

Build a better MOSSetrap... **and the world will beat a path to** **your door**

Starting SP Development, where to go from there, & lessons learned from the field.

James Ferguson

New Horizons of Minnesota

jferguson@nhmn.com

<http://nhmn.com/blogs/PointedMan.aspx>

MCTS, MCPD, MCAD, MCSD, MCDBA, MCSA,
MCSE, MCT, MCITP, etc...

Session Topics

- Where to start...
- The Importance of a Virtual Playground
- Tools of the Trade
- Your First MOSS Add-On:
- How to Modify, Menus, & Application Pages
- Issues with Security
- Debugging
- Deployment Scenarios: Features vs Solutions
- Next Steps

What Do "SharePoint Developers" Do?

The SharePoint2007 Developer
It Depends...



Enterprise Developer

- Designing Portals
- Extending the Search Engine
- Integrating with LOB systems
- Creating Public Web Sites
- Creating Business Forms
- Creating Custom Workflows
- Managing Enterprise Content
- Creating Custom Policies
- Creating Scorecard, BI Reports

Office Client Developer

- Creating Word Template Solutions
- Extending the Ribbon
- Developing Office Add-ins
- Developing VSTO solutions
- Creating Business Forms
- Programmatically Manipulating XML Documents

Web/WSS Developer

- Creating Features and Site Definitions
- Developing Custom Application Page
- Developing Page Templates and Web Parts
- Designing Custom List Types
- Developing Event Handlers
- Developing Custom Workflows
- Creating Solution Packages for Deployment

Where to start...



Virtual Playground Configuration

- Virtual PC 2007 or Virtual Server 2005
- One VPC to rule them all
 - Or the importance of everything on a single box for Development
 - What can you put on a box:
 - Windows Server 2003 R2 or just SP2
 - SQL Server 2005 +SP2
 - IIS 6.0
 - SharePoint Designer 2007 w/SP1
 - Visual Studio 2008 w/SP1
 - MOSS 2007 w/SP1
 - Office 2007 w/SP1

Tools of the Trade: Essential Additions to the Development Environment

- MOSS SDK (vs WSS SDK)
- Application Pool Manager
<http://www.harbar.net/articles/APM.aspx>
- Lutz Roeder's Reflector
<http://reflector.red-gate.com>
- SharePoint Manager 2007
<http://www.codeplex.com/spm>
- Fiddler
<http://www.fiddlertool.com/fiddler/>

- Internet Explorer Developer's Toolbar

Caveats and Cautions

- Always take the road more traveled:
 - Use Built in functionality first
 - Use MS Provided add-on's
 - CodePlex.com add-on's/ MVP provided additions
 - 3rd party additions (cost to build vs. simply buy)
 - Never underestimate SharePoint Designer...
 - If the above doesn't cut the mustard...
then by all means Build it!
 - Virtually all things are possible using the
SharePoint API's and Visual Studio
- But you must have the fortitude to carry them out...

Precursor to Your First MOSS Add-On...

- There are MANY tools available to speed/ease the process of developing with MOSS
- But Caveat Emptor; You are responsible for knowing the coding behind these tools (i.e. what they do and how they work)
- With this in mind let's look at creating a simple MOSS add on the good "old fashioned" way
- Later we'll talk about some additions and when/why to use them...

Your First MOSS Add-On In 5 easy Steps

Step 1: Access API using Console Application

Step 2: Wrap this functionality in an ASP.Net Page
(if necessary)

Step 3: Create the Feature to deploy it.

Step 4: Deploy the Feature

Step 5: Activate Feature and Enjoy an Ice Cold
Beverage of your Choosing as a reward!

Development Gotcha's

Watch Out: Inconsistent Terminology

New Term	Old Term	WSS Object Model
Site Collection	Site	SPSite
Site	Web	SPWeb
Top-Level Site	Root Web	SPSite.RootWeb

Step 1: Access API via Console Application

```
using System;
using Microsoft.SharePoint;
namespace FirstConsoleApp {
    class Program {
        static void Main() {
            string sitePath = "http://BenchMarkLearning.com";
            // enter object model via site collection.
            SPSite siteCollection = new SPSite(sitePath);
            // grab reference to the top-level site.
            SPWeb site = siteCollection.RootWeb;
            // enumerate through lists in this site
            foreach (SPList list in site.Lists) {
                if (list.Hidden == false) {
                    Console.WriteLine(list.Title);
                }
            }
            // clean up by calling Dispose.
            site.Dispose();
            siteCollection.Dispose();
        }
    }
}
```

Development Gotcha's : Memory Leaks 1/2

- Watch Out for Memory Leaks...
 - A prime source for long term problems
(not usually immediately seen or felt though...)
- Anytime you create an **instance** of an object you should dispose of it if possible (as most SP objects require disposal)
 - (e.g. SPSite, SPWeb, etc.)

