

Issue

6

UNDERSTANDING SHAREPOINT JOURNAL

Bjørn Furuknap

SP Themes and SP Sample Data Explained

UNDERSTANDING SHAREPOINT JOURNAL

SPTthemes and SPsampleData Explained

This book is dedicated to my wife.

© Understanding SharePoint
Rubina Ranasgt. 10
N-0190 Oslo, Norway
Phone +47 91 39 85 86 • Web <http://www.understandingsharepoint.com/>

Credits

About the Author



Bjørn Christoffer Thorsmæhlum Furuksnap is a senior solutions architect, published author of *Building the SharePoint User Experience*, speaker, and passionate SharePointaholic. He has been doing software development professionally for 16 years 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

Credits.....	i
About the Author.....	i
About Understanding SharePoint Journal	i
Other Credits.....	i
Table of Contents	i
Exercises	2
Introduction	3
Solutions Overview.....	1
The Problem with Themes in SharePoint.....	1
So... Why Even Bother?	2
What Does SPThemes Bring to the Table?	3
How Does SPThemes Work?.....	5
Adding Sample Data to SharePoint.....	6
What Does SPSampleData Bring to the Table?	7
How Does SPSampleData Work?	9
Mission Statement	9
Rules of Engagement	9
What You Will Need	10
SPThemes DelegateControl	12
The Techniques Used in SPThemes	12
DelegateControl Controls	13
What Are DelegateControls?	13
Creating a DelegateControl	15
Adding the DelegateControl	17
The Case of the Missing DelegateControl	23
I'm Sick and Tired of Waiting—Let's Get On with It!	27
Extensions	30
Creating Extension Methods in SharePoint.....	30
GetCustomProperty	31
SetCustomProperty.....	33
Wrapping Up the Extension Methods	35
Extending the User Information List	36
Setup Feature?	36
Creating the Style Library	38
CustomActions.....	43

What Are Custom Actions?	43
Declarative Custom Actions	44
Programmatic CustomActions.....	46
Custom Application Pages	50
Pages in SharePoint	50
Application Pages.....	50
Site Pages	51
Content Pages	51
Creating Application Pages.....	51
Creating the User Theme Selection Page.....	60
Finishing the Setup Feature	64
SPSampleData.....	67
The Techniques Used in SPSampleData.....	67
Adding the CustomAction.....	68
Adding the SPSampleData Page	70
Testing SPSampleData.....	79

Exercises

 Creating a WSP Solution	15
 Adding the DelegateControl.....	18
 Toying Around with the DelegateControl	22
 SharePoint Development Troubleshooting	23
 Building the SPThemes Delegate Control.....	27
 Reading Custom Properties	30
 Writing Custom Properties.....	33
 Creating More Extension Methods.....	35
 Creating a Setup Feature.....	36
 Creating a Document Library in Code	38
 Creating a CustomAction	44
 Creating a Programmatic Custom Action.....	46
 Adding a Custom Application Page	51
 Creating the User Theme Selection Page	60
 Creating the SPSampleData Solution	68

Introduction

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

Welcome to another issue of *Understanding SharePoint Journal*.

In this issue, I'll talk about two solutions to questions I have received from different readers. These solutions are the SPThemes solution and the SPSampleData solution, both available on CodePlex.

The SPThemes solution came about after someone asked me how to create a custom theme for SharePoint. Sadly, and this may surprise you, there is no way to implement custom themes in SharePoint without either breaking your supportability or creating your own themes solution. I'll explain more in detail throughout this issue.

SPThemes is such a custom themes solution that, contrary to many other examples of implementing custom themes, does not void your warranty. In addition, SPThemes introduces some new features not available in the built-in themes support, such as personal themes for users and site collection–scoped themes.

Chapters 2 to 5 of this issue cover the SPThemes solution.

The second solution, SPSampleData, is a response to a reader who had seen the video I created for issue 5 called “Beginning SharePoint Development.” That video showed how to use a feature to deploy sample data into a SharePoint list, and the question, or questions, related to how one should go about creating these features.

