European SharePoint Community

PROFESSIONAL SHAREPOINT 2013 DEVELOPMENT

CHAPTER 3 - DEVELOPER TOOLS FOR SHAREPOINT 2013

by Paul Swider, Reza Alirezaei, Brendon Schwartz, Matt Ranlett, Scot Hillier, Jeff Fried & Brian Wilson

IN THIS CHAPTER YOU WILL:

- Understand the different tools available to SharePoint developers
- Explore what's new in SharePoint Designer 2013
- Learn how to use the new SharePoint tools in Visual Studio

3

Developer Tools for SharePoint 2013

WHAT'S IN THIS CHAPTER?

- Understanding the different tools available to SharePoint developers
- Exploring what's new in SharePoint Designer 2013
- Using the new SharePoint tools in Visual Studio

SharePoint has become one of the most developed-on platforms over the last decade, and Microsoft has invested in the developer experience with every release of SharePoint. SharePoint 2013 continues to improve the tools available for developers such as Visual Studio and SharePoint Designer while making tremendous strides with the addition of apps, Office 365, and Microsoft Office development. The addition of apps in SharePoint 2013 is a drastic change along with using familiar programming web standards such as HTML, CSS, JavaScript, OData, REST, and OAuth. The developer tools have included this support as well with full support for development against the cloud platforms. If you have used SharePoint 2010, you will see that many of the same project files are available, but there are new additions to the array of items you can now use.

The development tools are more integrated with the platform; debugging is easier in complex scenarios such as the web and more; and new tooling containing designers and templates was added so that you can easily work on SharePoint and transition to another web-based framework. This chapter takes a deeper look at each of these tools so that you can understand what they can do for development with SharePoint.

CUSTOMIZATION OPTIONS WITH SHAREPOINT

SharePoint is full of options for customization and development. These changes can be made from many different tools by many different users. Although every developer doesn't need to know each tool in depth, it helps to know what each tool's capabilities are and what the strengths are of each tool. Table 3-1 shows an overview of the different types of users and the tools they use to customize their SharePoint experience.

TABLE 3-1: Tools for Customization

USER	PRIMARY TOOL
End user	SharePoint Sites
Power user/Designers	Microsoft SharePoint Designer 2013
Developer	Visual Studio 2012

The largest group of users is end users. They understand how to use the user interface to build an application for a specific need. End users primarily make out of the box (OOB) changes directly in the SharePoint site that the user has access to. Although many developers don't initially think of these changes as development, SharePoint has grown over the years to include end users that fully understand how to make small modifications to HTML, CSS, JavaScript, and customizable web parts that allow them to build dynamic applications.

Check out the user community at `https://www.nothingbutsharepoint.com/sites/eusp/`.

Many of the changes focus on changing the look and feel of the site, input entry, and information management with deep knowledge of the problem domain the user works on. For many end users SharePoint development is not their primary job, but a requirement to have functional and optimal applications for the solutions they solve.

The next group of users is the power users. They need more capabilities than the user interface provides them, and the tool of choice is Microsoft SharePoint Designer. As the name implies, the tool has its roots as an HTML editor for designing web pages. The tool is a must-have for many power users, especially with improved capabilities added in SharePoint Designer 2010 that included additional workflow capabilities, packaging integration, BCS integration, and richer HTML capabilities. The SharePoint Designer team understands the importance of both the end user and developers, especially for Application Lifecycle Management (ALM). Developers using SharePoint Designer can deploy solutions directly to their production sites if they build only a single site, or they can save the changes locally. The changes stored locally can be reopened in Visual Studio for further customization and stored in source control when they are completed.

The final group of users is developers, and the tool for coded or packaged solutions is Visual Studio 2012. Visual Studio 2012 gives a familiar interface for developers of .NET languages such as ASP.NET that requires less learning time to quickly start building apps. One of the major challenges facing new developers has always been learning the SharePoint API Framework and understanding how

to modify the XML that defines SharePoint. The enhancements in Visual Studio 2012 have created an improved experience and added new tools to give developers a user interface for many common changes in the XML.

OOB DEVELOPER EXPERIENCE

The user interface for SharePoint has become one of the most used development platforms. This is due to the number of users of SharePoint and the different ways users can change the application directly through the user interface. Many end users can create powerful applications and experiences with a few clicks. In addition to building the applications directly on the site, you can use the UI to quickly create mock-up lists and sites that can be reused in SharePoint designer and Visual Studio. This adds a great capability for developers looking for a rapid application development (RAD) platform for proof-of-concept code or expediting development. Each option has been improved with SharePoint 2013 for creating full-featured business applications.

The Get Started with Your Site web part provides users with a quick set of actions to start customizing their sites. They can change the look and feel, title, and logos, and share the site with others. Not every site must have the getting started web part, but you can allow site owners to add and remove it easily through the user interface. When the users no longer need the web part, they can click the Remove This link and start using the Settings menu for any future changes.

One of the quick actions is to add the new Project Summary web part. This new Project Summary web part can be quickly added to the page using the Working on a Deadline? link on the Get Started with Your Site web part. Click the link and you are prompted to add two apps: the Tasks (with a Timeline) and Calendar (as shown in Figure 3-1).

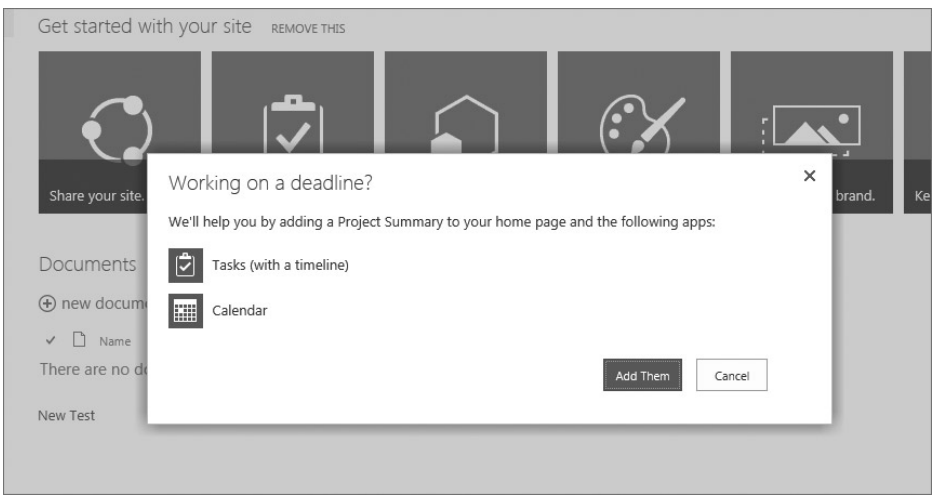

FIGURE 3-1

You can add other applications to the site to further customize the look or provide added functionality, such as a Site Mailbox. From the quick actions, click Add Lists, Libraries and Other apps to bring up the Add apps menu. The new SharePoint store enables you to select from a number of prebuilt applications without coding them. When you select the SharePoint store, you can access full prebuilt apps or apps that enhance your site.

Understanding the User Interface for Customization

Customization of sites is one of the main actions that a developer needs to perform, whether that modifies the look and feel or just uses web parts and apps on the page. There have been major changes to the user interface look and feel, but the page customization experience remains familiar. This means that the major changes to the page with editing using the SharePoint Ribbon have not changed. Don't forget to click the Page tab if the Ribbon does not automatically appear. The Ribbon is still there; it just might not be needed at the moment. There are still contextual tabs based on the actions you perform and the web part that you select. There is a subtle difference with how the Ribbon displays to the user. Instead of requiring page real estate to always be reserved, the Ribbon now drops down onto the page and then can be hidden to display the content under the Ribbon as shown in Figure 3-2 (hidden Ribbon) and Figure 3-3 (displayed Ribbon).

FIGURE 3-2

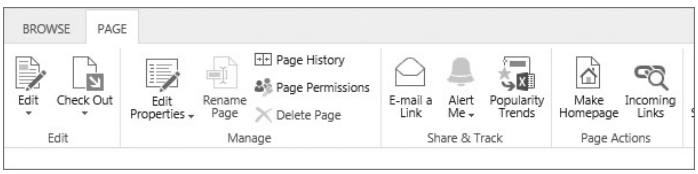

FIGURE 3-3

Branding the site continues to evolve because this is one of the biggest modifications performed. SharePoint 2013 introduces a new branding experience that builds on previous versions of SharePoint called *composed looks*. SharePoint 2010 has themes that are a grouping of the files needed to change the colors on a site. This makes it easy to quickly apply the colors you want to the CSS on the SharePoint pages without changing the CSS location or making any changes to the page. This did not allow designers to completely change the look and feel with a single package, as composed looks provide. Each composed look is made up of a Display Name, Master Page, Theme, Background Image, and Font Scheme. Also, a new Try It Out link has been added to make it easier to view the site before applying any changes. As you can see from Figure 3-4, you can change the entire look and feel of the site with a single click.

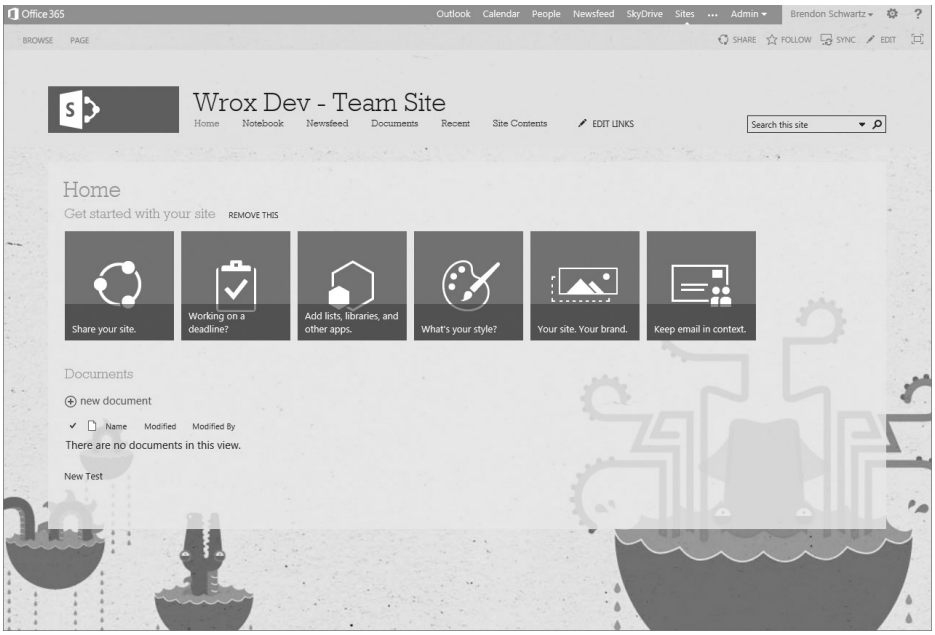

FIGURE 3-4

SharePoint 2010 made great changes to the way users edited pages using wiki page editing and adding the ability to place web parts anywhere on the page. The ability to create and design your layouts remains part of the rich user experience. Major changes have been added in SharePoint 2013 to the Web Content Management (WCM) and Publishing pages that can be reviewed in Chapter 10, "Web Content Management."