Development Gotcha's : Memory Leaks 2/2

- But Don't Dispose of absolutely everything with a Dispose method in it...
- Anytime you **Reference** existing objects **DO NOT** dispose of these objects
(as disposing of these causes untoward behavior)

```
SPWeb site = (SPWeb)SPFeatureReceiverProperties.Feature.Parent;
//(In Feature Receiver)
```

-OR-

```
SPWebApplication WAppCurrent = SPContext.Current.Site.WebApplication;
SPSiteCollection SC = WAppCurrent.Sites;
foreach (SPSite oSite in SC) {
    Response.Write(oSite.Url + "<BR>");
    oSite.Dispose();
} //In ASP.NET Code
```

Step 2: Wrap functionality in ASP.Net Page

- Create a Class Library DLL Project: SPCamp2
- Add References for System.Web and Microsoft.SharePoint (a.k.a. WindowsSharePointServices)
- Add Folder to root Directory called **TEMPLATE**
 - Embed **Layouts** directory inside TEMPLATE
 - Add new **ApplicationPage1.aspx** file here
 - Note: add a new text file and give it the name above.
 - Add Code File for ASPX page to Application root directory **ApplicationPage1.cs**
 - Note: add a new text file and give it the name above

Step 2: Cont.

- Create a Strong Name for the Project (using Project Properties > Signing)
- Build application and then use .Net Reflector to gain 4 part name for dll and also for Microsoft.Sharepoint.dll found in
C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\ISAPI\
- Add these names to the .aspx page at top of file in
`<%@ Assembly Name=" " />`

Step 2 Cont

- Add the @Page directive

```
<%@ Page Language="C#"
    MasterPageFile="~/_layouts/application.master"
    Inherits="SharePointCamp2.ApplicationPage1"%>
```

- Override the appropriate ContentPlaceholders:
 - Heather Solomon's Blog has a list of the 33 basic ones
 - Common Ones to override are:
PlaceHolderMain, PlaceHolderPageTitle,
PlaceHolderPageTitleInTitleArea

Step 2 Cont.

- Add code to the Code Behind File
 - Add the appropriate Using Statements:
 - Common Ones to add
 - Using System
 - using System.Web;
 - using System.Web.UI;
 - using System.Web.UI.WebControls;
 - using Microsoft.SharePoint;
 - using Microsoft.SharePoint.WebControls;
 - Have Class inherit from: `LayoutsPageBase`
`public class ApplicationPage1 : LayoutsPageBase {`
 - Add a variable for each control in aspx file
 - In our case : `protected Label lblLists;`

Step 2 Finished

- Override the appropriate Events as necessary:
 - In our case PageLoad event

```
protected override void OnLoad(EventArgs e) {  
    // get current site and web  
    SPSite siteCollection = this.Site; //No Disposal (Current Context)  
    SPWeb site = siteCollection.RootWeb; //Disposal Needed  
    // program against controls on .aspx page  
    foreach (SPList list in site.Lists) {  
        if (list.Hidden == false) {  
            lblLists.Text += list.Title + ", ";  
        }  
    }  
    site.Dispose();  
}
```

Step 3: Create the Feature to deploy it.

- In the Class Library Project:
- Inside the existing **TEMPLATE** folder
 - Embed **FEATURES** directory inside TEMPLATE
 - Embed Directory with **Projects Name** inside FEATURES
- Add a new XML file and name it feature.xml
- Add a new XML file and name it elements.xml
- Add the appropriate code to each file... but how...
no IntelliSense is there to guide us...