The trick here, as I will also explain throughout the issue, is that SharePoint uses somewhat cumbersome formats for internal data, and authoring large amounts of such data by hand is very tedious and error-prone.

SPSampleData solves this by extracting data from an existing list, allowing you to author the sample data inside SharePoint and then export that data.

Chapter 6 of this issue covers the SPSampleData solution. Yes, just one chapter. I'll show you why later. 😊

So, without further ado, let's get started.



Solutions Overview

Bird's eye on theming

Before we get started exploring the actual solution development, I'll introduce you to the problems we are trying to solve and also why these are problems at all.

I'll also lay some ground rules for how we should solve our problems to ensure that we create a maintainable solution that will keep your IT department happy.

The Problem with Themes in SharePoint

As you may know, SharePoint supports the notion of themes. In short, this is a method of applying a look and feel to a site using a set of custom CSS and image files. Using the themes in SharePoint, site owners can easily change the appearance of a site simply by selecting a different theme, as shown in Figure 1.

ICON KEY	
⌘	Valuable information
✍	Test your knowledge
🖨	Exercise
☠	Caution

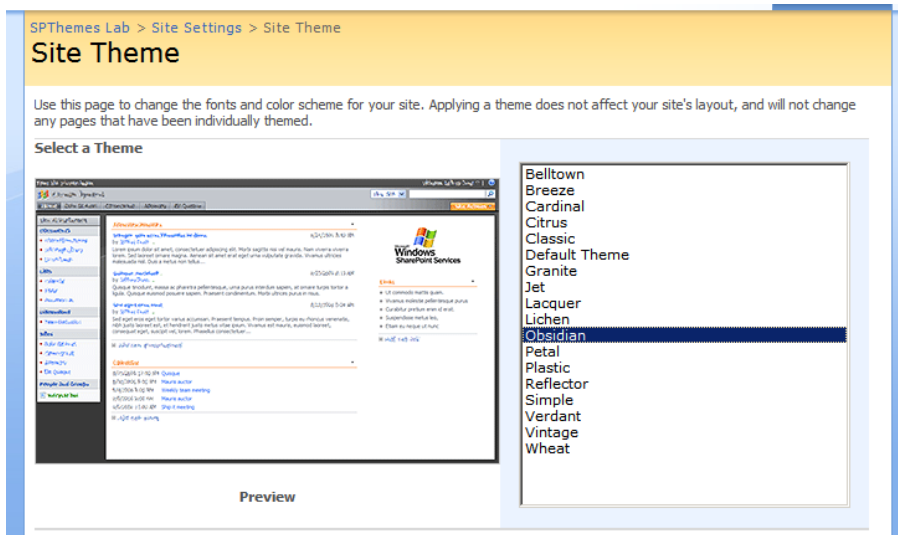


FIGURE 1. THEMES IN SHAREPOINT

However, despite their ease of use, themes in SharePoint have never been very popular with designers. First, themes are difficult to develop, and best practices recommend copying an existing theme and modifying that.

Second, the built-in themes work best with the out-of-the-box designs, and although there are inherent benefits to using the default design, many designers want more flexibility.

Third, adding custom themes requires modifying an out-of-the-box SharePoint file. You may think this is a minor issue, but in fact, if you modify any of these files, you are actually breaking any chance you have of getting support from Microsoft for your solution.

If the lack of support is not bad enough, Microsoft may, at any point and for no reason at all, overwrite your changes through an update such as a service pack.

Finally, themes affect only individual sites, so if you have many sites in a site collection, you also have a lot of theme application ahead of you. Yes, that granularity has benefits too, but when there are no options for applying themes to a site collection at all, the cool factor of themes falls.

So...Why Even Bother?

In defense of themes, they are the only chance you have of affecting the design of application pages. Application pages use their own, farm-wide master page, and modifications to that master page break your supportability. Even if you choose to modify that master page, those changes will affect your entire server, and you may not want that.

In addition, themes become somewhat more important and useful in the next version of SharePoint. SharePoint 2010 supports Office theme files (.thmx files) that allow for greater flexibility and easier ways of adding custom themes.