Microsoft has created a more intuitive user interface by adding common language terminology and adding inline editing for some common navigation. The use of the Pencil icon, along with the text "Edit", can be seen in any location that allows users to edit. For navigation you can now click the Edit Links icon to manage the top navigation links or the quick launch links.

Anyone new to SharePoint finds that the page editing still provides the familiar Microsoft Word experience using the wiki pages that enable you to add all types of rich content, from text to images. The experience is more fluid than previous versions with new features such as Paste Clean and a preview of applying styles before you select them. Users were always asking what the styles looked like to know which one to apply to the content. The Change with Preview feature enables users to know what the pages, styles, and fonts are before they make the content changes. Also the Styles menu is now similar to the Microsoft Word Style menu layout with the name and a preview of what the style looks like. Users now see the full set of styles on the Ribbon without needing to try each one or have a style guide printed out and next to each computer.

One feature that users have wanted is to add code to the page without the page manipulating it when it is saved. Two new options enable this change: the Embed Code option and the Video and Audio embedding option. With the exponential growth of many cloud services, users need a way to add

embedded video from such sites as YouTube and Vimeo (in addition to adding custom scripts to sites such as Facebook, LinkedIn, and Twitter). These cloud services enable users to build customized sites by using generated code that users can add by copying and pasting.

Getting your web parts onto the page and customizing them has not changed from SharePoint 2010. The Web Part menu appears on the Ribbon for inserting your web parts onto the page. You can select your web parts and insert them into the page as in the previous version. In SharePoint 2013 there are also app parts that can be added to your pages to enrich the content. These app parts are added to the page in the same way that traditional web parts are added to the page through the Parts viewer. To make it easier to find all of the parts, the layout of the Web Part menu has categories: the web parts available in that category, a description of the web part, and the location of where to add to the web part, as shown in Figure 3-5.

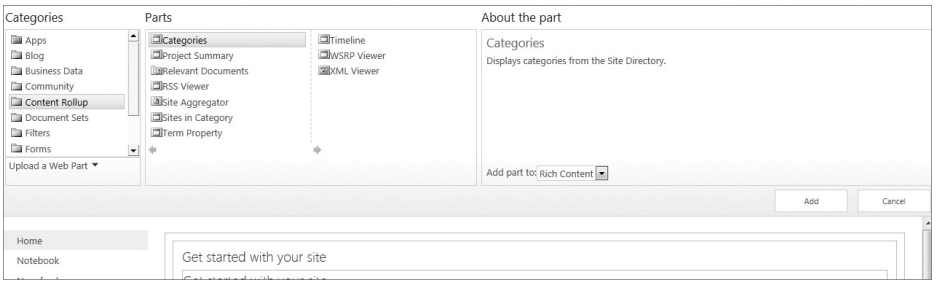

FIGURE 3-5

The Modify Web Part settings have not changed from previous versions of SharePoint. They continue to appear on the right-side tool pane, enabling you to customize the properties for the web part, change its appearance, or modify its layout.

One of the nice things about SharePoint development is that you get a number of out of the box (OOB) web parts to speed your development so that you do not need to write everything from scratch. Although the following isn't an exhaustive list of all the web parts in SharePoint, it includes a number of key, new web parts that you should be aware of: Timeline, Project Summary, and Community Changes.

Microsoft focused on the needs of users to keep all their tasks organized. One area that was needed was to keep tasks visible and updated. The quickest way to add the Project Summary web part to the page is to use the quick actions from the Get Started web part as previously mentioned. After the web part has been added to the page, you can allow end users to update the timeline or manage the task list providing the data to the web part. You can see that the Project Summary web part provides an overview of the upcoming tasks and a timeline of the current tasks. Now you can manage all your tasks, create a project site, and allow users to have a full view of the timeline, all within the user interface, as shown in Figure 3-6.

The Timeline web part provides the timeline without the Summary or Quick Edit links. This allows the user that is managing the site to use the web part that fits their needs, whether that is managing the tasks or simply viewing them. The Timeline web part provides the ability to move the layout of the tasks. Users can still quickly interact with the tasks on the timeline by clicking the task, which brings up a dialog with common editing options such as opening the full item or removing the task from the timeline.

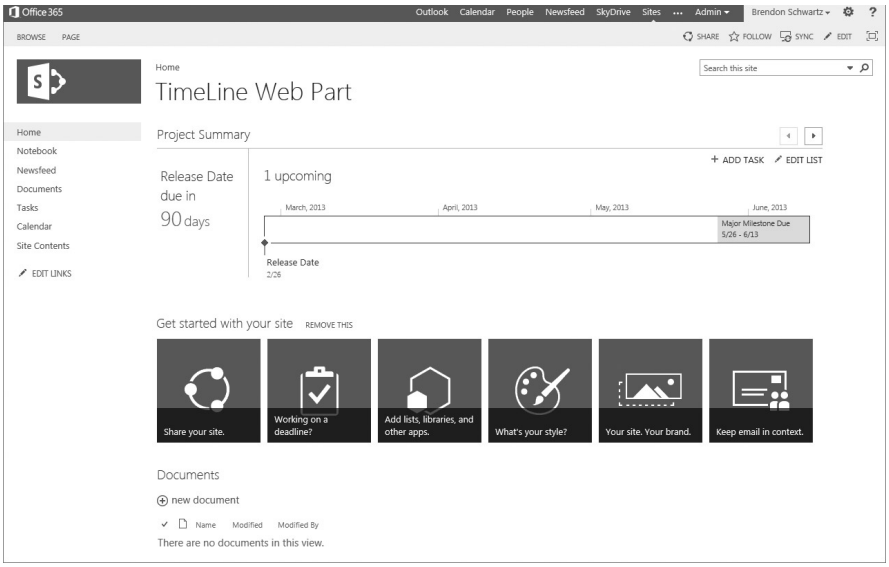

FIGURE 3-6

Microsoft has created a new set of web parts based on the new Community features. The new web parts are About This Community, Join, My Membership, Tools, and What's Happening. These web parts can be added to a site that has the Community features enabled, shown in Figure 3-7. The community site already has the web parts on the page, but you can also add them or move them on the communities you create. You can see they are all grouped into the Community category to easily find and add them.

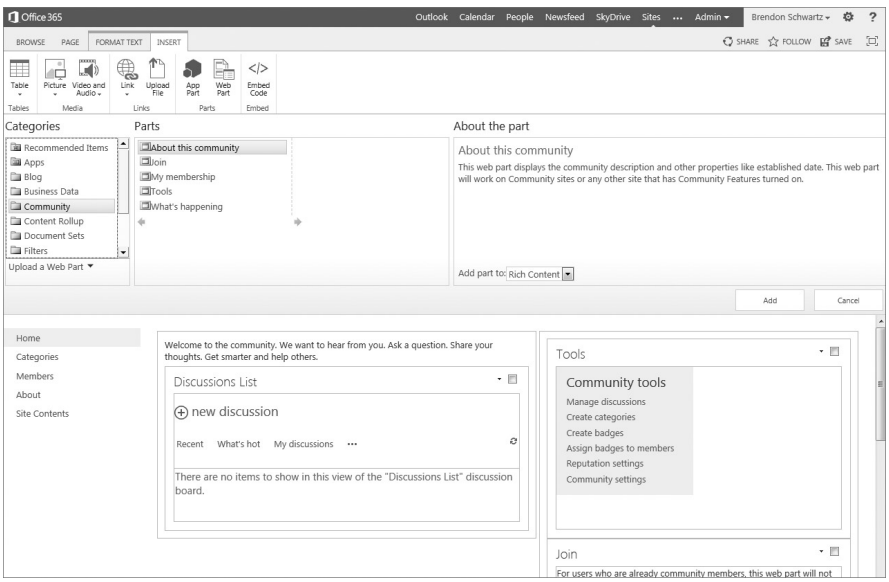

FIGURE 3-7

UNDERSTANDING SHAREPOINT DESIGNER 2013

SharePoint Designer (SPD) can be a powerful tool for developers and power users offering the ability to upgrade solutions to Visual Studio and the ability to export work directly from your site. SharePoint Designer 2013 has become the tool of choice for power users, especially with the additional workflow and BCS additions it received in SharePoint 2010. The design community now has other options for HTML editors, but none of them are as integrated to SharePoint as SharePoint Design. Even with the addition of SharePoint Design Manager in the SharePoint user interface, SharePoint Designer has improvements and will continue to be a useful tool for power users. One of the key goals for page layout and design by the SharePoint team was to make it easier to modify the key aspects. This means there will be alternatives to SharePoint Designer, but tight integration for designers and power users will still be SharePoint Designer for their tool of choice. For users building workflows, SharePoint Designer will be the primary tool of choice. When choosing the tool you will use for development, keep in mind that every tool has its purpose, and if you use the right tool for the job, it will make your SharePoint development easier.

New Features in SharePoint Designer

Because this is a professional development book, you will not see deep coverage of SPD here, but you still should know the primary features and enhancements in SharePoint 2013 beyond the user interface that is provided. These include features such as integration with Business Connectivity Services, Visio Integration, and enhancements to workflow.

Improved Workflow Experience

SharePoint Designer has become the one-stop shop for building, packaging, and installing workflows to SharePoint sites. This section covers the improvements within SharePoint Designer around workflow. For more information on building workflows for SharePoint, see Chapter 15.

Two types of workflow platforms are available in SharePoint Designer: the SharePoint 2010 workflows and now the SharePoint 2013 workflows. The workflow enhancements in SharePoint Design 2010 and the SharePoint 2010 workflow framework still work in SharePoint Designer 2013, but there is now the ability to create SharePoint 2013 workflows with the same level of support in the product. This enables users to maintain work already done with previous versions of SharePoint and the workflows that already are in place today, while creating new workflows that are maintainable and upgradeable in the future. Any time you select to create a new workflow, you will be prompted for the workflow platform type, as shown in Figure 3-8.

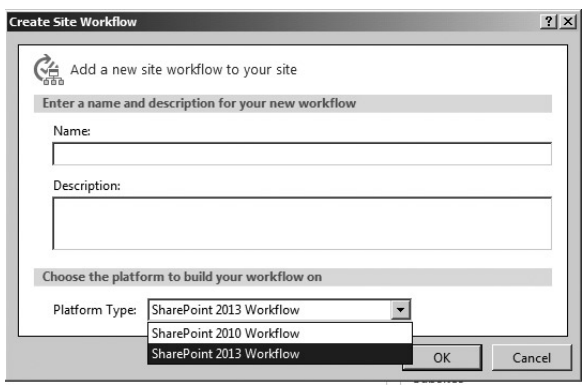

FIGURE 3-8

The new SharePoint 2013 workflow platform type is built using Windows Azure Workflow services. This requires that the SharePoint site that you connect SharePoint Designer with must have Windows Azure Workflow services installed prior to creating SharePoint 2013 workflows. The reason you need to have the new workflow service installed is because the underlying actions in SharePoint Designer must be able to communicate to the service for details on how the action will work. If you use Microsoft Office 365, you have integrated Windows Azure Workflow without needing to install any additional software, but you need to install the service locally if you use the On Premise installation.