Development Gotchas: xml IntelliSense

- Adding Appropriate IntelliSense:

- Navigate to

- [Visual Studio 2008 Install Director]\XML\Schemas

- Copy catalog.xml rename it SharePointCatalog.xml

- Remove all nodes except for the <SchemaCatalog> and one <Schema> Node and modify it as follows:

```
<Schema href="C:/Program Files/Common Files/Microsoft Shared/web  
server extensions/12/TEMPLATE/XML/wss.xsd"
```

```
targetNamespace="http://schemas.microsoft.com/sharepoint"/>
```

- Close and re-open Visual Studio (if necessary)

- In element.xml or feature.xml add the following element: <Feature xmlns=""> or <Element xmlns=""> and pick the sharepoint one.

Step 3 cont.

- Inside the Feature.xml file:

```
<?xml version="1.0" encoding="utf-8" ?>  
<Feature  
  xmlns="http://schemas.microsoft.com/sharepoint/"  
  Id="{48DB7A95-6335-4ae7-8B73-F81234940E5A}"  
  Title="Sample Feature: Custom App Page"  
  Description="Created for SharePointCamp2"  
  Version="1.0.0.0"  
  Scope="Web"  
  Hidden="FALSE"  
  >  
  <ElementManifests>  
    <ElementManifest Location="elements.xml" />  
  </ElementManifests>  
</Feature>
```

Step 3 Cont.

- Inside the elements.xml file:

```
<?xml version="1.0" encoding="utf-8" ?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <!-- Add Command to Site Actions Dropdown -->
  <!-- + Actions http://msdn.microsoft.com/en-us/library/ms473643.aspx -->
  <CustomAction Id=" SharePointCamp2ApplicationPage"
    GroupId="SiteActions"
    Location="Microsoft.SharePoint.StandardMenu"
    Sequence="2000"
    Title=" SharePointCamp2 Application Page"
    Description="Welcome to your site actions Menu">
    <UrlAction Url="~site/_layouts/ SharePointCamp2/ApplicationPage1.aspx" />
  </CustomAction>
</Elements>
```

STSADM.EXE Command-line Utility

- Useful for running administrative commands
 - Can be used interactively from command line
 - Commands can be scripted using batch files
- Add stsadm.exe location to Path statement (Demo)
C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\BIN

```
C:\WINDOWS\system32\cmd.exe
```

```
C:\>stsadm -o installfeature
```

```
Please use either the -name or the -filename argument to select a feature.
```

```
stsadm.exe -o installfeature
```

```
<-filename <relative path to Feature.xml from system feature director
```

```
y> :
```

```
-name <feature folder>>
```

```
[-force]
```

Step 4: Deploy the Feature

- Manually copy files into place:
 - Copy Template Directory over “12 Hive”/Template
- Add the SharePointCamp2.dll file to the GAC
 - Drag and drop the dll located in the project dir/bin location into the C:/Windows/Assembly directory
 - Note: you must drag-and-drop as copy and paste will NOT work here
- Install the Feature
 - From a cmd prompt type:
Stsadm –o installfeature –name SharePointCamp2
Press enter and wait for success message.

Step 5: Activate Feature & Enjoy Ice Cold Beverage of Choice as a reward!

- Activate the feature and examine the results:
 - Navigate to a site you wish to activate this feature on:
 - Example: <http://benchmarklearning.com>
 - **Site Actions Menu > Site Settings > Modify All Settings**
 - Select **Site Features** under Site Administration column
 - **Activate** the **Sample Feature: Custom App Page** feature
 - Take it for a test drive....

If it succeeds great... If not then consider the following...

[Go back to site](#)

Viewing Errors in SP **Error**

Contact your system administrator if this problem persists.
[Troubleshoot issues with Windows SharePoint Services.](#)

- On the Web server, navigate to the site directory:
C:\inetpub\wwwroot\wss\VirtualDirectories\[directory for WebApplication]
- Open Web.config in Notepad. (Backup first... just in case)
- Search for "**CallStack**". Set CallStack="**true**".
<SafeMode MaxControls="200" **CallStack="true"**
DirectFileDependencies="10" TotalFileDependencies="50"
AllowPageLevelTrace="false">
- Search for "**CustomErrors**". Set Mode="**Off**".
<customErrors **mode="Off"** />
- Save and close the file.
- CAUTION: Any change to a web.config file will cause your Web Application to RESET

Development Gotcha's: Errors in SP Cont.