Note

If you want to learn more about SharePoint 2010, check out the SharePoint 2010 Beta series of USP Journal, available at <http://www.sharepoint2010beta.com/>.

Finally, themes are a low-impact method of customizing the look and feel of SharePoint. Contrary to custom master pages, for example, themes are easily changed, and each site can host multiple themes to allow for easy swapping, even after the solution has been put into production. Themes do not alter the functionality of a site at all.

What Does SPThemes Bring to the Table?

I designed SPThemes to overcome some of the existing weaknesses of the default themes functionality in SharePoint. Some of the benefits include the following:

- Supportable custom themes
- Easier deployment of custom themes
- User-selectable themes
- Site collection–scoped themes

Figure 2 to Figure 4 show the basic functionality of SPThemes.

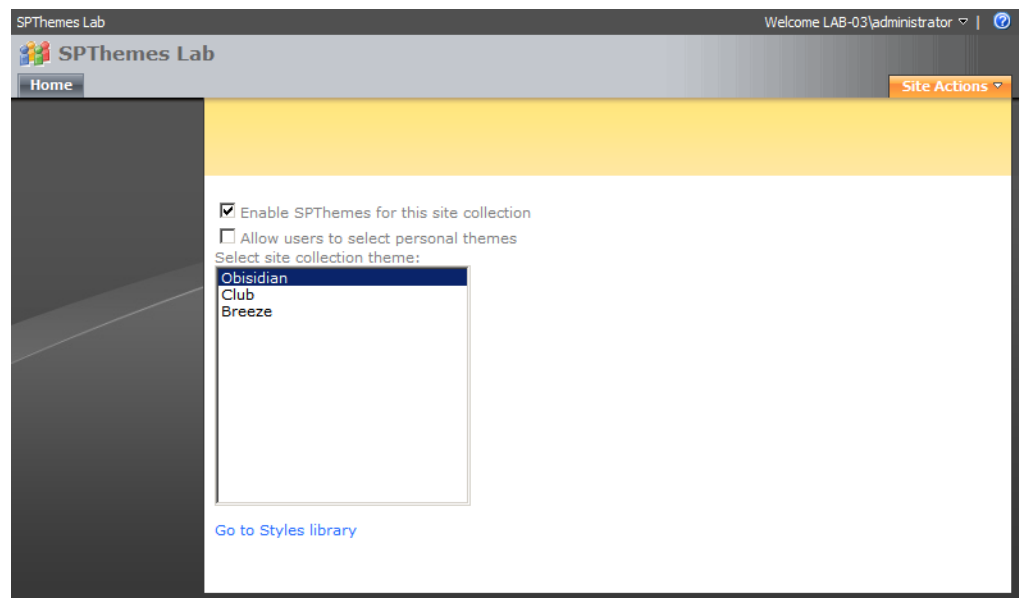


FIGURE 2. SITE COLLECTION THEMES MANAGEMENT

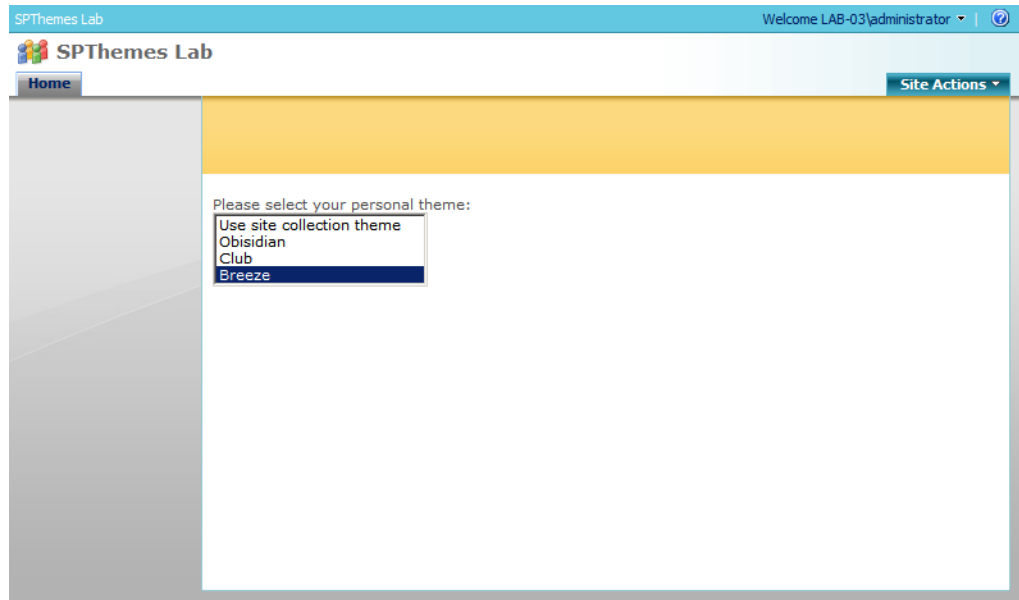


FIGURE 3. PERSONAL THEME SELECTION

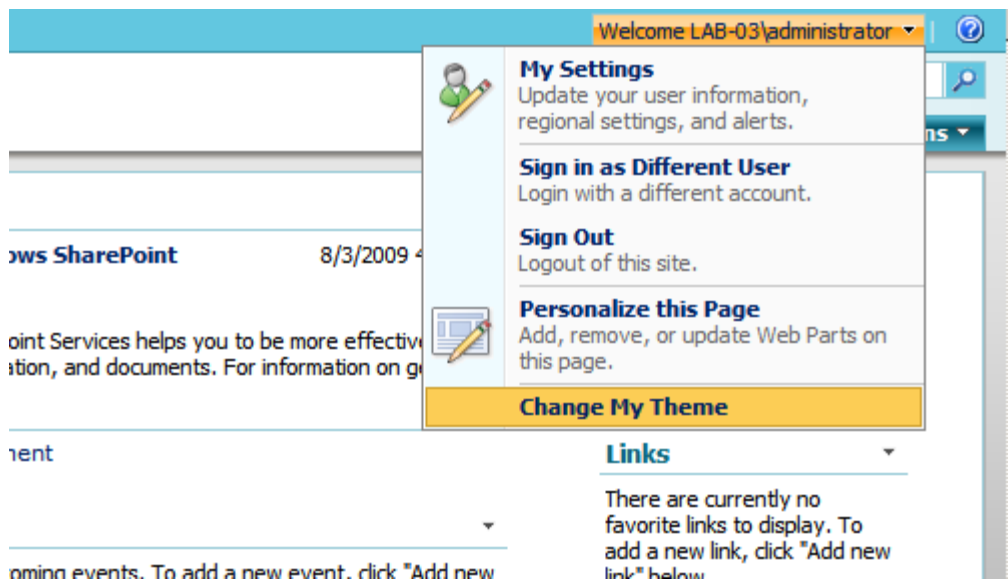


FIGURE 4. PERSONAL ACTIONS

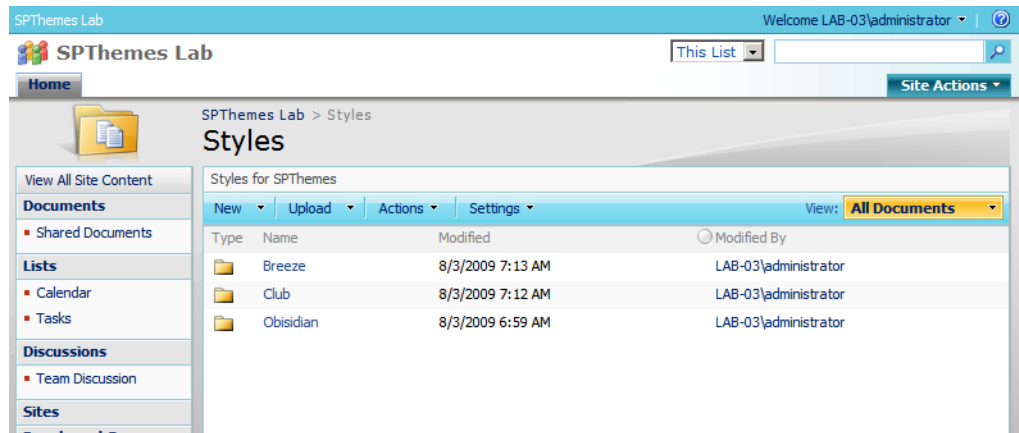


FIGURE 5. STYLE LIBRARY

The custom themes are stored in a separate document library, as shown in Figure 5. In the previous example, I have added three of the themes that ship with SharePoint, but you can easily add new styles to meet your design requirements.

How Does SPThemes Work?

The SPThemes solution utilizes several SharePoint techniques, and through this issue, I will explain these techniques to you:

- A DelegateControl will ensure that the selected theme is added to the page.
- CustomActions, both static and dynamic, will be used to add menu and list options to SharePoint.
- Two custom application pages will provide the user interface for selecting and configuring themes.