New shapes and control of workflow is provided with the Windows Azure Workflows. These new shapes available in SharePoint Designer are Stages, Loops, and Steps. These new shapes enable branching and looping logic that was provided by the new SharePoint 2013 workflows. No longer is it required to use Visual Studio Workflow Designer for support of looping. The new shapes can be added to the designer surface within SharePoint Designer by either dragging and dropping them or using the workflow Ribbon bar to add them in the selected location. When using SharePoint Designer to build your workflows, the required elements of the new shapes are added for you when they are placed on the design surfaces.

The Visual Designer view in SharePoint Designer extends the ability of business users and developers to work on workflows with Visio 2013 and SharePoint Designer. The layout of the Visual Designer provides the same rich representation as Visio with a graphical design surface and sets of shapes for use on the designer surface. To get the built-in functionality of Visual Designer, you must have Visio 2013 installed on the same machine as SharePoint Designer. You can see how quickly a workflow can be started by using this Workflow Visual Designer, as shown in Figure 3-9.

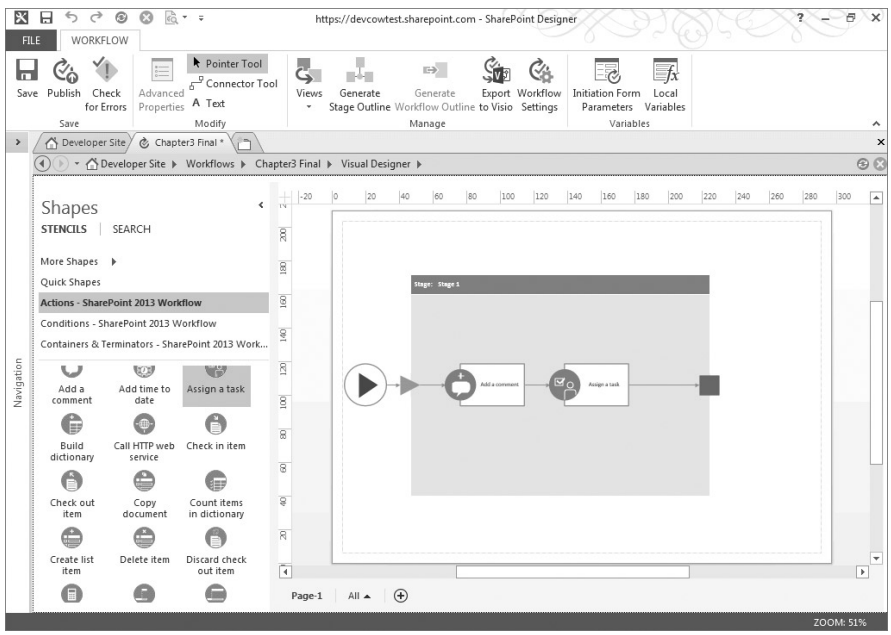

FIGURE 3-9

In addition to using the Visual Designer you can still switch the view back to the Text Based Designer that many users are accustomed to using with previous versions of SharePoint Designer. The changes that are made in the text-based designer are translated into the visual designer as well as the other way, too. If you need a high-level view of the stages, you now switch to the Stages view. The option still exists if you want to export to Visio and allow your business users to modify the workflow in a tool they commonly use.

Performing the same task multiple times in the workflow was difficult, especially if you wanted to quickly reuse some of the conditions or actions that had already been configured. The new Copy and Paste feature can help when designing large workflows that have some repeating information. Workflow designers can now use the same shortcut keys for copy and paste or the Ribbon bar to select a single action or an entire stage with all the contained actions. There are few limitations to this new feature such as no ability to use CTNL+Z to undo the last command. The copy and paste functionality is not like Excel or Word, and you cannot use control to select multiple objects or drag and drop items on the workflow. You can use this Copy and Paste feature in either the SharePoint 2013 Workflow type or the SharePoint 2010 Workflow type.

To make it easier to pass data around a workflow, the Dictionary type variable has been introduced into SharePoint 2013 Workflows. A Dictionary type has a collection of Name/Value pairs, and the value has a type. You can now create complex types that are stored in memory for use within the workflow. There are a number of actions that use the Dictionary type such as Build Dictionary, Count Items in a Dictionary, and Get an Item from a Dictionary, as well as actions such as Call HTTP Web Service. Some of the new actions create the dictionary object, whereas others use it to define the action.

New Workflow Actions

With the addition of Windows Azure Workflow, SharePoint Designer has added a number of new workflow actions that resolve a lot of the difficulties in the previous release. *Workflows* are composed of conditions and actions. *Actions* perform the functions you want, and you can customize workflows by writing custom actions. The new actions are designed for integration with SharePoint 2010 workflows and implement a similar custom action that was available on CodePlex for SharePoint 2010 workflows.

A new action that enables designers to call REST services and OData web services is called the Call Web Service Action. This action is designed to make an HTTP web service call and return the data in the JSON format. This action could be used to call any website that exposes web-based APIs, in addition to frameworks such as ASP.NET Web API endpoints. The importance of directly calling these services using the basic authentication supported in the RequestHeader is that you can now connect to data with dynamic structures. This action is available when you select the SharePoint 2013 workflows and can be added to the design surface using the new Shapes or the Actions drop down.

The Start Workflow Action has been added to allow SharePoint 2013 workflows to start SharePoint 2010 workflows' direction from the workflow. This provides an easy way to use the new workflow platform, but still use the investments that have been made in built and tested works using the SharePoint 2010 version. Just like all the other actions, you can add it directly from the Shapes or workflow Actions menu. When configured, your existing workflows are ready to use again without any modifications until you are ready to modify them.

Navigating the User Interface

SharePoint Designer provides a consistent user interface with the common Ribbon UI to help discover the tasks you can perform in SPD against your SharePoint sites. The navigation of SPD uses grouping of logical SharePoint artifacts for users to quickly navigate to the actions needed. This interface makes navigation and discovery of your SharePoint site and information architecture easier. Figure 3-10 shows SharePoint Designer in action.

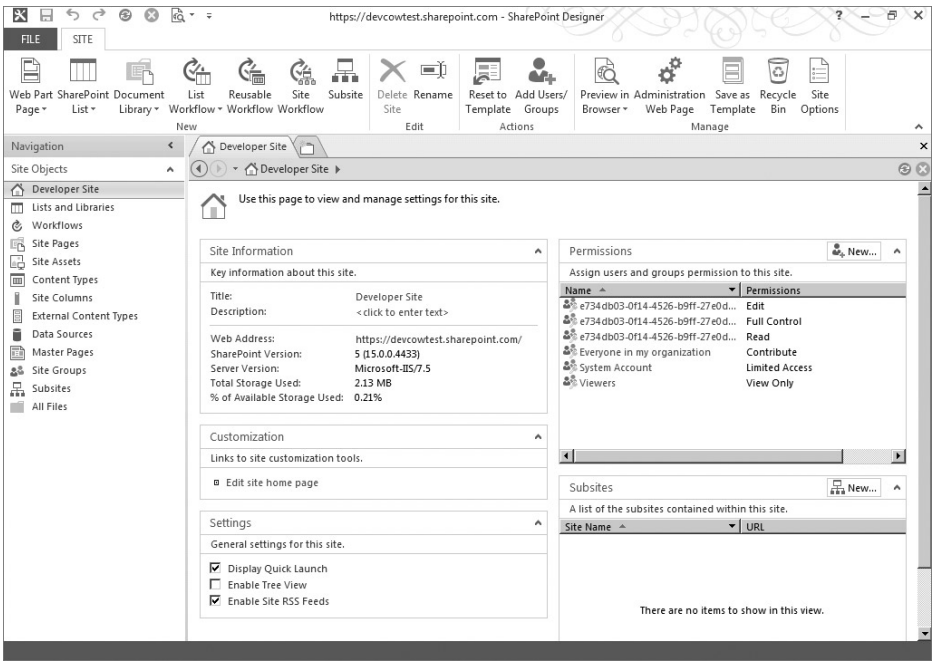

FIGURE 3-10

SharePoint Designer is tightly integrated to the SharePoint sites you are editing, and you must be connected to a SharePoint site to use the features of SharePoint Designer. After you connect to a site, the details display along with the permissions and settings. Connecting to a site is required for the information that is displayed and to allow for the direct changes to be made to the site. SharePoint Designer does not store the data locally but always makes the changes directly to the site after the Save and/or Publish buttons are pressed.

The navigation for SharePoint Designer 2013 has remained the same as the previous version with the site objects grouped in the navigation pane, which makes it easy to find what you are looking for. The site objects are grouped into common actions by the following categories: Site Information, Lists and Libraries, Workflows, Site Pages, Site Assets, Content Types, Site Columns, External Content Types, Data Sources, Master Pages, Site Groups, Subsites, and All Files. From the Navigation menu you can begin changing the content of the site as needed. Select the site objects you would like to change, and the Summary menu shows a list of all the items on that site. For

example, if you select Site Page, you see a list of all the site pages with actions on the Ribbon to make modifications.

The Ribbon was introduced in SharePoint Designer 2010 and is used to perform actions in SharePoint Designer just like in Microsoft Office products. The Ribbon user interface makes it easier for you to work with site objects by showing contextual tabs based on the selected objects that you click. The Ribbon interface makes it simple to manage your site with all the options available in a single interface and grouped together.

Workflow developers new to SharePoint Designer can find the full and rich capabilities wanted for any development, testing, and quick changes. New workflows that can be created for a site are based on the list, reusable, or site workflow. In addition to the new features added, you still have the capabilities around the existing actions and conditions. Management of the workflows can be done directly in SharePoint Designer as well as publishing the new workflows created.

Having the ability to create, design, and work with Business Connectivity Services (BCS) in SharePoint 2013 allows users to quickly manage their external content types. Using SharePoint Designer you can view your external content types and make new ones. The external content types can connect to databases, .NET types, or web services, and you can have SharePoint Designer autogenerate the methods needed to perform your create, read, update, and delete (CRUD) changes, and finder/query operations against the back ends. Finally, you can create the external lists associated with your external content type in SharePoint Designer. You learn more about the BCS in Chapter 13, so jump ahead if this topic interests you.

UNDERSTANDING VISUAL STUDIO 2012 TOOLS

SharePoint development with Visual Studio has become the primary developer tool familiar to most custom developers. Visual Studio 2010 made SharePoint development tools a first-class citizen inside of Visual Studio, and the improvements build on this progress. Visual Studio 2012 ships with a number of new templates and tools that make SharePoint development easier. The SharePoint and Microsoft Office development experience has been bridged and provides many enhancements.

In addition to the Visual Studio changes and enhancements, the SharePoint and Office teams focused improvements on some of the critical areas of need. The primary areas of focus were on the common tasks performed such as working with lists, debugging, and testing. You can see many new Visual Studio Templates as well for quickly building new apps and Office components. There are new Visual Designers, which provide the comfortable interaction similar to other data used in Visual Studio. Office 365 has become a popular tool for many organizations to leverage SharePoint 2013. To enable customizations on the Office 365 platform, Visual Studio now also enables Publishing SharePoint Solutions to the servers remotely, directly within Visual Studio.

With the introduction of apps both for SharePoint and Office as a recommended option for building enhancements in SharePoint, Visual Studio has added support for the four stages of developing apps. These stages are start, design, develop, and publish. Visual Studio provides the tools to accomplish both the develop and publish stages of building your app. All the hosting models for app development are supported as well as the ability to debug your apps when they are built. No matter which

type of integration or SharePoint UX extensions you select, the development experience within Visual Studio remains the same.

