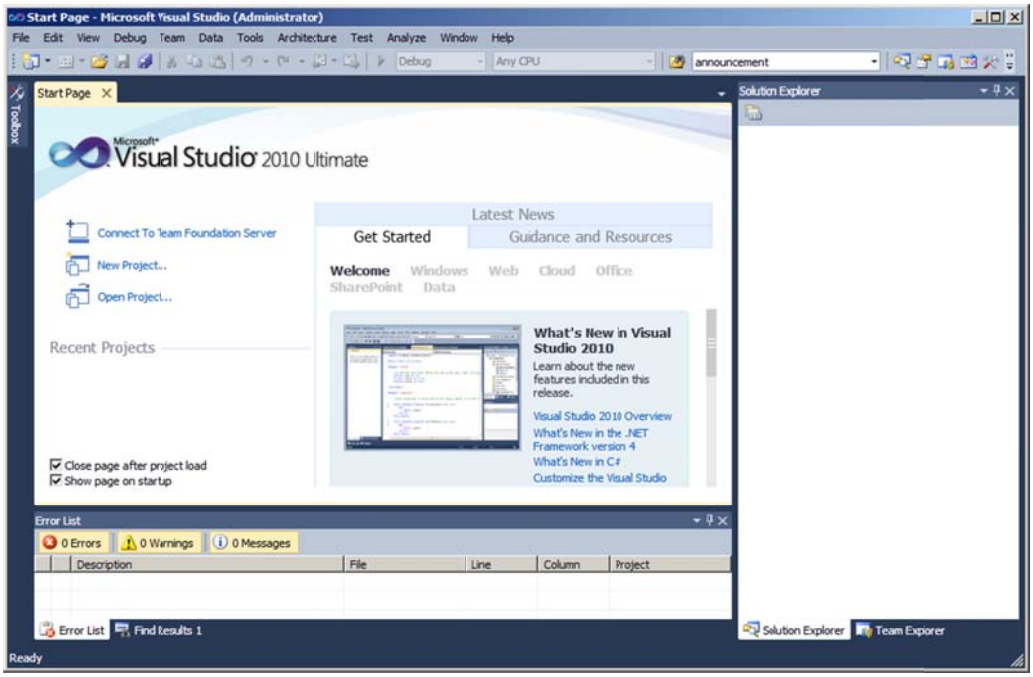

FIGURE 15. VISUAL STUDIO 2010 START PAGE

You'll quickly find, though, that most of the familiar elements from previous versions are right where you expect them to be. Most of the standard behavior is also the same. In fact, Microsoft has apparently gone to great lengths to decrease the amount of time needed to learn the new version.

That's the good news. The somewhat bad news is that the SharePoint tools experience is quite different from earlier versions of any Microsoft tools. I'm saying "somewhat bad news" because once you get over the initial learning hurdle, the new tools are also far, far better than anything that has come out of Redmond for a long time.

First create a new project by hitting the New Project link on the front page. Visual Studio will present a variety of project types, including a SharePoint node. Selecting that node will display the available SharePoint project types, as shown in Figure 16.

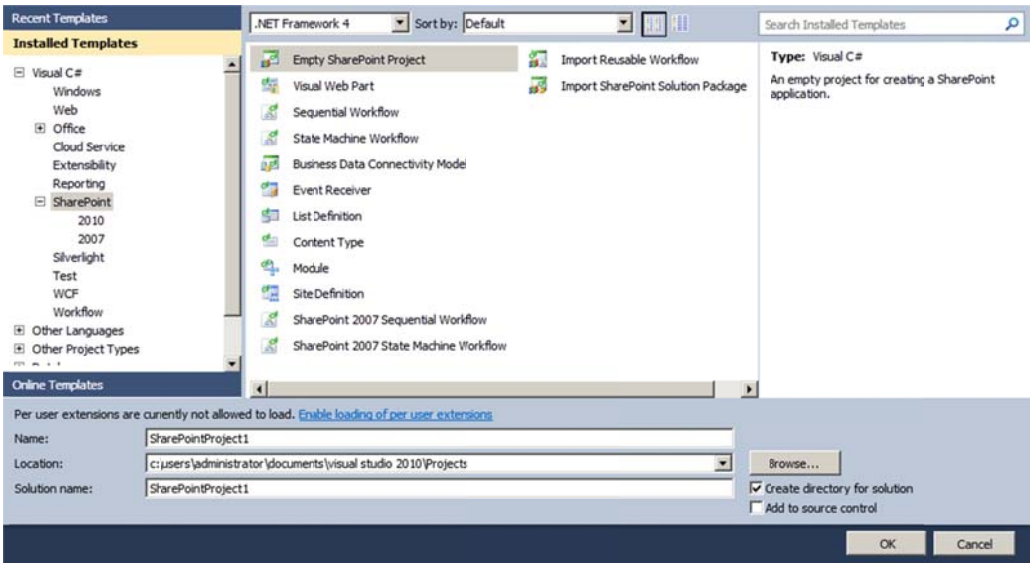

FIGURE 16. SHAREPOINT PROJECT TYPES

You can further narrow the choices available to SharePoint 2010 or 2007 projects, but the SharePoint 2007 projects are limited to two workflow project types. You should notice four especially cool new project types for SharePoint 2010, which I'll get back to in a moment.

Regardless of which SharePoint project type you select, Visual Studio will prompt you for debugging settings. These settings define what will happen once you hit F5 to do your debugging.

Oh, I didn't mention? Visual Studio SharePoint tools now support full F5 debugging. For all project types. Yeah, I know, the mind boggles.

In any case, Figure 17 shows how the debugging screen looks. Note that you select between a sandboxed solution and a farm solution.

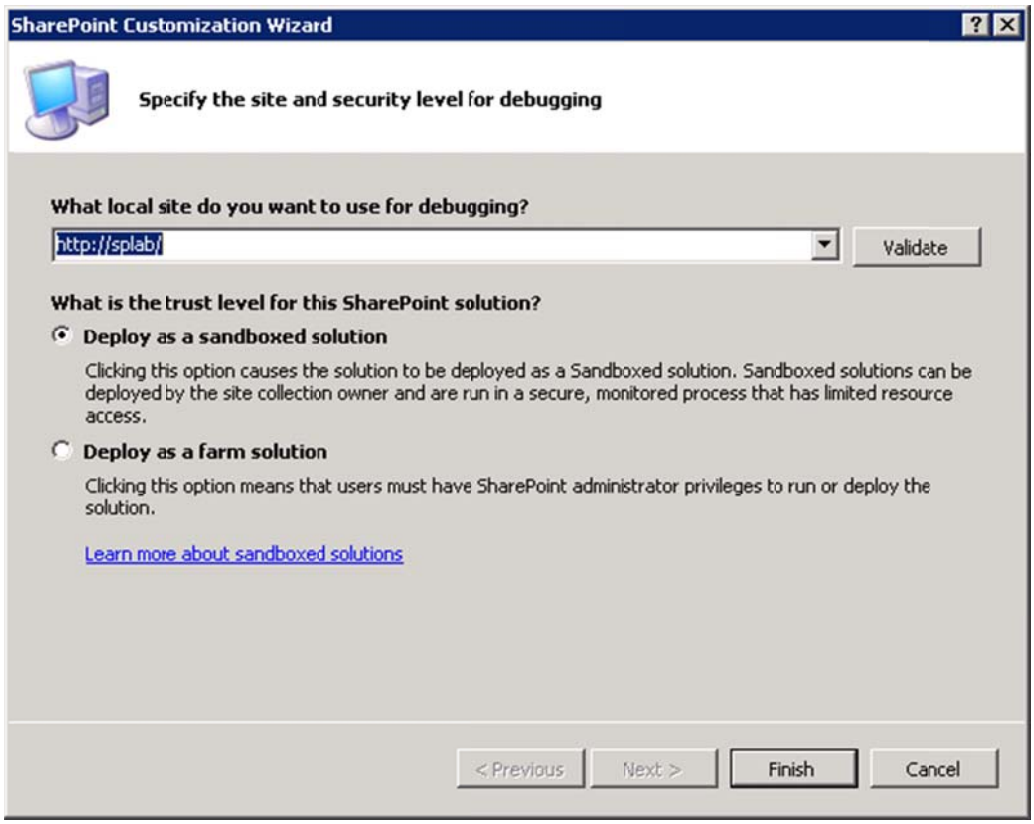

FIGURE 17. DEBUGGING OPTIONS

> **Note**
>
> If you want to swap between a sandbox and farm solution, you can do so by selecting the project in Solution Explorer and then changing the properties for the project. You need to modify the Sandboxed Solution property.