- A setup feature will provision the solution, including extending the user information list and creating the style library.

Note

Despite this being a fully working production solution, keep in mind that SPThemes has been developed with SharePoint training in mind. I want to show off some creative uses of SharePoint technology first and solve the problems second.

Let's look at the SPSampleData solution.

Adding Sample Data to SharePoint

As part of the warm-up for the launch of issue 5, I released a video describing a technique to add sample data to a SharePoint list using a feature. Apparently, the idea sparked interest, so I decided to write a solution to make authoring such features simpler.

The problem with creating such features is that the manual authoring of the data itself is complex. The format of the data follows the internal storage format for SharePoint. For example, references to data such as usernames are stored in an ID#;VALUE format. Any HTML or XML content must be converted so that it does not influence the surrounding CAML code.

Where would you want to use such sample data features? Well, the name isn't really doing the technique justice, because you can use the same technique to add production data as part of a solution deployment.

For example, consider setting up a range of project sites. These project sites may contain guidelines for completing the project, perhaps including some suggested tasks, links, or other relevant information.

Consider the following list instance, containing just a single task item based on the default task list in SharePoint:

```
<ListInstance Url="Lists/Tasks" Title="Tasks">
  <Data>
    <Rows>
      <Row>
        <Field Name="Content Type">Task</Field>
        <Field Name="Title">Sample Task</Field>