SharePoint 2013 focuses more on web development than ever before. The client-side frameworks, REST-based endpoints, and standards-based code now enable developers to create powerful applications without major changes to many components. Visual Studio improves the experience for developers with added IntelliSense for JavaScript. This can be one of the most difficult and time-consuming development tasks because the language does not show errors with a compilation like compiled code. To assist developers with this issue, Visual Studio now supports debugging the JavaScript.

> **NOTE** *The SharePoint 2010 templates for Visual Studio will be installed by default with Visual Studio 2012. These project templates would still work for non-deprecated features, but you should use the SharePoint 2013 templates for any new development.*

Finally, there have been improvements to building components such as Web Parts Project Item templates. To make it easier to create Silverlight components, there is now a Silverlight Web Part template. You can use this template to add your own Silverlight application or create one with Visual Studio. This might be useful if you know that Silverlight is supported in an environment such as an intranet and allows for another way to present data and information. The Sandboxed Visual Web Part is also a part of Visual Studio templates provided to developers. The SharePoint Project templates have been updated and streamlined to allow for a clean development experience. This process included moving some project templates to the project items list to allow for fewer items to select from when creating a Visual Studio project.

Before diving into the project types that Visual Studio supports for SharePoint, spend some time doing a quick walk around Visual Studio and the capabilities it provides for SharePoint regardless of the project type you select. These features include the templates for SharePoint 2013, the ability to import Web Solutions Packages (WSPs) in the Visual Studio environment, SharePoint Server Explorer node integration, exploring the Project Explorer, and finally the changes to the Package Designer that make it possible to build and deploy both SharePoint apps and SharePoint WSP packages.

Starting a New SharePoint 2013 Project

To start building solutions for SharePoint 2013 with Visual Studio 2012 you need to install the Microsoft Office Developer Tools for Visual Studio 2012. This set of tools installs all the project templates needed for apps, Microsoft Office, and SharePoint development. The developer tools are delivered using the Web Platform Installer (WebPI) and fully configure the system during the installation. The tools install on Visual Studio Ultimate, Premium, or Professional, which must be installed prior to installing the tools. The default target platform for the developer tools is x86-bit platforms, and you must install the required x64-bit assemblies separately for systems built targeting x64-bit hardware.

> **NOTE** *The page* `http://msdn.microsoft.com/en-us/office/apps/fp123627` *has a list of apps you can build and what you should download, as well as a link to the WebPI package needed for SharePoint 2013.*

After the tools are installed, you can create a SharePoint 2013 project by using the New Project menu, as shown in Figure 3-11. Depending on the language of choice, you can select either C# or Visual Basic, and then select Office/SharePoint to see all the project templates for the three categories: apps, Office, and SharePoint.

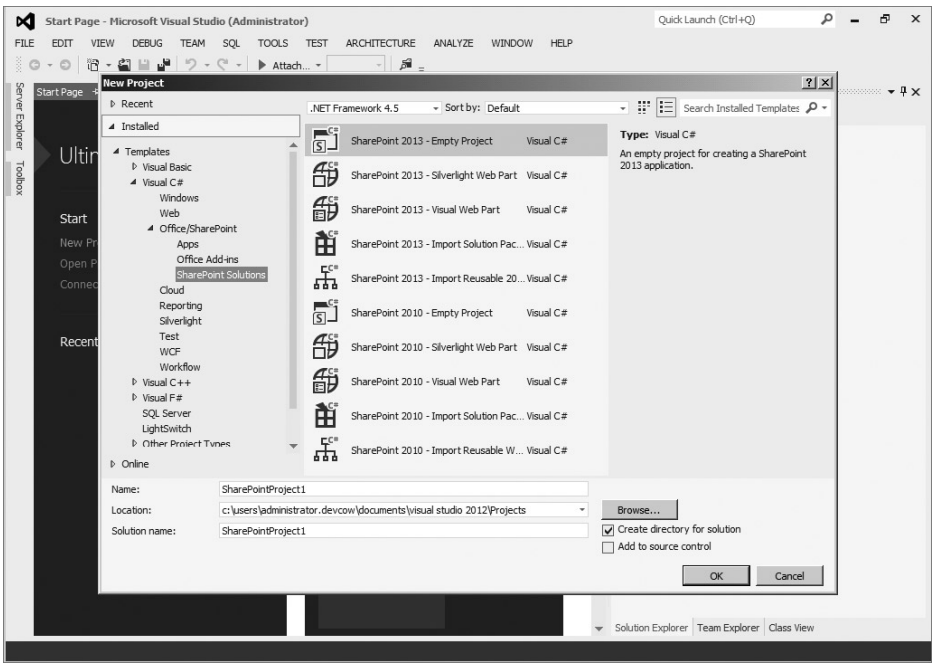

FIGURE 3-11

Visual Studio Integrated List and Content Type Support

One of the most common tasks associated with Visual Studio was creating lists and content types. This could be difficult at times because it required modifying XML files. There have been major improvements with the new editors that are provided and project items. To help assist in building out a content type, Visual Studio provides a project template item to create a custom list schema for use as a reusable column definition. Visual Studio also provides the new Content Type Editor that provides a visual interface for the name, type, and required fields of the XML, which generate the Elements.xml behind the scenes. This editor capability can be seen in the enhanced list editing experience. The List Editor provides the ability to change the list, views, and properties.

SharePoint Connections in Server Explorer

The Visual Studio Server Explorer provides a powerful way to visually represent different components of your server infrastructure, such as browsing through your data connections, services, event logs, and performance counters. When developing against SharePoint, you may want to browse your SharePoint site to understand what content types, fields, workflows, lists, and libraries are on your site. With the SharePoint Connections in the Server Explorer, you can see all this information inside of Visual Studio with a tree view of the site as well as browse the properties of these items. The SharePoint Connections in Server Explorer are read-only and cannot be used to modify the properties. Server Explorer saves you the time required to view the structure, look at properties, and open them quickly in a web browser. Figure 3-12 shows the SharePoint Connections within the Server Explorer.

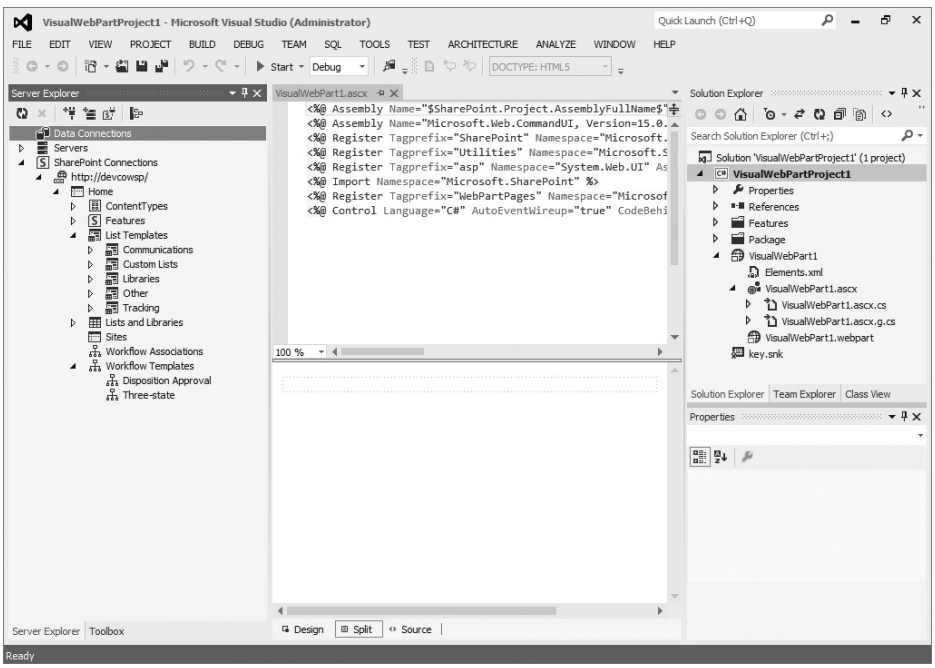

FIGURE 3-12

Solution Explorer Integration

As part of the Visual Studio experience, the SharePoint tools for Visual Studio integrate with the Solution Explorer so that you can see the files that make up your solution. By default, when you select your project type, Visual Studio creates all the projects and files needed for your solutions, such as the feature XML file, the package XML file, and a key to sign your features, so you can deploy it. In addition, Visual Studio logically lays out your solution so that you can quickly add new features or other projects to it.

Mapped Folders

Starting with SharePoint 2010, Visual Studio introduced the concept of Mapped Folders. Mapped Folders provides a quick solution to get files into the SharePoint Root or SharePoint Hive (`%Program Files%\Common Files\Microsoft Shared\web server extensions\15`). This was extremely difficult to do prior to the integration with Visual Studio due to how deep the files are buried in the filesystem. You could use different techniques, such as creating Windows Explorer shortcuts, to get to the different folders quickly, but that doesn't help you inside of Visual Studio projects, where you want to plan an image or add an artifact to the Layouts folder.

The Mapped Folders provide a way within the Visual Studio project to map to a designated SharePoint folder such as the Layouts folder in the SharePoint root. To add a Mapped Folder, you simply right-click your project in the Solution Explorer, and under the Add menu, you see three commands: SharePoint Images Mapped Folder, SharePoint Layouts Mapped Folder, and SharePoint Mapped Folder. The last one displays a user interface for you to select the folder you want to map to. By using these capabilities, you can drag and drop items into your Mapped Folders, and Visual Studio will deploy your artifacts to the right location in SharePoint.

Applications for SharePoint

The new SharePoint app model brings along with it new experiences for developing for SharePoint. The primary difference is more reliance on web-based technologies such as JavaScript. Luckily with Visual Studio there have been enhancements to the JavaScript IntelliSense features to make it a more fluid development experience. There are new Visual Studio templates for supporting the app model, as well as a Project Layout and Packaging framework. All apps for SharePoint can be developed within Visual Studio, but you need a developer site to run and debug the apps, as shown in Figure 3-13.

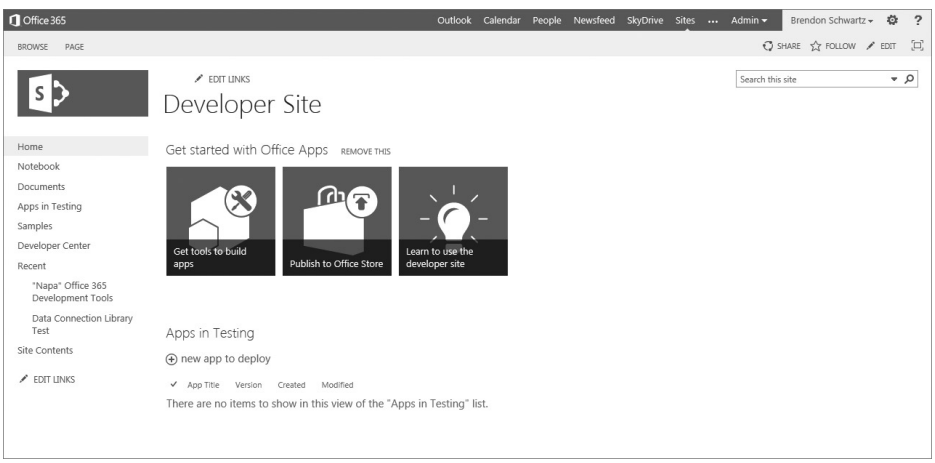

FIGURE 3-13

SharePoint Solutions Project and Item Type Templates

The new Project Type templates needed for supporting apps is straightforward, with only a single project type needed for the SharePoint apps that you will build (see Table 3-2). After you create the project, you can build any of the specific app designs based on what the functionality of your app needs (see Table 3-3).