Let's take a look at a few of these project types.

Visual Web Part

The visual web part is a very cool new project type that gives you an easy-to-develop user control web part. In previous versions of SharePoint, using a user control web part required manually loading either the user control or other tools. Now, you can get the fancy visual design experience that ASP.NET page developers have enjoyed for many years, as shown in Figure 18.

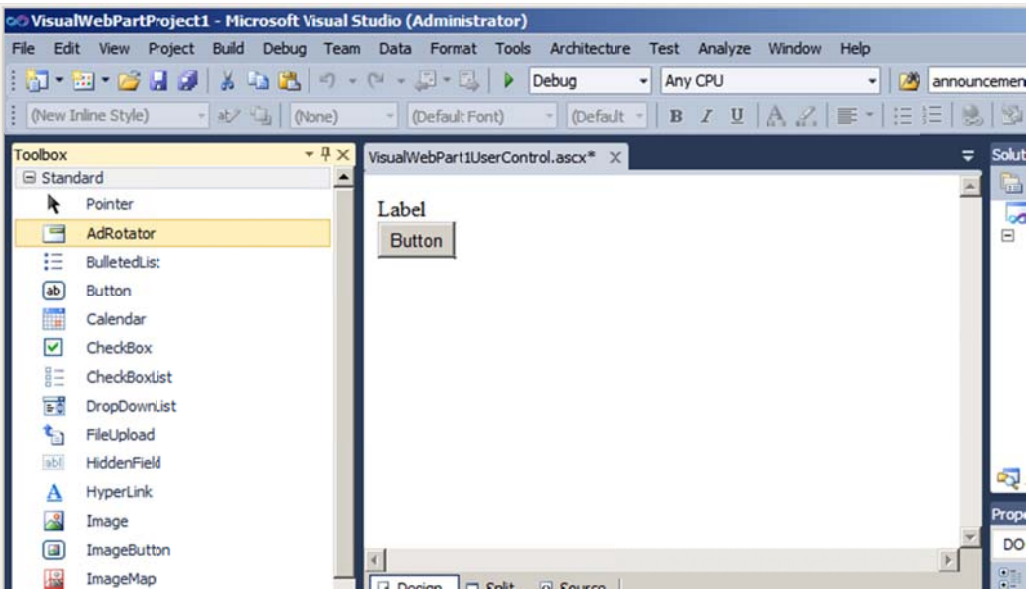

FIGURE 18. VISUAL WEB PART

Visual web parts cannot be deployed as sandboxed solutions, though, since they require access to the user control stored on the file system. Sandboxed solutions do not have access to the file system.

> **Note**
>
> Microsoft has released a Visual Studio add-on package called Visual Studio 2010 SharePoint Power Tools that allows you to mimic the visual web part in a sandbox solution. This works by deconstructing the user control when you build the solution, thus circumventing the need to access the file system.
>
> Please note that this method has several caveats, but unfortunately a full discussion on this is beyond the scope of this report.
>
> http://visualstudiogallery.msdn.microsoft.com/en-us/8e602a8c-6714-4549-9e95-f3700344b0d9

Now, before you rush out to buy your Visual Studio 2010 license to get your WYSIWYG web part experience, you should know that this really isn't anything new. What goes on behind the scenes, as you can see in the class file in a default Visual Studio web part project, is that the user control is loaded, just like you could do manually in previous versions:

```
Control control = this.Page.LoadControl(_ascxPath);
Controls.Add(control);
```

```
base.CreateChildControls();
```

Put that code in your SharePoint 2007 web part projects, and you get exactly the same thing. So, although visual web parts are touted as the coolest thing since the first *Die Hard* movie, they really aren't anything magical—or even impressive.

Business Connectivity Model

Business Connectivity Services (BCS) is the evolution of BDC. Oh, but it's a good evolution, I'll tell you that much before you cringe at the thought of touching anything related to BDC.

In Visual Studio, the Business Connectivity Model project allows you to create a BCS project to hook external data into SharePoint. The good news this time is that BCS allows full CRUD (create, read, update, and delete) operations on data, whereas BDC was limited to reading data.

Even better, you get a mostly visual development experience, and you rarely need to write code beyond the queries required by the various CRUD methods.

I won't walk you through BCS in this issue because of the amount of space such a walk-through would require. However, MSDN has an excellent walk-through that I used initially to get a grasp of the new technology:

http://msdn.microsoft.com/en-us/library/ee231515(VS.100).aspx

To follow along with that walk-through, you will need an external database, such as the AdventureWorks sample database, which is available as a free download from CodePlex:

http://msftdbprodsamples.codeplex.com/

> **Note**
>
> Installing the AdventureWorks database can be…tedious. Make sure you follow the installation instructions exactly, and be prepared to do some SQL Server configuration if you haven't installed the full-text search engine and the analysis engine.
>
> Here are the installation instructions:
>
> http://msftdbprodsamples.codeplex.com/wikipage?title=Database%20Installer%20Help&referringTitle=Home

Import Reusable Workflow

Workflows in SharePoint 2010 get a major overhaul, and they now offer far better authoring experiences for developers, power users, and even business users.

One of the major advances is the introduction of reusable SharePoint Designer workflows. In previous versions, these workflows were locked to a single list forever once you deployed them. Now, however, you can create reusable workflows that you can use on any number of sites.

You can also import SharePoint Designer 2010 workflows into Visual Studio. This is, however, a one-way import, so you cannot export the workflows to SharePoint Designer again.

> **Note**
>
> Sadly, you still cannot import SharePoint Designer 2007 workflows.

Import SharePoint Solution Package

Initially, this feature alone seems to be a gift from the gods to SharePoint solution development; you can now import your existing WSP solutions directly into a Visual Studio project.

Importing a WSP solution is a simple process, requiring you to select which parts of the WSP you want to import. This is probably thought of as an easy way to import site templates, which are now WSP solutions instead of in the arcane STP format.

They have definite problems, though. First, you cannot import any .NET source code, so if your solution uses .NET code, you will need to add that code manually. Any slightly complex SharePoint solution contains at least some code, so you're really limited to the most basic of solution packages.

Second, importing a site template WSP solution either gives you far too much information, including every field, content type, form, and page in the entire site, or gives you too little information to reuse the template at a later time.

Knowing that WSP files are really just CAB files, you'll probably save time by just renaming the .wsp file to .cab, exporting the contents, and then adding that content to the Visual Studio project. You still won't get the .NET source code.

But What's the Point?

Of the four really new project types in Visual Studio 2010, only two are really useful: the BCS model project type and the reusable workflow import project type.

That's not necessarily bad. To be honest, most seasoned SharePoint developers end up using the blank SharePoint project type in any case.

The specialized types, with the exception of the BCS model project type, are most useful for deploying only a single type of feature. However, most SharePoint projects utilize a wide variety of feature types. As such, starting with a specialized project type rarely makes sense.

Let's move on an investigate the new way of developing SharePoint solutions in Visual Studio.

Your First Visual Studio SharePoint Project

Let's see whether we can make heads and tails of this new development environment.

Start by creating a new blank SharePoint project in Visual Studio 2010. Name it whatever you like; I've just used the default SharePointProject1 name suggested by Visual Studio. Next, select to create a farm solution rather than a sandboxed solution.

> **Note**
>
> You can change the type of solution later, but be aware that some features are not available as sandboxed features.