- Use the KISS Method:
 - Create the simplest action possible and then add complexity a step at a time.
- If at first you don't succeed...
 - Fix your code
 - Re-Copy your Template Directory to the 12 hive
 - Uninstall the GAC dll and then drag-and-drop the update
 - IISRESET before you retest after changes to dll

Development Gotcha's: Security...

-or-

What we have here is a failure to remunerate...

- If you try to access information in SharePoint and you receive an access denied security error... SharePoint cannot retrieve this information generically for everyone... so we need to run with elevated privileges to gain access.
 - To do this We would need to modify our ApplicationPage1.cs file as Follows:

Development Gotcha's: Security Cont.

Plus Skills, More Knowledge

```
protected override void OnLoad(EventArgs e)    {
    // get current site and web
    SPSite siteCollection = this.Site;
    SPWeb site=siteCollection.RootWeb; //Disposal needed
//run just this code block with elevated privileges
    SPSecurity.RunWithElevatedPrivileges(delegate()    {
        using (SPSite ElevatedsiteCollection = new SPSite(siteCollection.ID)) {
            using (SPWeb ElevatedSite =
ElevatedsiteCollection.OpenWeb(site.ID)) {
                foreach (SPList list in ElevatedSite.Lists)    {
                    if (list.Hidden == false)    {
                        lblLists.Text += list.Title + ", ";
                    }
                }
            }
        }
    });
Site.Dispose();
```

Debugging your MOSS Application

- In addition to the Changes needed in the Application web.config file for viewing Errors from earlier:
 - Modify the web.config as follows
 - Change `<compilation batch="false" debug="false">` to `<compilation batch="false" debug="true">`
 - Save changes

Debugging your MOSS Application Cont.

- Next you must configure Visual Studio to Allow this type of debugging:
 - In Visual Studio:
Tools menu > Options > Debugging >
 - Remove the Check from
Enable Just My Code (Managed only)
 - Click OK

Debugging Cont.

- Next Attach to the correct w3wp process:
 - Go to Debug -> Attach to Process...
 - Check the Show processes from all users and Show processes in all sessions checkbox.
 - Scroll down to the w3wp process and click Attach.
- Important: although you can attach to more than one w3wp process at a time **ONLY THE FIRST ONE** attached will be debugged.
 - So you need a way to see if the correct w3wp has been attached...

Debugging Cont.

- How to tell if you've got the correct w3wp:
 - open the Modules window by pressing CTRL+ALT+U
 - Look for your dll...
 - Repeat this process (detach and attach to w3wp) until you locate the correct w3wp
- Verify that the symbols are loaded...
- Check the Symbol Status column:
 - Symbols loaded; great you are ready to set breakpoints
 - Cannot find or open the PDB file; you've got work to do

Debugging Almost Finished...

- If you see: Cannot find or open the PDB file
 - Right click and set the path to the projects bin/debug directory and click OK
 - You should now see Symbols Loaded...
 - Set breakpoints and try the page.
 - If you STILL can't debug...

Debugging Finished

Print Skills. Harvest Knowledge.

- Right Click on the Breakpoint & Select Location...

The screenshot shows the Visual Studio IDE with a File Breakpoint dialog box open. The dialog box is titled "File Breakpoint" and contains the following fields:

- File:
- Line:
- Character:
- Allow the source code to be different from the original version

The dialog box also has an "OK" button and a "Cancel" button. The "OK" button is circled in red. A red arrow points from the "OK" button to the breakpoint icon in the code editor.

The code editor shows the following code:

```
13         protected Label lblLists;  
14  
15         protected override void OnLoad(EventArgs e)
```

Features vs. Solutions

- Features are great for testing and development... but not so much for enterprise deployment...
- Why: Manual deployment to EACH AND EVERY server, Manual updates etc... and If I add an additional Web Server later... I must remember to re-deploy these files to that server as well.
- Solutions (wsp) meant for enterprise deployment and maintainability

What are Solutions?

- Evolution of Web Part Packages from WSS 2.0
 - Solution Package is a CAB file with .wsp extension
 - Solution Package contains a manifest
 - Solution Package contains files required on Web server
 - What can be deployed via a Solution Package
 - Feature definitions
 - Application Pages
 - Assembly DLLs
- And much more...

Solutions Cont.