TABLE 3-2: App Project Type Templates

NAME	DESCRIPTION
App for SharePoint 2013	The project type enables developers to create apps built for SharePoint and include the ability to choose the hosting type and UX experience.
App for Office 2013	The project type is used to create apps inside of Office 2013 to provide additional content and functionality.

TABLE 3-3: App Item Type Templates

NAME	DESCRIPTION
List	This template now provides the ability to create a list with a custom set of fields or create a new list from an existing list.
Remote Event Receiver	This template enables you to create a remote event receiver to handle SharePoint events using a remote service.
Content Type	This template provides a wizard for creating a content type item with a reusable collection of fields.
Workflow	This template provides a wizard that enables you to create SharePoint 2013 workflows that can be based on a list or site in Visual Studio.
Empty Element	This template creates an elements.xml file that enables you to define SharePoint artifacts using XML. The most common usage would be defining a field in your SharePoint project.
Site Column	This template creates the elements.xml file and default field attributes for the custom site columns that can be used in the fields or content types.
Module	This template creates a simple module file with a sample text file showing how to deploy files.
Client Web Part (Host Web)	This template creates the elements.xml needed to host an app for SharePoint inside of a web part called a Client Web Part.
UI Custom Action (Host Web)	This template creates the elements.xml needed to create a SharePoint custom action that links to an app for SharePoint as the resulting URL action.

continues

TABLE 3-3 *(continued)*

NAME	DESCRIPTION
Task Pane App	This template provides a wizard for creating an app for Office 2013 with the Task Pane option selected and the available Office applications selectable.
Content App	This template creates an app for Office 2013 for building out content that appears in the body of the Office documents.

Files and Project Layout

The new SharePoint App template in Visual Studio creates the solution with the project, project items, and files needed to get started. The basic code with all the required properties are set up with the wizard. The files created will be based on the type of hosting model you select for the apps for SharePoint.

The cloud-hosted apps with either the Auto-Hosted or Provider-Hosted apps have the same folder and files structure. The difference between the two is how Visual Studio handles packaging, deploying, and debugging the apps based on where they are hosted. The projects contain an App Project and a Web Application Project. The App Project contains the app-specific files, whereas the Web Application Project contains the files needed to host the app. The following files are required for the app projects:

- *AppIcon.png* — This image is the one used to display on the homepage.
- *AppManifest.xml* — This file contains the app elements just like the elements and feature .xml files in SharePoint Solutions.

The Web Application Project contains a specific file called the TokenHelper file that enables your app to make secure access calls into SharePoint resources. This is accomplished by using the access tokens defined per application and stored within the app. For more details on building apps, read Chapter 6 for deep-dive details. The other files are all standard web files with the app framework built into them, such as the ClientID and ClientSecret values stored in the Web.config file.

The SharePoint-hosted apps provide a slightly different layout because the packaging for a SharePoint Solution is needed in addition to the app components. Also, these projects are hosted within SharePoint and do not need a separate web application because SharePoint is a web application. If you are familiar with traditional SharePoint Solutions, you can quickly see that this solution is similar to those in SharePoint 2010 and that the project layout is similar with Features, Packages, and HTML folders. Because this is still an app, you still need the AppManifest.xml file that will be created for you from the Project template.

Depending on the type of project you are working with, either the App Project or the Web Application Project, you can set the required project properties from the Properties window, and you can use the built-in designers for any support tool such as the AppManifest.xml Editor, Feature Editor, and SharePoint Packaging Editor. Figure 3-14 shows the Autohosted app with the App properties and AppManifest.xml Editor opened.

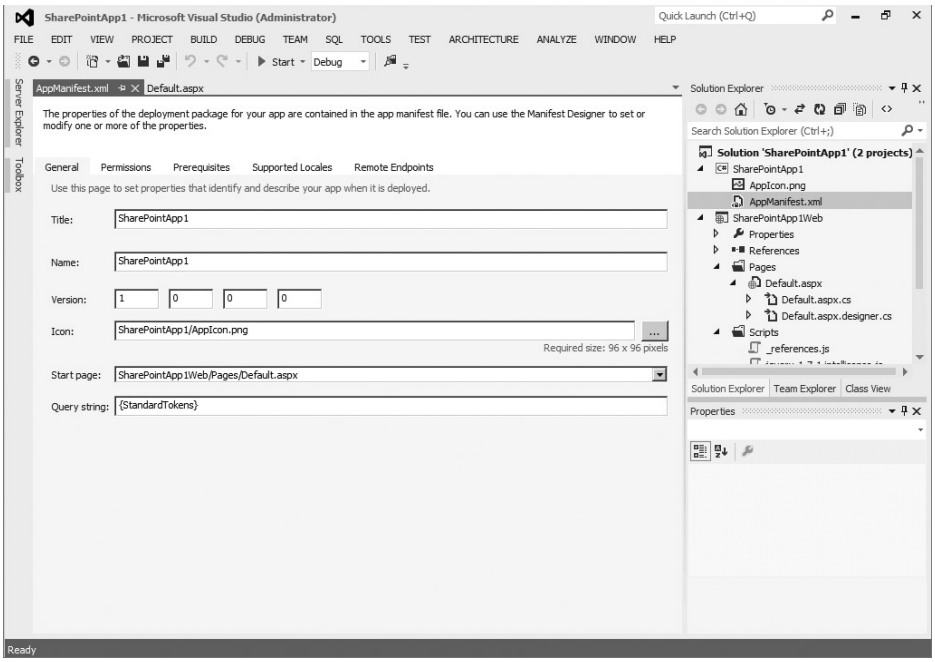

FIGURE 3-14

Packaging

A new packaging framework is used specifically for apps based on the Open Packaging Conventions (OPC). This new packaging format is what enables apps to be hosted outside of SharePoint and even integrated into Microsoft Office. Each app for SharePoint package has the extension of .app. The file that drives all the supporting app files is the AppManifest.xml file, which contains the properties and links to other files. This file is required in an app to allow it to be packaged correctly. In addition to the required app files, there could also be other packages for SharePoint (WSP), Resource files (RESX), Data Tier Application Packages (DACPACs), and Web Deploy Packages.

> **NOTE** *The .app packaging format, which is based on the Microsoft Office format, is essentially .zip files. To view the contents of an app, just rename the file extension to .zip, and you can view it in Windows Explorer.*

To complete the packaging, you need to decide where you will publish your app to for other users to consume. There are two places that you can publish your packages to:

- *The public Office Store* — This enables other users to view and download your app.
- *An internal organization app catalog* — This option enables you to create internal organization apps for users of your internal deployment.

There is a new Publish Office apps Wizard that can guide you through final packaging of your app for SharePoint for publishing. This wizard walks you through the process and asks different questions based on the type of app for SharePoint you have selected. During this process you must provide the identity of your app with the client ID and the client secret that was found in the web.config. After you work through the wizard, Visual Studio automatically generates the files needed to publish your app. To see these files you can navigate to the <app>\bin\Debug\app.publish folder in your project. All apps contain the .app file, which can be uploaded to the right catalog for deployment. If there is also a Web Application Project, Visual Studio generates a few files in addition to the required web application files that are stored in a .zip file in the same directory that are used during Web Deploy:

- *ProjectName*.`deploy.cmd` — The batch commands used to deploy your package.
- *ProjectName*.`SetParameters.xml` — The parameters used in the deploy.cmd file.
- *ProjectName*.`SourceManifest.xml` — Provides the files and layout of the package used only when creating the package itself.

Apps for Office

These apps are critical for business apps because they work in both Office Applications and Office Web Applications. This enables apps for Office to run inside of SharePoint 2013 without any changes. These apps are built with the same concept of portability and use standard web technologies such as HTML, CSS, REST, JavaScript, and more. Currently, the supported Office Web Applications that are supported are Excel and Outlook; although the Rich Office Application also supports Word and Project. Other Office Applications will be supported in the future as well as enhancements to current applications. These apps are displayed using the IE9 add-in, so all HTML 5 is supported just like the browser.

SharePoint Solutions (Classic Solution)

SharePoint Solutions, which are now referred to as Classic Solutions, are recommended only for administration automation and tasks. To create these solutions you must install SharePoint on a Windows Server machine and develop locally with Visual Studio.

Feature Designer

Building SharePoint Solutions as Classic Solutions will still require using the Feature Designer to manage your solutions features and package. A feature can have multiple items in it, such as a Delegate Control or Event Receiver. In addition, features can be dependent on the activation of other features. For example, Feature A may require that Feature B be activated. Features are also scoped to different levels in SharePoint at the Farm, Site, Web, and Web Application level. The Feature Designer enables you to configure your features with all this functionality. Figure 3-15 shows the Feature Designer.

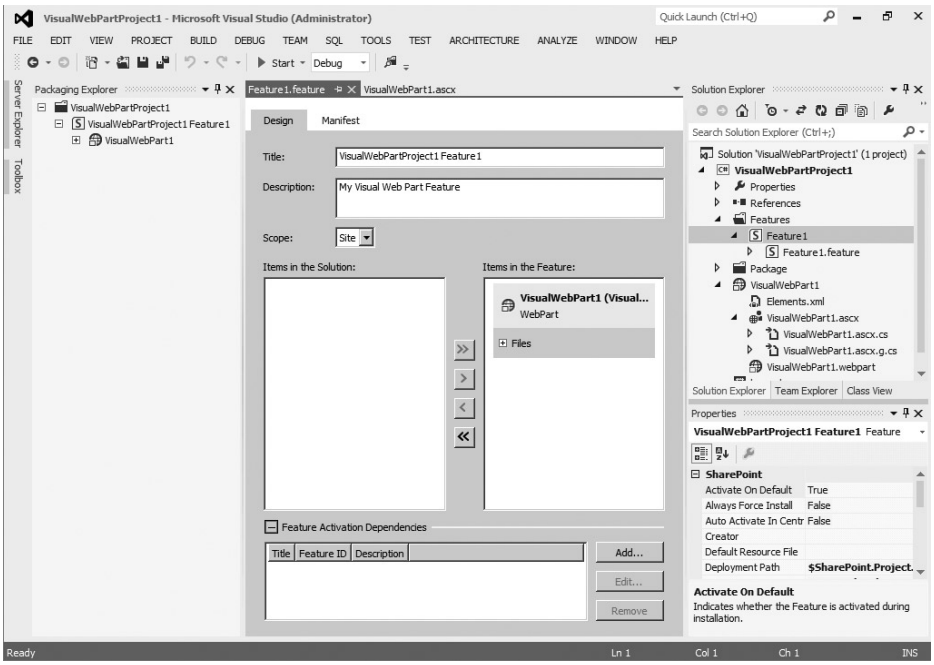

FIGURE 3-15

In addition to working with the graphical designer, you can also work with the XML that Visual Studio creates for your feature. You have two choices in working with the XML. First, you can add custom XML to the auto-generated XML that Visual Studio creates. Second, you can edit all the XML, even the auto-generated parts. If you get your edits wrong, it could stop you from working with your feature in Visual Studio. Editing all the XML is recommended only for advanced users who cannot meet their needs by inserting new XML into the Visual Studio auto-generated XML.