Once you have selected the site you want to use for debugging and hit OK, your project should open. Allow your eyes to slide over to the Solution Explorer, shown in Figure 19.

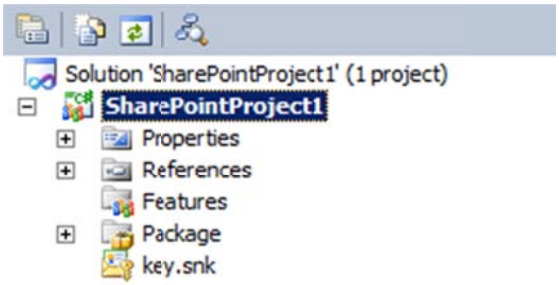

FIGURE 19. BASIC SOLUTION EXPLORER

Note that you have a Features node and a Package node. We'll add features in a moment, but first let's take a look at some of the tools we have available.

Visual Studio 2010 take a page from WSPBuilder's book and adds several very useful context menu options. When you right-click the project (SharePointProject1 in my example), you'll get many of the same options that WSPBuilder users have enjoyed for several years, as shown in Figure 20.

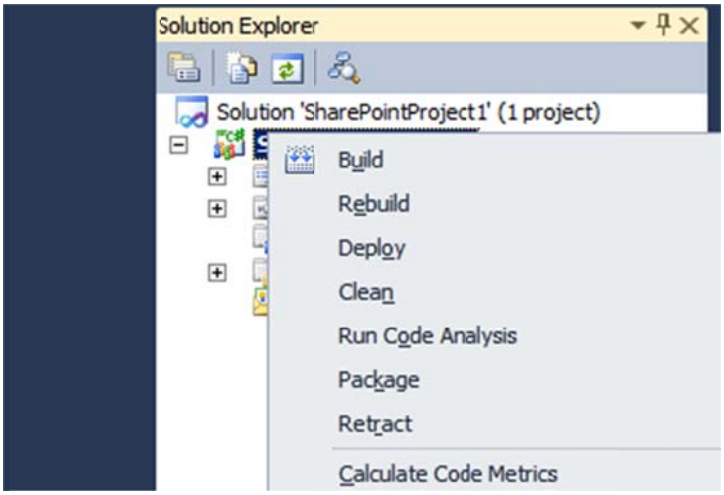

FIGURE 20. PROJECT CONTEXT MENU

The SharePoint-related options here are Deploy, Package, and Retract. These options allow you to deploy your package, to create the WSP solution file, or to retract an existing solution.

You won't necessarily use these options a lot, though. As I mentioned previously, Visual Studio now supports F5 debugging, meaning you can simply hit F5 to have Visual Studio deploy your solution to the chosen site, attach the debugger, and launch your site to begin testing.

> **Note**
>
> The Package option is useful when you want to create and move your solution to another server.

Deployment Configurations

Speaking of deployment, Visual Studio now gives you detailed control over what happens when you deploy your solution through what are known as *deployment configurations*.

You set up the deployment configurations in the project properties. Right-click your project node in Solution Explorer, click Properties, and then select the SharePoint node. At the bottom of the property page, shown in Figure 21, you'll find the out-of-the-box deployment configurations.

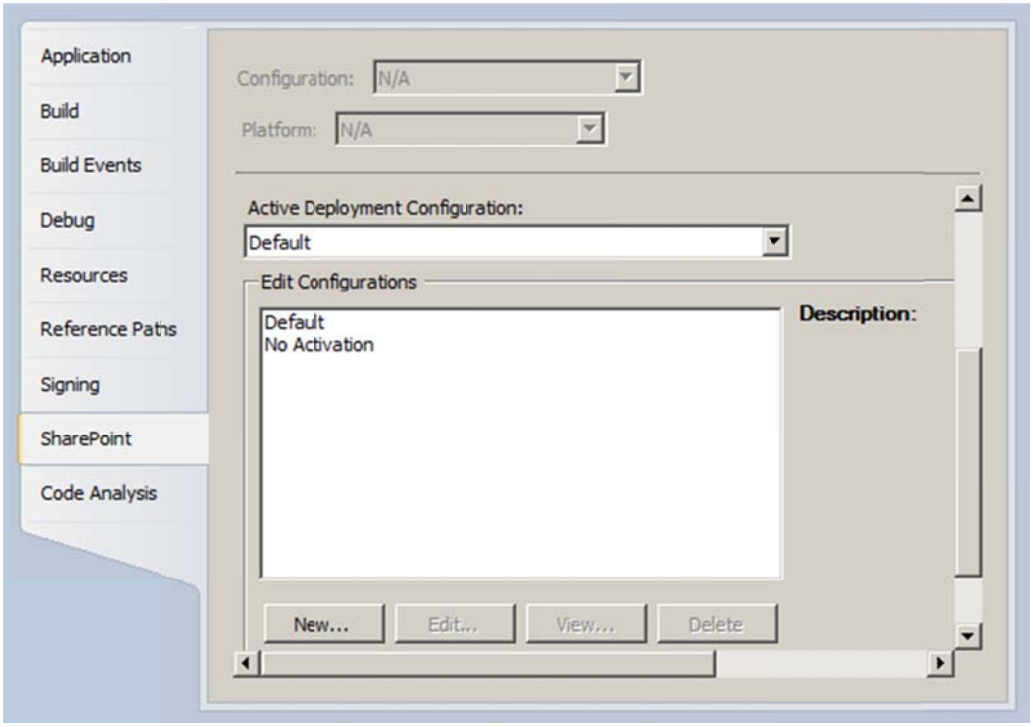

FIGURE 21. OUT-OF-THE-BOX DEPLOYMENT CONFIGURATIONS

You cannot edit any of the out-of-the-box configurations, but you can view them. Click the Default configuration and then the View button to see what happens when you deploy your solution. Figure 22 shows the default Default configuration.