        <Field Name="Priority">(2) Normal</Field>
        <Field Name="Status">In Progress</Field>
        <Field Name="% Complete">0.15</Field>
        <Field Name="Assigned To">6;#LAB-03\someuser</Field>
        <Field Name="Description">&lt;div&gt;Rich
&lt;strong&gt;text &lt;/strong&gt;&lt;font size=4&gt;must
&lt;/font&gt;&lt;u&gt;be &lt;/u&gt;&lt;font
color="#ff0000"&gt;converted&lt;/font&gt;&lt;/div&gt;</Field>
        <Field Name="Start Date">2009-08-03T00:00:00Z</Field>
      </Row>
    </Rows>
  </Data>
</ListInstance>
```

Although this may not scare you completely out of your sanity, consider having to write this manually for a few hundred items. No wonder this isn't a widely used technique.

However, there should be a simpler way, and SPSSampleData aims to be that way.

What Does SPSSampleData Bring to the Table?

SPSSampleData allows you to extract list items from an existing list into a list instance, ready to be added to a feature. I extracted the ListInstance element shown earlier using SPSSampleData, and it took less than a minute to create the task item, export it, and add it to a feature.

Important!

SPSSampleData extracts from most lists in SharePoint, but since the data will be deployed using a ListInstance element, attachments or document libraries are not supported.

Also, list items containing lookup columns may work, but there are some inherent issues with lookup columns that SPSSampleData will not resolve.

In a typical scenario, you may work in a development or staging environment in which you create a range of lists, either as ListInstance features or through code. For these lists, you may want to include some form of default data. Using SPSSampleData, you can create that data in the development or staging environment and then simply extract that data into a deployable feature.