SharePoint Solutions Project and Item Type Templates

The project types for Visual Studio 2012 are based on a combination of SharePoint 2010 and SharePoint 2013 project templates. There are some overlapping items for each version, but all project templates are listed together under the Office/SharePoint ➪ SharePoint Solutions category. Table 3-4 lists the different project type templates, and Table 3-5 lists the item type templates that you can use. Many of the project types have been moved to item types to make it easier to find what type you are looking for.

TABLE 3-4: SharePoint Solutions Project Type Templates

NAME	DESCRIPTION
SharePoint 2013 Project	This templates enables you to start with an empty project that has all the necessary elements for you to get started, such as folders for references, features, and solutions, and a key to strong-name your assembly.
SharePoint 2013 Silverlight Web Part	This template provides the files need to create a SharePoint 2013 package that hosts a Silverlight application and also provides the web part project with which to associate the Silverlight application.
SharePoint 2013 Visual Web Part	This template creates a new SharePoint 2013 Visual web part, which enables you to drag and drop controls onto your web part for your user interface rather than writing the user interface in code. It contains a Web Part and a User Control item.
Import SharePoint 2013 Solution Package	This template enables you to import an existing WSP package for SharePoint 2013.
Import Reusable SharePoint 2013 Workflow	This template enables you to import an existing reusable SharePoint 2013 Workflow that you create in SharePoint Designer 2013, which you can then customize and deploy from Visual Studio. The import is one way, and after it is modified in Visual Studio, you cannot go back to SharePoint Designer.
SharePoint 2010 Project	This templates enables you to start with an empty project that has all the necessary elements for you to get started, such as folders for references, features, and solutions, and a key to strong-name your assembly.
SharePoint 2010 Silverlight Web Part	This template provides the files needed to create a SharePoint 2010 package that hosts a Silverlight application and also provides the Web Part project with which to associate the Silverlight application.
SharePoint 2010 Visual Web Part	This template creates a new SharePoint 2010 Visual web part, which enables you to drag and drop controls onto your web part for your user interface rather than writing the user interface in code. It contains a Web Part and a User Control item.
Import SharePoint 2010 Solution Package	This template enables you to import an existing WSP package for SharePoint 2010.
Import Reusable SharePoint 2010 Workflow	This template enables you to import an existing reusable SharePoint 2010 Workflow that you create in SharePoint Designer 2010, which you can then customize and deploy from Visual Studio. The import is one way, and after it is modified in Visual Studio, you cannot go back to SharePoint Designer.

TABLE 3-5: SharePoint Solutions Item Type Templates

NAME	DESCRIPTION
Silverlight Web Part	This template adds the required Silverlight project and asks how you would like to associate the web part in the project by either creating a new Silverlight Web Part or associating it later.
Visual Web Part	This template adds a new Visual Web Part to the current solution.
Web Part	This template enables you to create a web part for your SharePoint environment.
List	This template now provides the ability to create a list with a custom set of fields or create a new list from an existing list.
Event Receiver	This template provides a wizard that enables you to select the type of event receiver to create, the event source it is created with, and the events you would like to implement.
Content Type	This template provides a wizard to create a content type item with a reusable collection of fields.
Workflow	This template provides a wizard that enables you to create SharePoint 2013 workflows that can be based on a list or site in Visual Studio.
Workflow Custom Activity	This template creates a custom activity that can be reused in Visual Studio or in SharePoint Designer.
Sequential Workflow (Farm Solution Only)	This template provides the ability to create the SharePoint 2010 sequential workflows.
State Machine Workflow (Farm Solution Only)	This template provides the ability to create the SharePoint 2010 state machine workflows.
Business Data Connectivity Model	Use this template to create a resource file for your BCS model. A resource file enables you to localize the names in your model and apply permissions to objects.
Empty Element	This template creates an elements.xml file that enables you to define SharePoint artifacts using XML. The most common usage would be defining a field in your SharePoint project.
Application Page (Farm Solution Only)	Use this template to create an application page, which is just an ASP.NET page hosted in SharePoint.
Site Column	This template creates the elements.xml file and default field attributes for the custom site columns that can be used in the fields or content types.

continues

TABLE 3-5 *(continued)*

NAME	DESCRIPTION
Module	This template creates a simple module file with a sample text file showing how to deploy files.
Site Definition (Farm Solution Only)	This template enables you to create the SharePoint 2010 site definition files that can be deployed at a farm level.
User Control (Farm Solution Only)	You can create a user control that you can use in an application page or web part with this template. You can design the control using the graphical designers in Visual Studio by dragging and dropping your controls onto the design surface.

Importing Packages

The concept of importing and exporting SharePoint packages has been around in the user interface since SharePoint 2007. Since then the product team has improved the capabilities and standardized on the packages that are created. The capabilities are now found in SharePoint, Visio, SharePoint Designer, and Visual Studio. This means that the combination of those tools enables you to develop your solutions quickly and use the tool that is right for the step in development. Visual Studio provides the largest set of capabilities for completing the packages and fine-tuning changes. To import SharePoint packages known as Web Solution Packages (WSP) there are two options: either a generic WSP package or a Reusable workflow. Both of the packages have a Visual Studio Project template with wizards to help guide you through importing the packages. If you are unfamiliar with WSPs, you should learn more about the internals of how they work because they provide valuable features that enable you to install and ship your SharePoint Solutions to multiple environments.

Importing WSPs

When doing SharePoint development, you must perform your work inside of the user interface or SharePoint Designer. After you complete designing your solution, you must export items or even the entire sites to move the information around and modify it. With Visual Studio you can import the WSP solution, which contains the exported site or items that you have exported to move into your Visual Studio solution. Visual Studio imports your lists, fields, content types, and other artifacts, so you can start working on them quickly in Visual Studio. WSPs are still the recommended packaging format for all SharePoint Solutions that are not apps.

Reusable Workflows

Similar to the project template for importing a generic site WSP, you can also import a reusable workflow that you created using SharePoint Designer. The reusable workflows created in SharePoint are declarative workflows that consist of XML statements to define the workflow instead of code. The reusable workflow template enables you to create your workflow in SharePoint Designer and then import it into Visual Studio to convert it to a code workflow that can be reused on your SharePoint sites.

Package Designer and Explorer

After you create your features, you need to package them together and deploy them to your server. This is where the Package Designer and the Explorer come into play. If you have used SharePoint previously, you know that SharePoint supports a format called a Web Solution Package (WSP), which is just a CAB file that contains your solution files and a manifest or XML file that tells SharePoint what to do with your solution when deployed. You could write all the XML yourself and compile your CAB file, but Visual Studio makes this much easier. Figure 3-16 shows the Package Designer.

The Package Designer gives you the ability to do the following:

- Add multiple items to the solution using a graphical interface
- Control whether the web server resets
- Add assemblies to your package
- Write package rules that enable you to validate your package programmatically before deploying it to the server

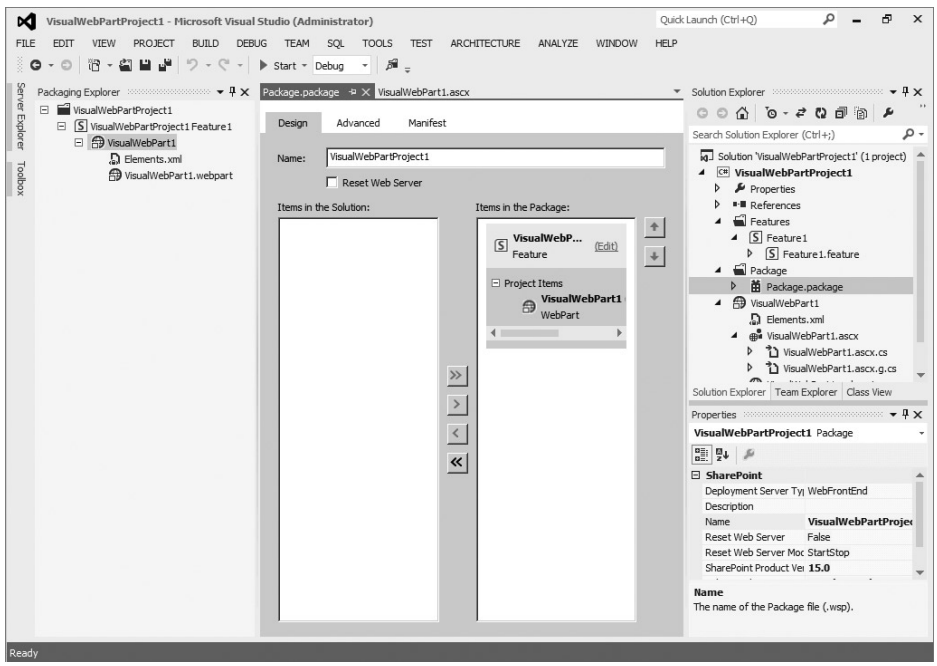

FIGURE 3-16

SETTING UP YOUR DEVELOPMENT ENVIRONMENT

There are many options for SharePoint development with the changes to SharePoint and Office 2013. With all these changes, understanding what you are trying to develop and the components needed to be set up is critical. The two most common development environments are the Office 365 solutions or the On Premises solutions. The new app model makes it easier to create development environments and has fewer dependencies than previous versions of SharePoint. This change also brings back a restriction that all classic development for SharePoint Solutions must be done on a server OS. In addition to setting up a local development environment, Microsoft has provided a full set of development tools for apps that are hosted on the web for quick development.

Applications for SharePoint and Office 365 Development Environment

The steps to create an environment to develop apps are simple in SharePoint 2013. Microsoft has recommended that app developers sign up for an Office 365 Developer Site to help with development and debugging. These sites are already configured with the required app isolation and OAuth that would be required to set up in a local SharePoint deployment. Also you get the full deployment experience from Visual Studio, and you can deploy only to the Developer Site. As discussed earlier all you need to do is install Visual Studio on any support operating system for Visual Studio that includes Windows 7. After Visual Studio is installed, you install the Office Developer Tools for Visual Studio, which includes the following necessary developer components:

- Office Developer Tools for Visual Studio 2012 — Preview
- SharePoint Client Components (containing the client assemblies)
- Windows Identity Foundation (WIF) SDK
- Workflow Tools SDK and Workflow Client SDK
- Windows Identity Foundation SDK and Windows Identity Foundation Extensions

Napa Office 365 Development Tools

The Napa Office 365 Development Tools are a set of tools provided with Office 365 that enable apps developers to start quickly without installing any tools locally. To develop apps you can use the full code editor with syntax highlighting that is provided in the browser. To get this tool you must sign up for an Office 365 account and create a developer site that enables you access the tools. If at any time you want to continue to edit your solution in Visual Studio, there is a button to open your project in Visual Studio. This is a straightforward way to create your apps, as shown in Figure 3-17.