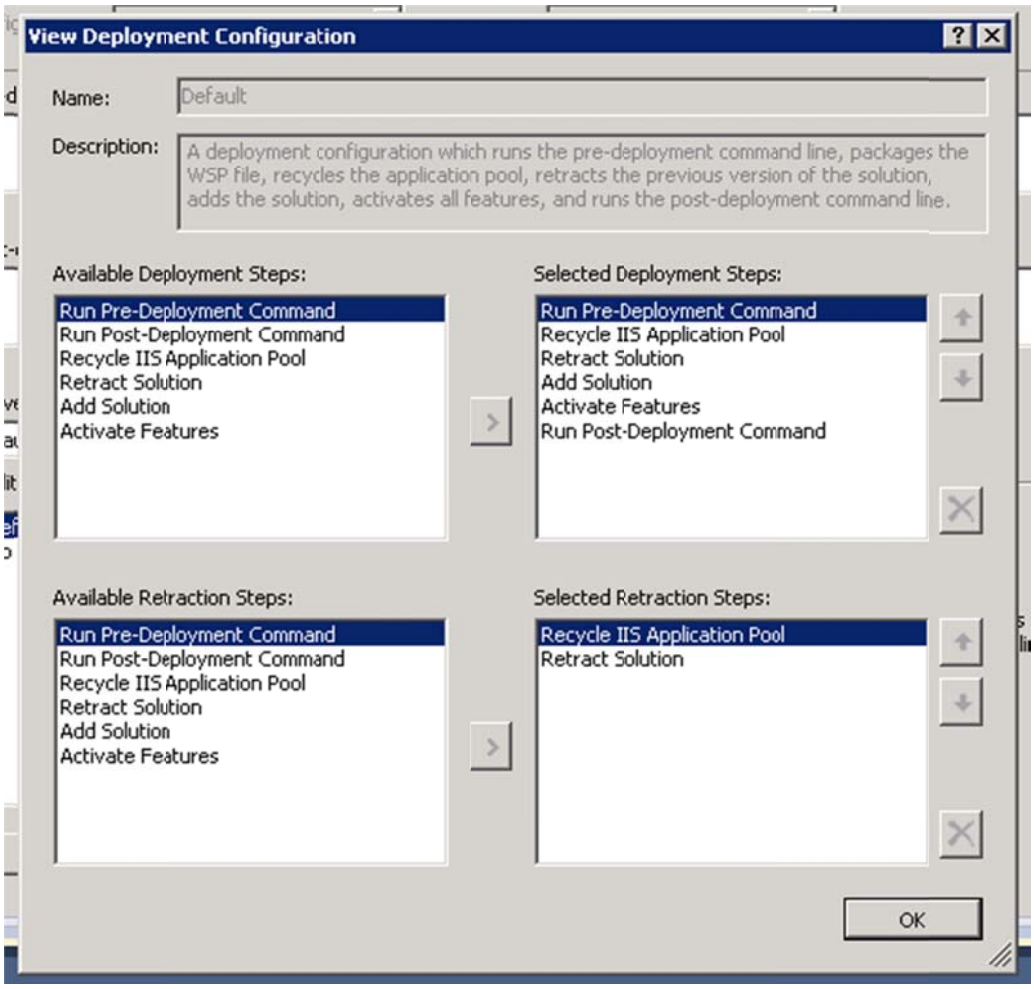

FIGURE 22. DEFAULT DEPLOYMENT CONFIGURATION

If the out-of-the-box configurations don't meet your needs, you can create new deployment configurations. Since you cannot edit the default configuration, it makes sense to create your own custom configuration so that if you need to change any of the deployment steps, you can do so in your own configuration.

To do so, click the New button to launch the Add New Deployment Configuration dialog box, shown in Figure 23.

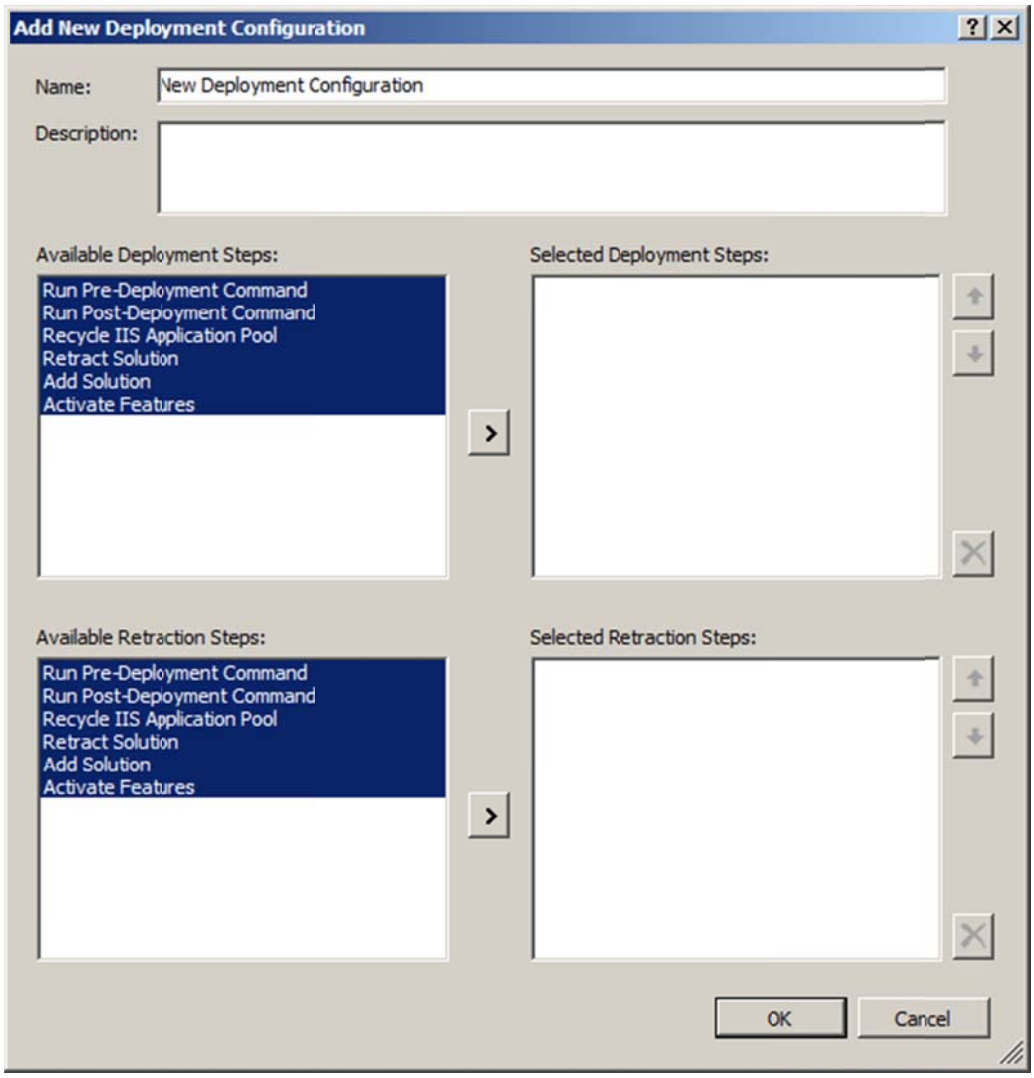

FIGURE 23. ADD NEW DEPLOYMENT CONFIGURATION

To replicate the default configuration, select all the available deployment steps and all the available retraction steps, and hit the > buttons between them to add them all to the respective process.

You can also change the order of steps, but keep in mind that some steps require previous steps to finish first. For example, you cannot activate the features before you add the solution.

> **Note**
>
> You can also create custom steps for both deployment and retraction.

Once you hit OK, your new configuration is available for use. If you want to use that configuration, select the configuration from the Active Deployment Configuration drop-down menu.

For now, close the property page.

SharePoint Server Explorer

Next, I'll show you a potentially very useful tool, the Server Explorer with its new SharePoint Connections option. You can find the Server Explorer by selecting it from the View menu or by pressing Ctrl+W and then L.

When the Server Explorer starts, it shows up as a tool pane, and you'll see the SharePoint Connections node. Right-click that node, and select Add Connection. Fill in the URL of your SharePoint site, and hit OK to connect.

Doing so will enable you to use the Server Explorer to browse your SharePoint site, as shown in Figure 24.

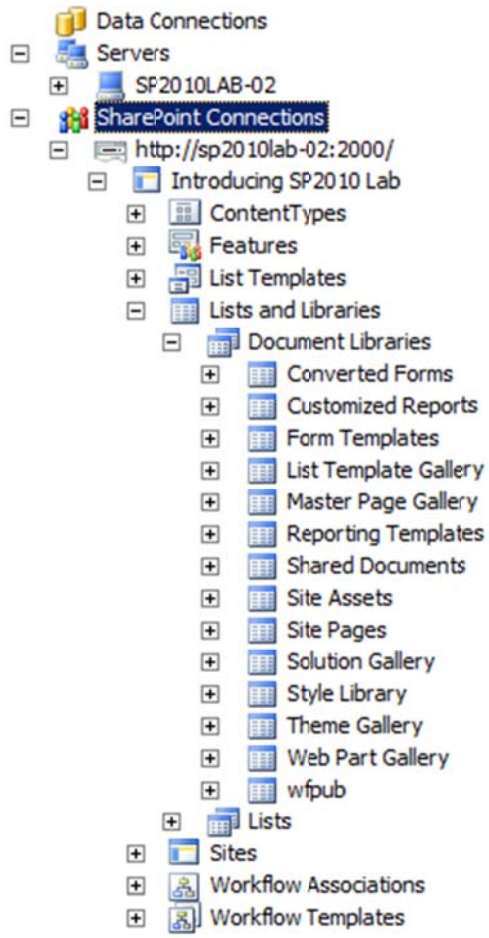

FIGURE 24. SHAREPOINT SERVER EXPLORER

If you have used SharePoint Manager 2007 (or the new SPM2010), this may seem familiar. However, don't be deceived or get your hopes up. Beyond viewing the structure and content of your site, you can hardly do anything here.

Check out the context menu for the Shared Documents library in Figure 25.