The technique is based on the fact that SharePoint will not overwrite an existing list instance if another list instance with the same name is activated. As such, as long as the list URL and title remain the same in both the list creation code and the sample data code, the items defined in the sample data feature will be added to the list.

Note also that the data is *added* to the list and does not overwrite any existing content. As such, each time you activate the feature, SharePoint adds a new set of data to the list in addition to any existing content.

Figure 6 and Figure 7 show the very simple interface of SPSSampleData.

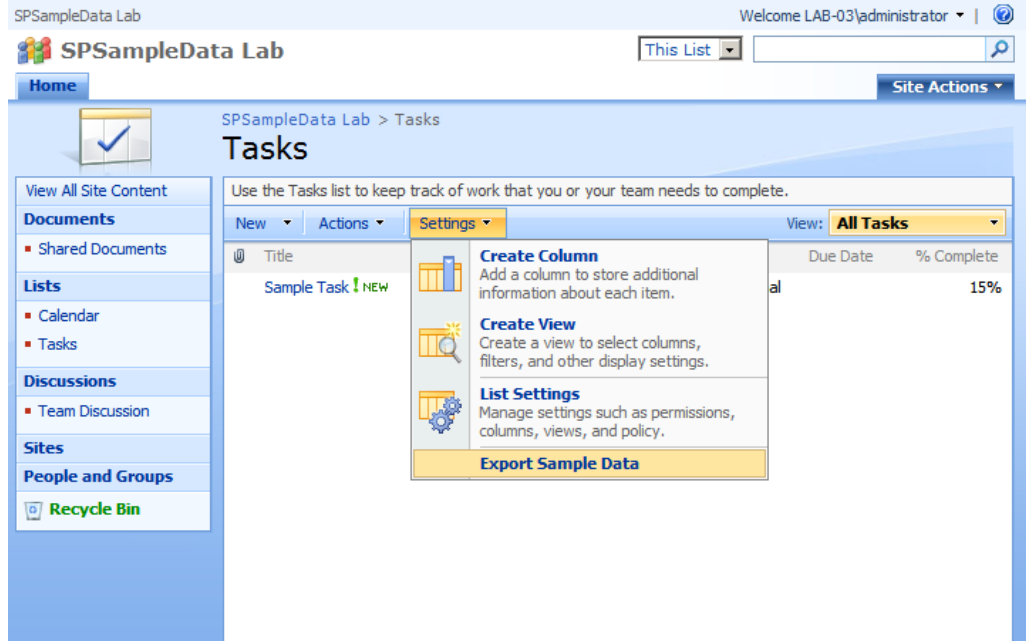


FIGURE 6. EXPORT SAMPLE DATA MENU OPTION

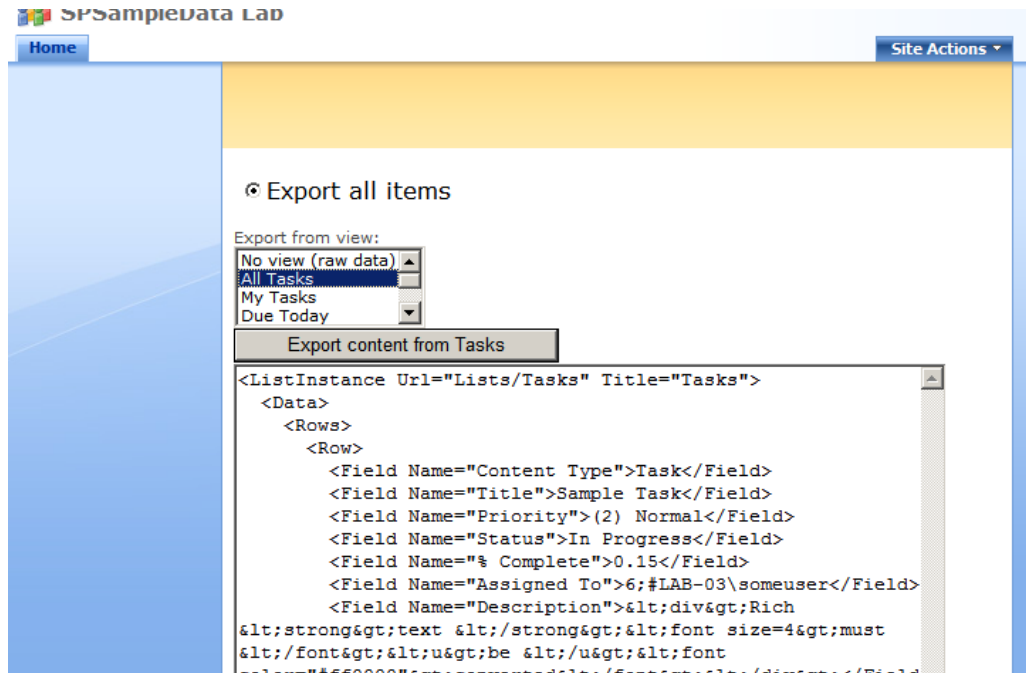


FIGURE 7. EXTRACTING DATA

How Does SPSTSampleData Work?

SPSTSampleData uses a custom application page for the actual extraction. This application page will loop through the items in a list or a view and build the CAML code required to output a ListInstance element. The “magic” happens by using the .NET serialization options that convert classes to XML code.

The resulting ListInstance CAML code can be added directly to an element manifest for deploying data to an existing list as a feature. Alternatively, the ListInstance element can be augmented to become a full list element. In effect, this will allow you to create a feature to deploy both the list and the data included in the list.

Note

SPSTSampleData does not support automatic list instance creation for complete lists, only for data deployment.

SPSTSampleData is a very simple solution, but since the techniques used are closely related to the SPSThemes solution, I thought I’d include it in this issue.

Mission Statement

In this issue, we will re-create the SPSThemes and SPSTSampleData solutions. During the exercises, we will explore the various techniques used to develop these solutions:

- CustomActions, both dynamic and declarative
- Delegated controls
- Custom application page development