When you have an Office 365 account, you can access the editor directly from the URL `https://www.napacloudapp.com/`.

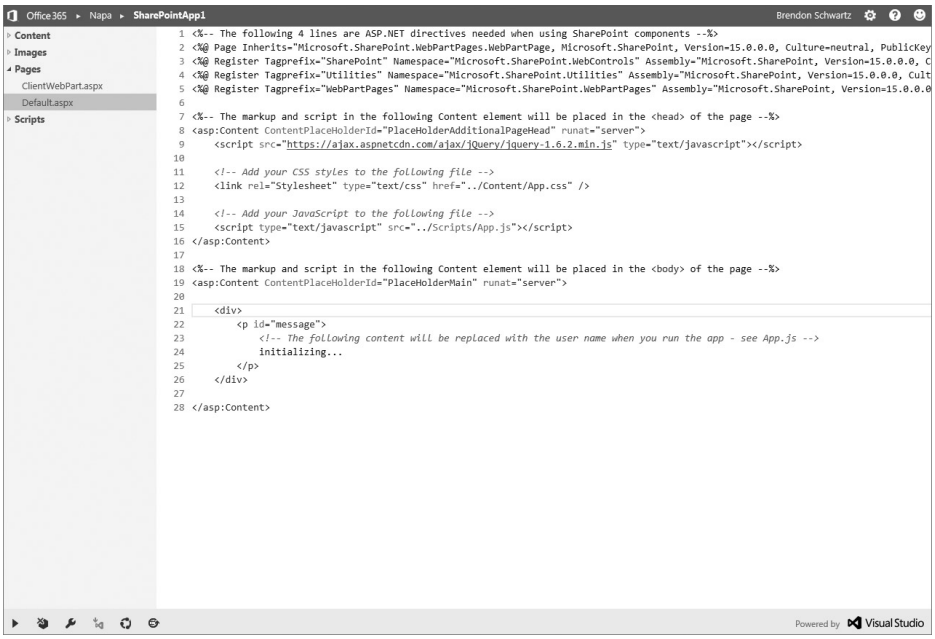

FIGURE 3-17

Local Development Environment

The local SharePoint development environment maps to the traditional SharePoint environment that current developers of SharePoint are accustomed to. This environment requires that you install SharePoint 2013 locally on a Windows Server 2008 x64-bit server to begin development. The major change is that you cannot perform the installation of SharePoint 2013 on a Windows 7 operating system and therefore cannot do classic SharePoint development on these system configurations.

System Requirements

The system requirements for a machine required for local development are not as large as a production environment, but you should be aware of a couple key requirements when setting up your development workstation. Because SharePoint 2013 has only x64-bit installations, your development machine also needs to be an x64-bit machine. The current recommendation is that the machine has at least 6 GB of RAM to install and run SharePoint 2013. This is less than the single server instance of the production hardware, and you should optimize the development environment when working with less memory.

Virtual or Physical?

This local development environment can be installed either on virtual or physical hardware depending on what systems are available and your budget. Installing SharePoint virtually or physically on

your machine is usually a tough decision. Many times, the answer depends on the operating system you want to run on your guest OS and also whether you want to trade off performance for flexibility. Now step through each issue in a little more detail.

For the host OS, if you don't mind using Windows Server 2008 as your primary operating system, you will have many options for installing SharePoint (whether that's physical or virtual) because Windows Server 2008 supports Hyper-V. When you know the hardware and software, you can decide whether you want a physical or virtual environment. The advantages of Hyper-V for a developer are that you can have an isolated development environment that can be copied or moved to another location.

> **NOTE** *To install a local development environment using Hyper-V, your hardware must support Hyper-V.*

If you want to run on a desktop operating system such as Windows 7, your choices are more limited because these desktop operating systems don't support Hyper-V. This means that if you want to virtualize, you need to use another product such as VMWare or Virtual Box because Virtual PC and Virtual Server don't support x64-bit.

When you have the right virtualization technology for your host OS, the question becomes whether to virtualize. Virtualization provides a lot of nice features, such as portability, ability to roll back changes, different environments on a single host OS, and so forth. With all these positives to virtualization, there is also a negative with the cost of performance. Of course, this performance cost has decreased over the years with improvements to software and hardware changes. The reason for the performance impact is that you need to give the guest OS and SharePoint a few GBs of memory, and you definitely need a fast hard drive, preferably 7200 RPM and above. If you have the necessary hardware and you're developing solutions, the first choice should be virtualization. One last option that developers have started to look at is the dual-boot system with the Windows 7 dual-boot capabilities. This is sometimes not an option for larger organizations due to adding machines to a domain ad hoc but can be a quick way to evaluate a machine or use the full hardware.

> **NOTE** *Many scripts and deployment guides will be released as the product releases. For a full guide, check out the SharePoint Server 2013 Preview Virtual Machine Setup Guide (v0.5) in the free members section at* `http://www.criticalpathtraining.com/`.

SQL Server Version

SharePoint 2013 supports SQL Server 2008 R2 and SQL Server 2012. If you select the stand-alone option when installing the product, SharePoint installs SQL Server 2008 RS Express with SP1. Although this option installs the product quickly, you might run into issues with development if you try to access the database through Visual Studio. A good alternative would be if you have an MSDN subscription, you should use the SQL Server Developer Edition for a full set of features that you can develop with.

TROUBLESHOOTING WITH DEBUGGING

Development is the first part to create an application, but when there are issues with the code, having an easy way to debug the code and determine the issues is critical. Visual Studio and SharePoint provide the tools needed to do this during the development of the code as well as after the code has been deployed to a production site. The major changes for this release are around debugging SharePoint apps and the authentication needed to debug code running on remote servers. The Developer Dashboard has been given a makeover and now provides integrated debugging with logging output.

F5 Debugging

The standard for debugging in Visual Studio is called *F5 Debugging* after the shortcut key to start the debugger and attach the code to the running process. Visual Studio added F5 Debugging to the previous version of SharePoint for the classic SharePoint Solutions, and now that same feature can be used with SharePoint apps.

SharePoint Applications

Debugging apps for SharePoint requires a few more authentication handshakes than the classic SharePoint Solutions due to the architecture. Luckily Microsoft has made pressing F5 just as simple as it is for the classic SharePoint Solutions. By default, Visual Studio uses IIS Express as the local-host for debugging sessions. The IIS Express application is a portable version of IIS that does not need to be installed prior to running it. If there is interaction with the remote app and host web, you may be promoted to grant permissions prior to debugging. To debug the app for SharePoint, you need to be connected to SharePoint Developer Site and allow the application to be trusted if needed. If you are not, you will see an error like the one shown in Figure 3-18.

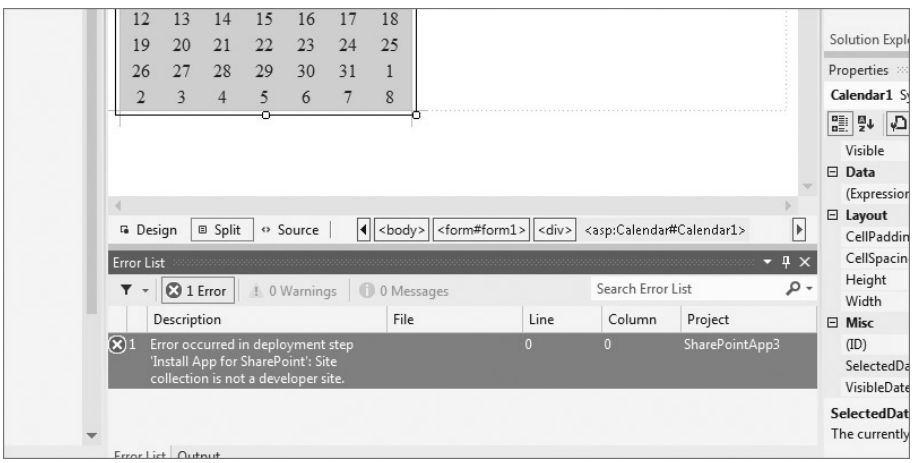

FIGURE 3-18

Following are the steps performed by Visual Studio when you press F5:

1. It builds the Web Application Project if needed.
2. It changes the URLs in the AppManifest.xml and other project files.
3. It packages only the SharePoint-specific artifacts into the Open Package Convention (OPC).
4. It uninstalls the SharePoint App.
5. It installs the SharePoint App (OPC) using the life cycle API.
6. It updates the Hosts file if Peer Name Resolution Protocol (PNRP) is not used.
7. It launches the Web Application Project in IIS or IIS Express if there is one.
8. It updates the web.config with the correct client ID.
9. If the LaunchUrl in the App manifest is set to a site page in the App Project, then launch the browser to the LaunchUrl. If not, Visual Studio just launches the All apps page.
10. Visual Studio attaches the script debugger or Silverlight debugger.

The debugging experience will be familiar to any ASP.NET developer with the web application projects. Because there may be a large portion of the code in JavaScript, it is nice to have the script debugger automatically attached and ready. The experience with the SharePoint-hosted apps contains more files to work through because all the JavaScript files will be loaded from the SharePoint page. A good strategy for quickly building your app would be to develop any of the Web Application components that do not rely on other frameworks first and then add the dynamic functionality after you have the code ready to go. It is a great debugging experience as long as the development environment and server locations are correctly configured. Make sure to do the pre-planning to set up your environment, and you can quickly build powerful apps and debug them by simply using F5.

Classic SharePoint Solutions

Using Classic SharePoint Solutions, you can allow debugging support by setting a breakpoint and starting the debugger by pressing F5. The same experience is provided if you select the Farm or Sandbox solution from the Visual Studio projects. The first time you debug a SharePoint solution in Visual Studio, you will be asked to automatically configure your web.config on the SharePoint server to support the debugging sessions. This is done to prevent the need for manual changes to the server for every development environment. Also, this helps reduce the mistakes and frustration of not remembering every change needed to begin debugging. These steps also include recycling the app pool, retracting the solutions, deploying the solution, and activating the required features. This entire behavior can be modified from the Visual Studio properties and editors.

> **NOTE** *You need administrative permissions to the server to change web.config from Visual Studio, and these changes should not be done on a production server.*

Visual Studio takes three steps on your behalf. First, it turns on the call stack in the web.config by adding the line CallStack=true. Second, it disables custom errors in ASP.NET so that you receive detailed error information if there is an error, using `<customErrors mode="Off"/>` in your system.web section. Lastly, Visual Studio enables complication debugging, which makes ASP.NET compile your binaries with additional information to make debugging easier using `<compilation debug="true"/>`.