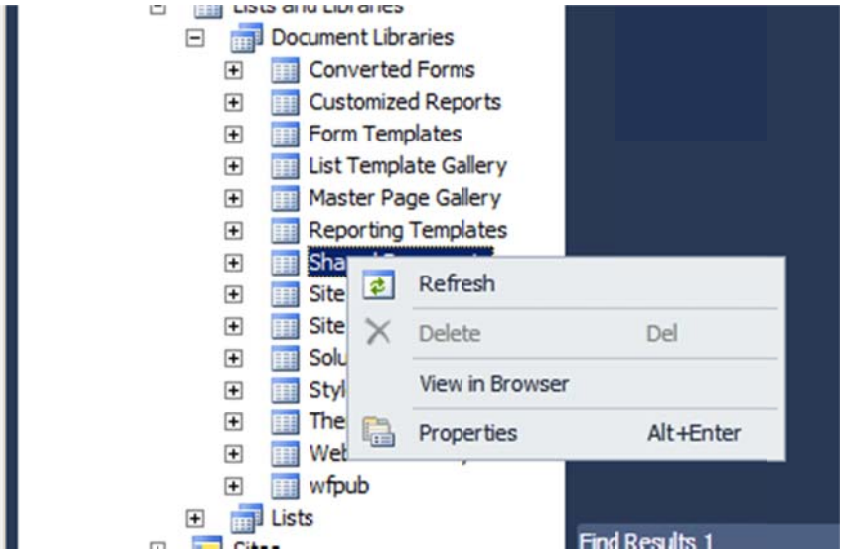

FIGURE 25. SERVER EXPLORER CONTEXT MENU

The story is more or less the same for all types of objects.

Selecting the Properties option fills the Properties pane of Visual Studio with all the properties for the object, but you cannot change any of the properties, so again, it is read-only.

So, should you simply forget about the Server Explorer?

No.

In fact, the Server Explorer is a great feature, even despite the default lack of functionality. You see, you can customize the Server Explorer and extend it using third-party add-ons.

One of those add-ons is made by SharePoint MVP Waldek Mastykarz. His extension gives you an easy way to generate SPMetal definitions that you need in order to use LINQ to SharePoint.

> **Note**
>
> You can download Waldek's add-on from the Visual Studio gallery at http://visualstudiogallery.msdn.microsoft.com/en-us/c523e7ba-ba9d-45c4-98ea-b02b19f81640.

After installing the extension, Figure 26 shows what happens to the context menu.

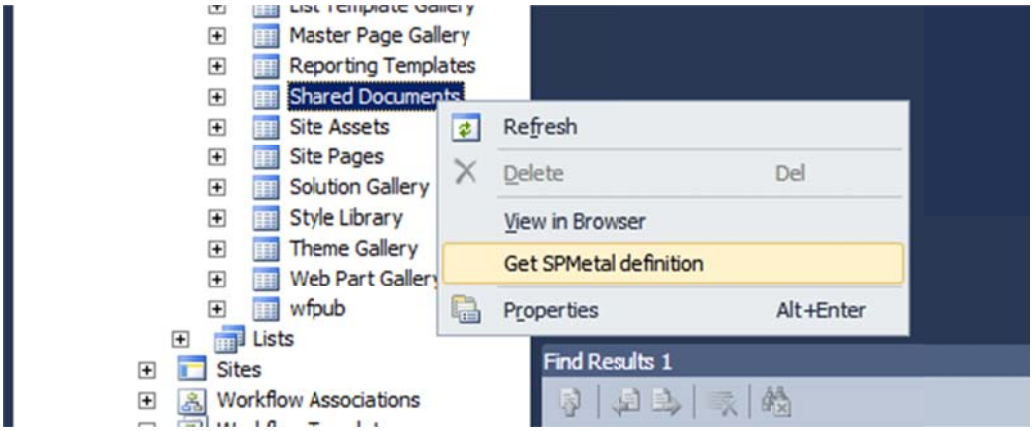

FIGURE 26. SPMETAL EXTENSION

If you want to use LINQ to SharePoint, you'll definitely want to grab a copy of Waldek's add-on.

> **Note**
>
> By default, Visual Studio prevents you from loading user extensions such as the SPMetal add-on. To enable loading, select the Tools→Extensions menu, and click the "Enable loading of per user extensions" link.
>
> You may also need to enable loading of user extensions explicitly for the administrator user, and the link in the Extensions dialog box will take you to the correct spot to do so. Otherwise, select Tools→Options and then the Extension Manager node where you will find a checkbox to enable loading of extensions for the administrator.

I'll show you more about LINQ to SharePoint in Chapter 7 of this issue.

Now, though, we should start creating some content to see how Visual Studio 2010 really works.

Adding Features to Visual Studio 2010

Let's see one of those fancy new visual web parts in action. Right-click your project node in Solution Explorer, and select Add→New Item. Note that the Add menu contains three SharePoint-specific entries for adding mapped folders, as shown in Figure 27.

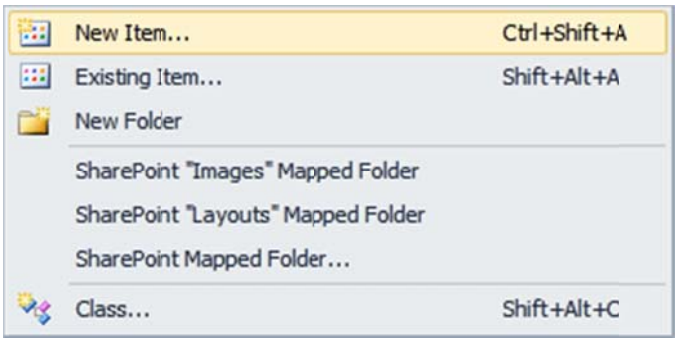

FIGURE 27. ADD MENU

These Images and Layouts mapped folders relate to the SharePointRoot\Template folder and their respective subfolders. These two named folders are often used for storing resources global to the farm, but you can create a mapped folder to any folder in the SharePointRoot hierarchy.

> **Note**
>
> SharePointRoot is the new name for the folder formerly known as *12-hive* or *14-hive*.

In the Add New Item dialog box, select the Visual Web Part item type from the SharePoint/2010 node. Name it anything you like; the default is fine for our exercise. Click Add to add the visual web part to your project.

When Visual Studio has added your new item, it opens the ASCX user control file for you. Right at the first line is one of those great time-savers in Visual Studio 2010:

```
<%@ Assembly Name="$SharePoint.Project.AssemblyFullName$" %>
```

That's right, you can now use tokens inside your SharePoint markup. This particular token finds the full strong name of your assembly and adds it to the .ascx file.

Tip

> **Tip**
>
> Microsoft has a full list of available tokens on MSDN:
>
> http://msdn.microsoft.com/en-us/library/ee231545(VS.100).aspx

The next thing you may notice is that you now get a whole range of files and items added to your Solution Explorer, as shown in Figure 28.

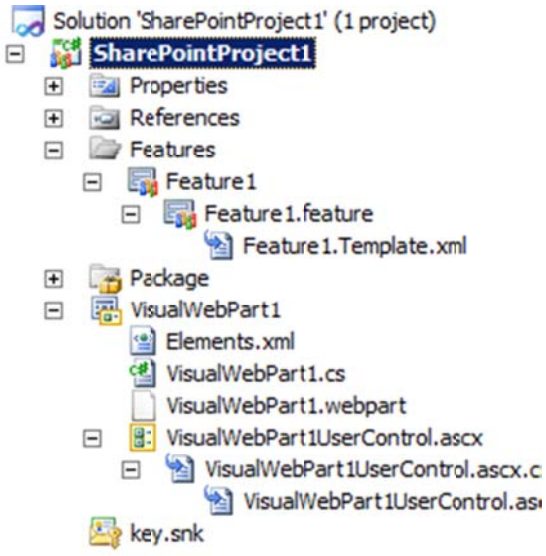

FIGURE 28. VISUAL WEB PART IN SOLUTION EXPLORER

Pay particular attention to the Features node and its child nodes. A huge improvement from previous versions of Visual Studio is that you now have visual editors for your features.