- User Information List manipulation

In addition, we’ll use several other SharePoint-related development tasks. I’ll explain which features we will use later in this issue.

Rules of Engagement

We need to define a few rules for our development. If you are a regular reader of *USP Journal*, these rules of engagement should be familiar. Keep in mind that the motivation here is to learn how to build such solutions in SharePoint.

- **We will use only supportable methods and not harm a single Microsoft-provided file.**

If you modify any file that ships with SharePoint, you will not be able to get support for your solution from Microsoft.

- **We will use the least amount of effort possible.**
As per the introduction, we are going to learn how to utilize SharePoint, not necessarily using the “best practice” for any given scenario. Expect to add or improve error handling, for instance.
- **We will utilize SharePoint native techniques whenever possible.**
We might end up taking a step or two back to solve a problem. That is OK; we want to learn SharePoint.

What You Will Need

To follow along and re-create the same solution that is online on CodePlex, you must meet certain requirements.

Of course, you’ll need SharePoint, so grab a copy of that. If you need help installing and setting up a lab environment, check out the *USP Journal* YouTube channel and the free first chapter of the “Beginning SharePoint Development” issue of *USP Journal*.

Setting Up a SharePoint Lab Environment

USP Journal YouTube Channel

<http://www.youtube.com/uspjournal>

First Chapter of “Beginning SharePoint Development”

<http://www.beginningsharepointdevelopment.com/>

WSPBuilder:
<http://www.codeplex.com/wspbuilder>

SharePoint Manager 2007:
<http://www.codeplex.com/spm>

SPTThemes:
<http://www.codeplex.com/SPTThemes>

SPSampleData:
<http://www.codeplex.com/SPSampleData>

You will also need Visual Studio. In these exercises, I am using Visual Studio 2008, and there is a good reason for that, because I am using a particular feature of the C# 3.0 language.