Besides making these changes, Visual Studio performs a number of steps when you start the debugging session from deployment to attaching the debugger as follows:

1. It runs your predeployment commands that you can customize.
2. It creates your WSP using MSBuild places in the bin\<build> directory.
3. If you deploy to the farm, Visual Studio recycles the IIS application pool to free resources.
4. If you deploy a new version of an existing solution, it deactivates your feature, uninstalls your existing solution, and deletes the existing solution package on the server. If you have feature receivers, your code will be executed.
5. It installs your new solution and features onto the server.
6. If you build a workflow, Visual Studio installs your workflow assemblies.
7. It activates your Site or Web features. You need to activate Web Application or Farm features. Again, the feature receivers will be executed to run the code.
8. For workflows, Visual Studio associates your workflow with the list or library you selected in the Workflow Wizard.
9. It runs your post-deployment commands.
10. It attaches the debugger to the SharePoint process (w3sp.exe) for Full Trust solutions and to the SPUCSPUWorkerProcess.exe for Sandbox Solutions.
11. If you deploy to the farm, Visual Studio starts the JavaScript debugger.
12. Visual Studio launches your browser and displays the correct SharePoint site for your solution.

A few notes about these steps. First, if you debug a workflow, you need to trigger the workflow through the web browser, the client applications, or custom code that you have written. Visual Studio doesn't automatically trigger your workflow. Also for workflows, any additional assemblies you reference must be in the global assembly cache (GAC).

Second, if you work with feature event receivers, don't have Visual Studio activate that feature event receiver for you. Instead, manually activate your feature event receiver so that it is in the same process as the debugger. You can disable activation in your deployment in your project settings.

Because SharePoint builds on many layers below it, such as Windows Communications Framework (WCF), you may want to enable advanced debugging in your Visual Studio environment. To do this, go into the Registry Editor, find `[HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\10.0\SharePointTools]`, and change the `DWORD` value for `EnableDiagnostics` from **0 to 1**. If the `DWORD` value does not exist, create it as a new `DWORD` value. When you set this

value, you see in the output window in the Visual Studio all the information that Visual Studio gets form SharePoint via the stack trace.

Debugging Using the Developer Dashboard

The Developer Dashboard has become a must-have tool for any user of SharePoint. A total redesign of the Developer Dashboard was required to make it easier to use and to provide better performance. Now the Developer Dashboard is not just for developers but can be used by IT professionals because of the diagnostic information provided from the dashboard. The main use of the Developer Dashboard still remains to provide diagnostic information for the page that is rendered. The information can range from basic page information to the ULS logs associated with the page. One of the major changes is how the Developer Dashboard displays and how it gathers the information. Figure 3-19 shows the Developer Dashboard and its components.

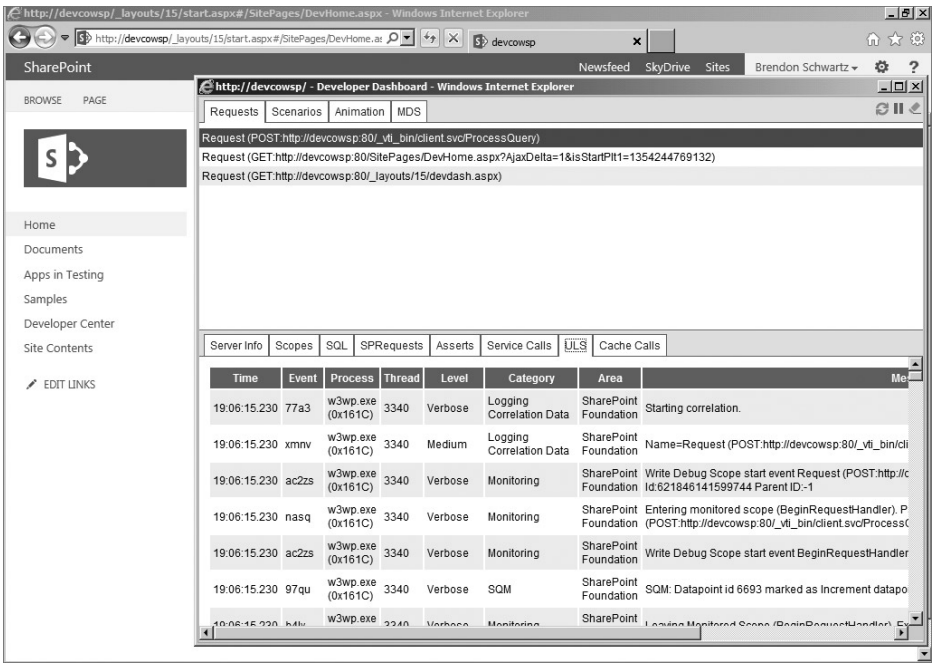

FIGURE 3-19

A number of tabs can now provide more detailed information about each request, and all the information is not in a single page. The trace information provides more information when available, such as the SQL tab, which enables you to click any request to see the detailed SQL command and analytics on each call. Also, you can see the ULS logs for the correlation ID for the page directly from the browser. The data that populates the browser pop-up window comes from a dedicated WCF service named *diagnosticsdata.svc*. This service was designed specifically for the Developer Dashboard and provides tracing information.

> **NOTE** *Diagnostic data is dependent on the Usage and Health Data Collection Service Application, which must be created and running.*

By default, the Developer Dashboard is off but allows a few options to turn it on. In the previous version of SharePoint, you could select three options, but SharePoint 2013 has only two modes, which are on or off. The On option is now equivalent to the On Demand option that places the icon on the page. The reason that there are only two options now is because the control is not embedded in the page, which means you don't need to worry about the control affecting content on the page. To enable the Developer Dashboard from PowerShell, you can use the following lines of script; just make sure to turn it off when you finish using it in production:

```
$contentService = ([Microsoft.SharePoint.Administration.SPWebService]::
  ContentService)
$devDashboardSettings =$contentService .DeveloperDashboardSettings
$devDashboardSettings.DisplayLevel =
  [Microsoft.SharePoint.Administration.SPDeveloperDashboardLevel]::On
$devDashboardSettings.Update()
```

This same script could be written in code because the PowerShell script and code use the same set of APIs. This depends on the usage you need to turn on the Developer Dashboard. For example, you might turn on the Developer Dashboard anytime you throw an exception in your code but turn it off on any other page render. As you can see, you need to add a reference and Using statement to Microsoft.SharePoint.Administration in your code. The code also needs to be run with the correct security context because the Developer Dashboard is a farm-wide setting.

Debugging Using SharePoint Logs

With the changes to the Developer Dashboard, you no longer need to go to the Unified Logging System (ULS) for a page. However, if you have other SharePoint Solutions or need to understand other actions happening at the same time as a page render, you still must use the SharePoint logs. The ULS logs contain logging information about actions that happen within SharePoint, so you might not see any logs related to the new apps for SharePoint. If you write apps hosted in SharePoint, you could still take advantage of logging to the ULS. Although, you could browse the ULS logs, using your favorite text editor. One great tool to check out is the ULSViewer, which is a free download from MSDN at `http://code.msdn.microsoft.com/ULSViewer`. It is an unsupported tool, but it is good at parsing the ULS logs and provides real-time viewing, smart highlighting (in which it highlights similar log entries when you hover over them), and a number of other features.

Debugging Silverlight Code

Visual Studio enables script debugging by default. If you want to debug a Silverlight application that runs in SharePoint, you need to change the properties under your project in the SharePoint section to check the Enable Silverlight Debugging check box.

Silverlight does not allow cross-domain scripting by default. If you make calls across domains, such as copying from one SharePoint site to another that may be in a different farm or uses a different URL, you need to become familiar with the *clientaccesspolicy.xml* file that you can use with Silverlight to override this policy. This will become common for developers building apps and including Silverlight on the web pages. You must place this file in the root of your SharePoint web server in the filesystem so that Silverlight has access to the new policy file.

> **NOTE** *MSDN has good resources to understand these restrictions at* `http://msdn.microsoft.com/en-us/library/cc645032(VS.95).aspx`.

Other Useful Tools for Debugging and Testing

Many community members have built tools for SharePoint since SharePoint 2007. Although these tools have not always been upgraded, many of the APIs in SharePoint have not changed, which means the tools may still work. Beyond Visual Studio, other useful tools can help you with debugging and testing in SharePoint. Following are some recommend ones, but there are more on popular sites such as CodePlex (`www.codeplex.com`).

SPDisposeCheck

One of the primary tools used for classic SharePoint solutions has been SPDisposeCheck. This tool enables scanning of code to determine if there are any memory leaks based on known patterns. The SharePoint APIs allocate COM-based memory that the Common Language Runtime (CLR) garbage collector does not release. For this reason, you need to explicitly call the `Dispose()` method on certain objects such as the `SPSite` and `SPWeb` objects. If you don't dispose of them, you get memory leaks in your application, and it can be hard to track down which pieces of your code are causing the leaks.

For this reason, Microsoft released a tool called SPDisposeCheck, which scans your code to tell you where you are not releasing this memory because of not calling the `Dispose()` method. This tool saves you a lot of time and heartache in tracking down memory leaks. You can download SPDisposeCheck from `http://code.msdn.microsoft.com/SPDisposeCheck`.

Internet Explorer Developer Tools

Sometimes the best debugging tools are the ones built right into the product. Internet Explorer provides you the ability to browse your HTML, script, and Cascading Style Sheets (CSS) code inside of its developer tools. To get to the developer tools, press F12 in the browser. You can debug your HTML and CSS by using the tree view and editing both sources on the fly. It also has a built-in debugger for JavaScript so that you can set a breakpoint and have the tools break when they hit your breakpoint. You get watch windows, local variables, call stacks, and an immediate window called the console. IE also includes a JavaScript profiler that shows you the performance of your script code, including the number of times a function was used and the amount of time it took. With these tools, you can track down any issues in your client-side code.

Firefox and Firebug

If you use Firefox as your browser, you can use Firebug as your HTML development and debugging tool. Firebug provides similar functionality to the IE developer tools in the Firefox environment.

Visual Round Trip Analyzer

The Visual Round Trip Analyzer (VRTA) sits on top of the network monitor tool from Microsoft and is a free add-on. It provides a graphic representation of how long it takes a client to talk to a server. This information can then be used to determine whether you are making excessive round trips, if your code is slowing down the pages (for example, because of loading many small JavaScript or CSS files), or if there are network issues causing any problems between your application and the server. You can download VRTA from `http://www.microsoft.com/downloads/details.aspx?FamilyID=119f3477-dced-41e3-a0e7-d8b5cae893a3`.

Fiddler

No discussion of debugging tools would be complete without mentioning Fiddler (`www.fiddler2tool.com`). Fiddler is a web debugging proxy that logs all HTTP and HTTPS traffic between your computer and the Internet. Fiddler enables you to inspect all HTTP traffic, set breakpoints, and view your incoming and outgoing data. It is an essential tool to help you understand what server variables are coming back from your server, what their payload is, how many calls your client-side code took, and other factors that provide insight into your applications.

SUMMARY

This chapter introduced the existing tools as well as new tools that accompany SharePoint 2013 and the changes to the app model. There are more options for OOB developer experiences from adding apps to building them with no installed tools. Each option for development has value and should be used together correctly to quickly build the applications needed. SharePoint has evolved into a developer platform with full tool support for almost every scenario. These tools were introduced to help you understand what is needed from a SharePoint developer to build compelling applications. Finally, you reviewed how debugging can now be performed with both the classic SharePoint Solutions and new app model as well as other tools that assist in tracking down the issues.

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

European SharePoint Community

Join the European SharePoint Community by following us:

BLOG

For more **FREE** SharePoint content such as webinars, presentations, eBooks, videos & more check out our Resource Centre.

To visit the Resource Centre please click **here.**

www.sharepointeurope.com