Feature Editor

Double-click either the Feature1 or Feature1.feature node, and take a look at the feature editor, as shown in Figure 29.

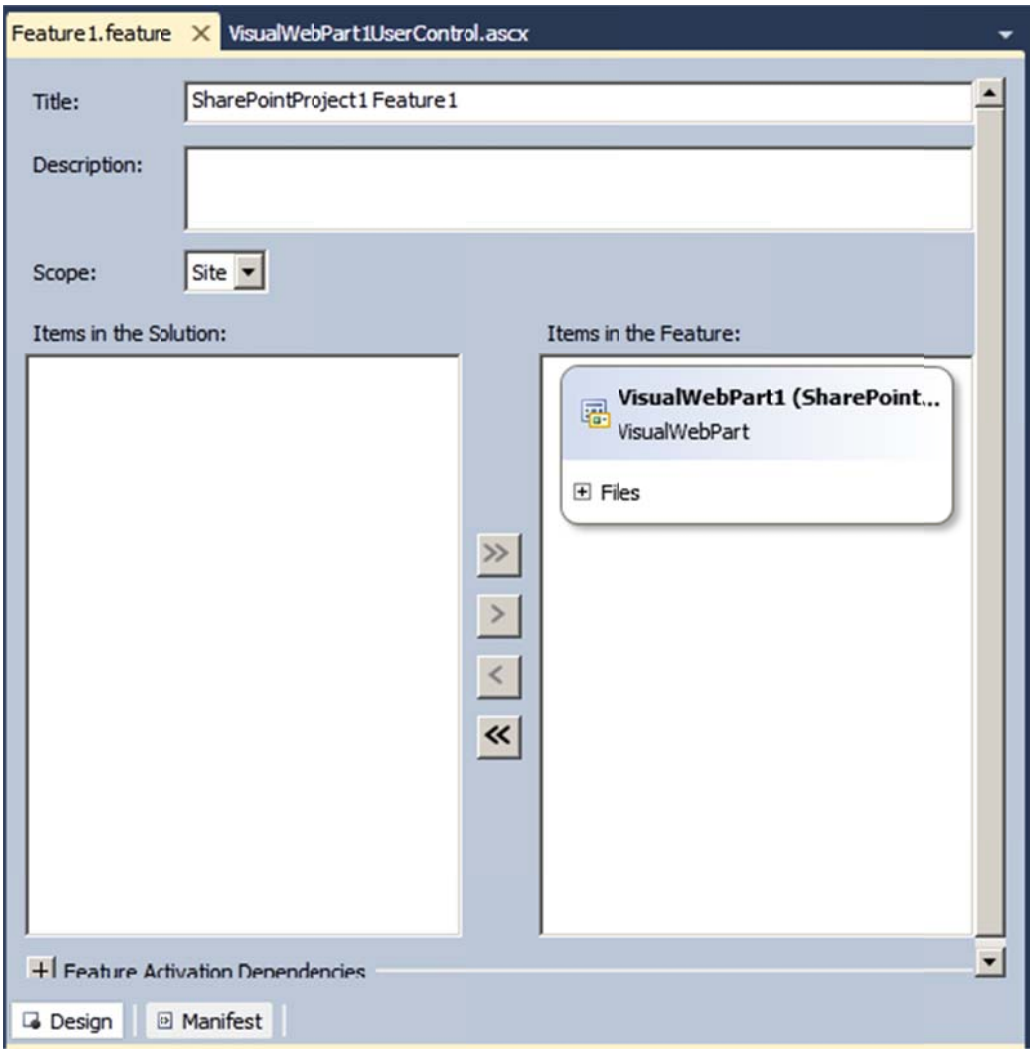

FIGURE 29. FEATURE EDITOR

The feature editor allows you to move items or elements in and out of features. You can also change the scope of a feature, but be aware that you still need to heed the item feature scope requirements. For example, if you change your feature scope to anything other than Site, you cannot add the visual web part back into the feature.

Near the bottom you will find buttons for two modes of operation, Design and Manifest. Clicking the Manifest button gives you the feature's XML code. You can edit this if you want in order to give you full control over what is added to the feature's XML code.

You can even control the entire feature.xml file if you prefer absolute control.

Package Editor

Speaking of editors, you have a similar editor for the solution package. Double-click the Package.package item in the Solution Explorer to open the package editor, shown in Figure 30.

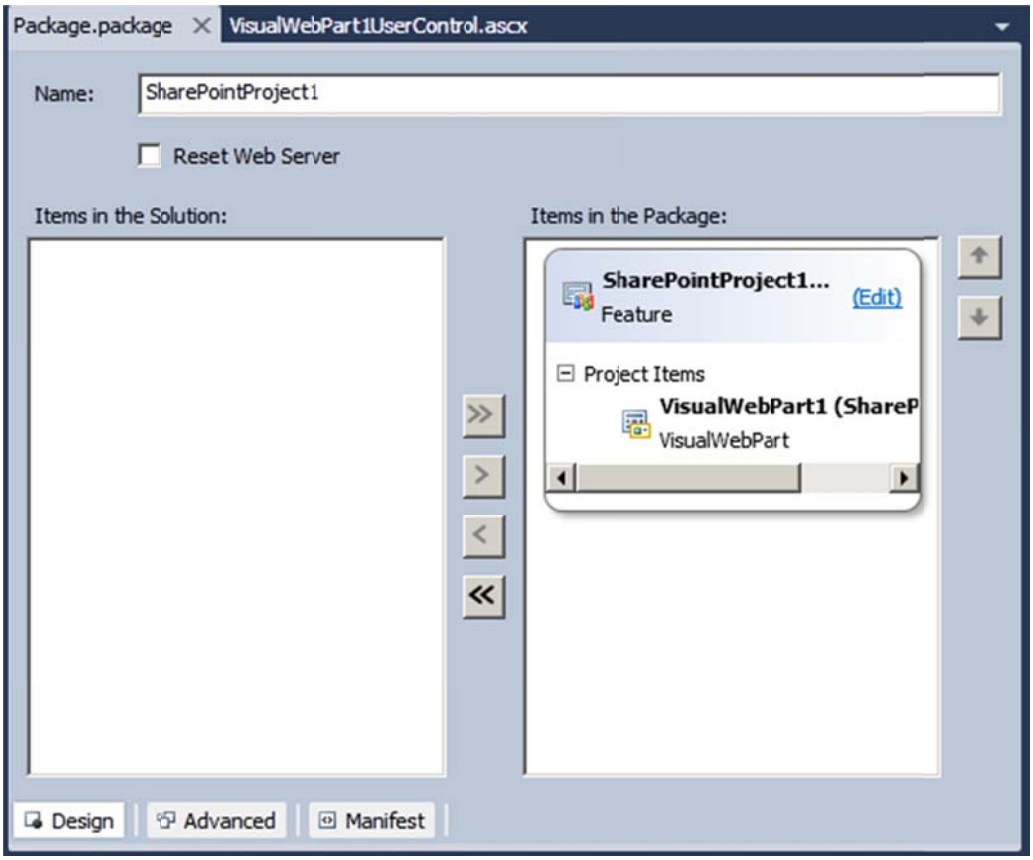

FIGURE 30. PACKAGE EDITOR

The package editor allows you to move features and items in and out of packages. Like the feature editor, only legal items are displayed. For example, if your project is a sandboxed solution, you cannot add visual web parts.

Also similar to the feature editor, you can modify the solution manifest directly by clicking the Manifest button at the bottom.

The Advanced button allows you to add assemblies for deployment. This is especially useful if you have multiple projects in your Visual Studio solution, because it allows you to deploy the output from all projects as part of a single SharePoint package.

While you are in either the package editor or the feature editor, notice that you get a Packaging Explorer tool pane. This tool pane gives you a nice overview of your package and allows you to add new items as well as rearrange features and items. Figure 31 shows the Packaging Explorer.