I’ll use WSPBuilder Visual Studio Extensions for this project, so grab a copy of that. I also prefer to have SharePoint Manager 2007 installed. Besides that, your regular setup should work.

I am also going to assume that you are fairly familiar with working with Visual Studio and have a working knowledge of ASP.NET, understand XML, and know how to create .NET classes. If you have basic SharePoint and object model experience, that will be a major plus.

I'll explain everything you need to know beyond that, and, worst case, you can always copy the code if you don't understand all the details.

With that said, I think it is time to start building.

SPTThemes

DelegateControl

Theming as it should be.

The SPTThemes solution utilizes several SharePoint features to work its magic. Through this chapter, I will explain these features, focusing on what goes on behind the scenes. Then, I will walk you through creating the first of the required components, the DelegateControl.

Note

There isn't actually any magic involved. Harry Potter wanna-bes may move along to track 9¾ of King's Cross. This train leaves for SharePoint Guru Status in about two minutes.

This chapter is targeted at new SharePoint developers. If you have developed SharePoint features before, this may seem a bit simple, so feel free to skip to the exercises later in the chapter.

The Techniques Used in SPTThemes

Let's create a list of the topics we need to explore in SPTThemes.

SPTThemes uses a custom DelegateControl control to add a custom style sheet to the current page. So, add "learning DelegateControl development" to the list.

Further, we need to store both the user-selected theme and the site collection theme somewhere. The site settings are stored in the site's RootWeb object, so we need to work with custom properties of SPWeb objects as well. We also need to have some way of disabling or enabling both personal theme selection and custom themes, and these properties will also be stored as custom properties.

SPTthemes stores the user preferences in the user object in the site's User Information List, so we can add working with that list to our topics.

Then, we need to wrap everything up in an easy-to-use package, so we want to create a nice setup feature that handles all the tasks required to get SPTthemes installed and configured. This means we need to work with a feature receiver.

Next, the SPTthemes solution offers simple user interface controls that are deployed using CustomAction elements.

Finally, since we want a nice interface to set all of these preferences, we need to create a couple of application pages. These pages will allow both site administrators and users to set custom themes.

Our topics list thus looks like this:

- DelegateControl controls
- Custom property storage
- Site User Information List management
- Feature receivers
- CustomActions
- Application page development

Let's just start at the top.

DelegateControl Controls

The first task we want to accomplish is to create the DelegateControl control that will handle the CSS output. If you have never worked with DelegateControls before, be prepared to learn an extremely powerful feature of SharePoint.

What Are DelegateControls?

A DelegateControl is a SharePoint control that allows you to insert other ASP.NET controls into a SharePoint page. Think of DelegateControls as plug-ins or, rather, as placeholders for plug-ins.

The default SharePoint master pages and content pages contain several such DelegateControls allowing you to modify the functionality of any SharePoint page without touching the pages at all.

For example, the search box you see on most SharePoint sites is a `DelegateControl`. This is how external solutions such as MOSS or Search Server 2008 can replace the search box in a supportable manner.

Note

Oh, yes, you can replace the search box too. Just read along, and I'll explain how.

The `DelegateControl` system has two components. First, there is the on-page `DelegateControl` itself, which is the placeholder for the content to be added. Second, we have the controls we want to add to the `DelegateControl`.

Each of the `DelegateControls` has a `ControlId` property to identify them. One example of this is the `AdditionalPageHead` control, present in all default SharePoint pages:

```
<SharePoint:DelegateControl runat="server"
ControlId="AdditionalPageHead" AllowMultipleControls="true"/>
```

As you can see here, the `AdditionalPageHead` `DelegateControl` supports multiple controls, which means you can add several controls to the `DelegateControl`, and all the controls will be added. The default behavior is that only one control gets added in case you try to add multiple controls.

The default out-of-the-box layout has several `DelegateControls` added already. For example, in the default.master page, you have the following `DelegateControls`:

ControlId
AdditionalPageHead
GlobalSiteLink0
GlobalSiteLink1
GlobalSiteLink2
SmallSearchInputBox
TopNavigationDataSource
PublishingConsole