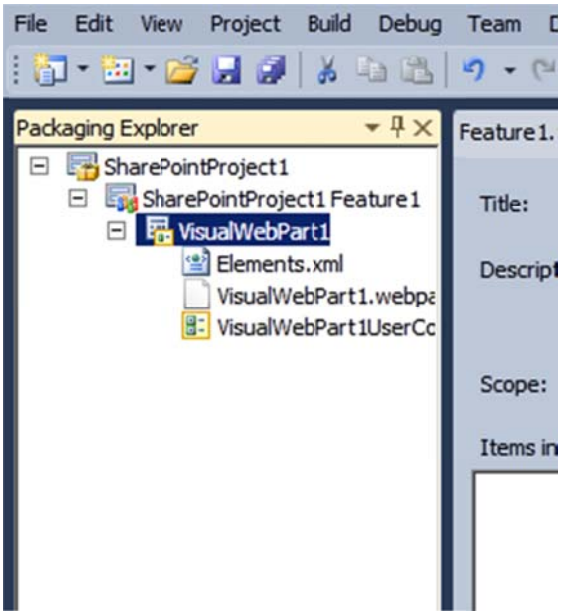

FIGURE 31. PACKAGING EXPLORER

I think it is time to see how all of this works in a simple solution walk-through.

Visual Web Part Walk-Through

We're really interested in putting some content in our site, aren't we? Let's pull our focus back to the visual web part.

If you haven't already, create a new Visual Web Part project in Visual Studio 2010. Open the .ascx file, and change the mode to Design (available below the source code).

Next, make sure you have the Toolbox tool pane open, and drag a button and a label from the Toolbox to the design surface. The result should resemble Figure 32.

FIGURE 32. AS SIMPLE AS IT GETS

Yeah, this is the most used example in the world, but we're going to update the label once we hit the button.

To do so, double-click the Button button to enter the code behind and generate the method outline for the Click method.

Update the Button1_Click method as such:

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "Hello World!";
}
```

Still extremely fascinating, I know.

Now—and this is the really cool part—hit F5. Assuming you haven't cocked up anything, here's what's going to happen...

Visual Studio will follow the deployment configuration you have selected to deploy the solution package. You can follow along either in the status bar or in the output pane, available from the View menu.

Next, once the solution deployment completes, Visual Studio will fire up your default browser and open the site you designated as your debugging site. It will also hook the debugger to that browser session.

Once your site opens, however, you're on your own for testing. What you want to do to add your web part to a page is to click the Edit page button, shown in Figure 33, and then go to the Insert tab in the Editing Tools tab of the Ribbon.

FIGURE 33. EDIT PAGE BUTTON

Once you have the Insert tab open, click the Web Part button, and browse to the Custom folder. You should see your VisualWebPart1 there, so select it and hit Add to add it to the web part zone of your choice, as shown in Figure 34.

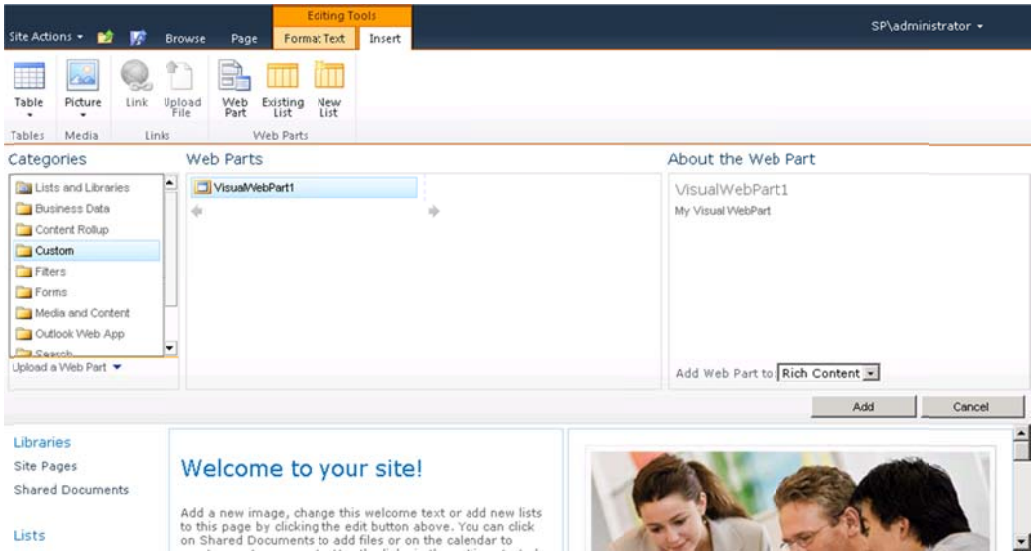

FIGURE 34. ADDING THE WEB PART

Once you hit Add, SharePoint will add your fancy new web part. Feel free to hit the Button button to verify that the label changes, as shown in Figure 35.

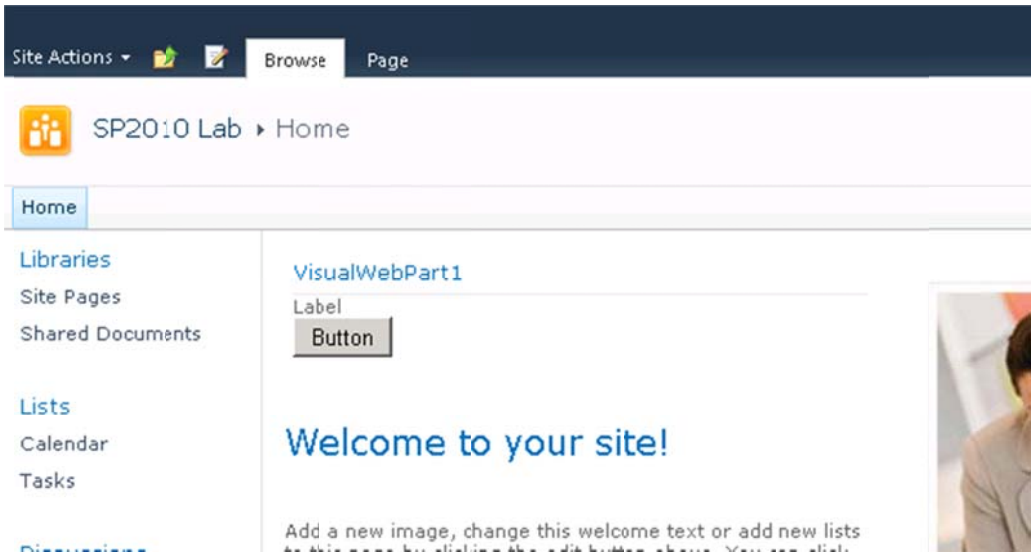

FIGURE 35. WOO-HOO! AREN'T WE GOOD?

OK, the web part itself may not be the cure to all ailments, but the development experience and the F5 debugging functionality are really nice.

There's more, though. Without closing your browser, return to Visual Studio, and add a breakpoint on the Label1.Text line. Then, return to the web page, and try hitting the button again. As if by magic, Visual Studio pops up and lets you step through the code, as shown in Figure 36.

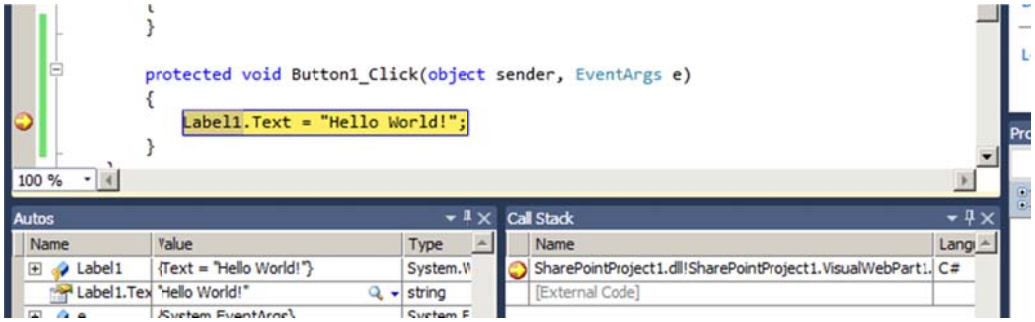

FIGURE 36. DEBUGGING

I think that about does it for the visual web part. You'll get the same development experience with the F5 debugging for any other features as well.

> **Note**
>
> Many developers find the F5 debugging experience to take too long to start. You can still manually attach to the processes using the Visual Studio Debug menu though, or, as I do, just use WSPBuilder.

Before we end this chapter, I'll show you one more ace up Visual Studio 2010's sleeve.

Workflows

Although I won't walk you through an entire workflow development scenario, I want to show you a few improvements to the workflow infrastructure and authoring in SharePoint 2010 and Visual Studio 2010.

Add a new item to your project of type Sequential Workflow. Note that once you do, you get a SharePoint Customization Wizard to guide you through setting up your new workflow.

The first page of the Customization Wizard allows you to select either a list workflow or a site workflow. This is one of those new features in SharePoint 2010; you can now have workflows that are independent of any single list or library. I mentioned this briefly in Chapter 4 when we looked at SharePoint Designer 2010, and you'll find the same feature in Visual Studio 2010.

For now, since we won't be developing this workflow fully, just select a site workflow, and hit Next.

On the next page of the Customization Wizard, you hook your workflow up to a workflow history list and a task list. Leave the default selections, and hit Next.

If you followed my advice and chose a site workflow, the workflow start options on the final page of the Customization Wizard are fairly simple; you can only select that the workflow starts manually when a user selects to do so.

Once Visual Studio finishes adding the new workflow item, the workflow designer opens. If you have worked with Visual Studio workflows before, the experience is very similar.

However, one big improvement is how you work with forms. In previous versions, adding initiation and association forms was a nightmare. Not so anymore. Simply right-click your workflow in the Solution Explorer, and select Add→New Item. Next, select "Initiation form" from the available item types.

This will cause Visual Studio to add a new ASPX form for you and even update the workflow XML with the correct InstantiationUrl value. Simply add what you need to the form; for simplicity's sake, you can also just edit the existing code. Next, hit F5, and twiddle your thumbs while Visual Studio deploys your workflow.

Once that process completes, your new workflow is available on the Site Actions menu. Start a new instance of your site workflow, and your new workflow initiation form shows up, as shown in Figure 37.

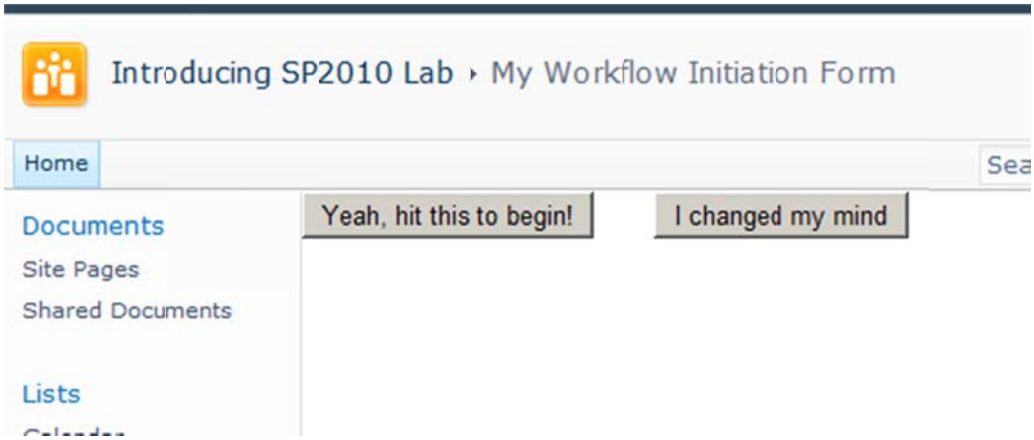

FIGURE 37. CUSTOMIZED WORKFLOW INITIATION FORM

You can do the same for association forms, by the way.

Final Thoughts and Additional Resources

Never stop learning.

SharePoint 2010 gives developers a range of new toys with which to play. You have seen many of these toys described in this issue, but I can assure you there is still plenty to learn.

Of course, the best news of all is that most of what you've previously learned for SharePoint 2007 still works and is core functionality even in the new version.

As such, if you are looking to dive deeper into SharePoint developer, I recommend you pick up "Beginning SharePoint Development" and get the basics of development and then expand your knowledge by rereading this report.

http://beginningsharepointdevelopment.com/

You'll be building your first SharePoint 2010 solutions in no time!

To get information about the upcoming issues of *USP Journal*, make sure you sign up for the mailing list on the *USP Journal* website, http://www.understandingsharepoint.com/journal. Members of the list receive special offers, discounts, and previews of new issues.

Finally, check out my blog at http://www.understandingsharepoint.com/url/1000 where you will find shorter articles, tips, questions and answers, and downloadable content.

Thank you for reading this issue, and I hope to see you next time.

.b

Previous *USP Journal* Issues

If you are considering buying multiple issues, you can save big by purchasing a subscription bundle from http://www.understandingsharepoint.com/journal/subscriptions

Volume 1 (SharePoint 2007)

Issue 1: SPCurrentUsers Explained

This issue deals with the solution SPCurrentUsers available from CodePlex.
URL: http://www.understandingsharepoint.com/journal/volume-1/issue-1

Buy now!

Issue 2: Developing SharePoint Content Types

If you want to learn everything about developing SharePoint content types, you should get your hands on issue 2.
URL: http://www.understandingsharepoint.com/journal/volume-1/issue-2

Buy now!

Issue 3: SPTags Explained

If you have ever tried to build custom field types in SharePoint, you definitely want to pick up issue 3.
URL: http://www.understandingsharepoint.com/journal/volume-1/issue-3

Buy now!

Issue 4: SharePoint Designer Workflow

In issue 4 of *Understanding SharePoint Journal*, you will learn how to create workflows in SharePoint Designer 2007.
URL: http://www.understandingsharepoint.com/journal/volume-1/issue-4

Buy now!

Issue 5: Beginning SharePoint Development

Issue 5 introduces you to development for SharePoint. You will learn the basics of various SharePoint development techniques.
URL: http://www.understandingsharepoint.com/journal/volume-1/issue-5

Buy now!

Issue 6: SPThemes and SPSampleData Explained

Understanding SharePoint Journal presents two new solutions, SPThemes and SPSampleData, designed to teach you new aspects of SharePoint development.
URL: http://www.understandingsharepoint.com/journal/volume-1/issue-6

Buy now!

Issue 7: Introducing SharePoint Visual Studio Workflows

In Introducing SharePoint Visual Studio Workflows, I will introduce you to authoring Visual Studio workflows, and show you various aspects of developing workflows for SharePoint in Visual Studio 2008. URL:
http://www.understandingsharepoint.com/journal/volume-1/issue-7

Buy now!

Issue 8: Professional SharePoint Development

This issue teaches you a powerful method for agile SharePoint development, including the 42 lines of code that will save your SharePoint solutions forever URL:
http://www.understandingsharepoint.com/journal/volume-1/issue-8

Buy now!

Issue 9: Advanced Content Type Development

In this issue of USP Journal, I'm going to show you the power of content types in ways you may never have seen before. I'll show you how to extend SharePoint itself by creating custom functionality that you control through the development of content types. URL: http://www.understandingsharepoint.com/journal/volume-1/issue-9

Buy now!

Volume 2 (SharePoint 2010)

Introducing SharePoint 2010

The first SharePoint 2010 book to hit the market, introducing you to SharePoint 2010 as a developer, administrator, or end user.
URL: http://www.introducingsharepoint2010.com/

Buy now!

SharePoint Designer 2010 Workflows

In volume 2, issue 2 of Understanding SharePoint Journal, you will learn how to create workflows in SharePoint Designer 2010.
URL: http://www.introducingsharepoint2010.com/

Buy now!