- WSS Deployment done with Solution Packages
 - Solution Package is CAB file with .wsp extension
 - Created using DDF file and MAKECAB.EXE
 - Deployed using STSADM.EXE or WSS Central Admin

	CodeCampV.dll	Application Extension	
	manifest.xml	XML Document	
	elements.xml	XML Document	CodeCampV\ [redacted]
	feature.xml	XML Document	CodeCampV\ [redacted]
	ApplicationPage1.aspx	ASP.NET Server Page	LAYOUTS\ CodeCampV\ [redacted]
		CodeCampV.wsp	

Creating a Solution

- In the project root directory add a **Solution** folder
- Inside this folder add:
 - A **manifest.xml** file
 - A **cab.ddf** file (e.g. a text file)

Manifest.xml file:

```
<?xml version="1.0" encoding="utf-8" ?>
<Solution SolutionId="{81EFF3D8-159F-4936-9250-364D34ED4F2F}"
          xmlns="http://schemas.microsoft.com/sharepoint/">
  <FeatureManifests>
    <FeatureManifest Location=" SharePointCamp2\feature.xml" />
  </FeatureManifests>

  <TemplateFiles>
    <TemplateFile Location="LAYOUTS\
SharePointCamp2\ApplicationPage1.aspx"/>
  </TemplateFiles>

  <Assemblies>
    <Assembly Location=" SharePointCamp2.dll"
              DeploymentTarget="GlobalAssemblyCache" />
  </Assemblies>
</Solution>
```

Cab.ddf file:

```

;
.OPTION EXPLICIT ; Generate errors
.Set CabinetNameTemplate= SharePointCamp2.wsp
.set DiskDirectoryTemplate=CDROM ; All cabinets go in a single directory
.Set CompressionType=MSZIP;** All files are compressed in cabinet files
.Set UniqueFiles="ON"
.Set Cabinet=on
.Set DiskDirectory1=Package

Solution\Manifest.xml Manifest.xml
TEMPLATE\FEATURES\SharePointCamp2\feature.xml SharePointCamp2\feature
TEMPLATE\FEATURES\SharePointCamp2\elements.xml SharePointCamp2\eler
TEMPLATE\LAYOUTS\SharePointCamp2\ApplicationPage1.aspx LAYOUTS\ SharePointCamp2\ApplicationPage1.

bin\Debug\SharePointCamp2.dll SharePointCamp2.dll
;*** <the end>

```

Create the wsp and deploy!

- From a cmd prompt in the project directory run:

```
makecab /f Solution\cab.ddf
```

```
cd package
```

```
STSADM -o addsolution -filename  
SharePointCamp2.wsp
```

```
STSADM -o execadmsvcjobs
```

```
STSADM -o deploysolution -name  
SharePointCamp2.wsp -immediate -  
allowGacDeployment -force
```

```
STSADM -o execadmsvcjobs
```

Alternate deployment Automation...

- In the project root directory create a Package dir.
- Inside dir Create a DeployProjectSolution.cmd file
- Configure this file to do the items on the prior slide
- Configure Project Properties...> Build Events > Post-build event command line: as follows
 - cd \$(ProjectDir)\Package
 - DeploySolutionPackage.cmd

Summary

- Where to start...
- The Importance of a Virtual Playground
- Tools of the Trade
- Your First MOSS Add-On:
- How to Modify Menus, & Add Application Pages
- Issues with Security
- Debugging
- Deployment Scenarios: Features vs Solutions
- Next Steps

Next Steps – Tools of the Trade

- Caution: As stated earlier you MUST understand these information that is subsumed by these processes for when they fall short...
- STSDEV
 - <http://www.codeplex.com/stsdev>
- WSPBuilder
 - <http://www.codeplex.com/wspbuilder>
- Spence Harbar's STSADM Add Solution Registry Hack
- SharePoint Solution Installer
 - <http://www.codeplex.com/sharepointinstaller>
- Codeplex SharePoint 2007 Features project
 - <http://www.codeplex.com/features/>
- Microsoft's Visual Studio Extensions for WSS
- Microsoft SharePoint Products and Technologies Team Blog
<http://blogs.msdn.com/sharepoint/>

Next Steps – Resources 2 of 3

- SharePoint Designer Blog
<http://blogs.msdn.com/sharepointdesigner/default.aspx>
- WSS Demo site: (a plethora of SP goodies) <http://www.wssdemo.com>

Next Steps – Resources 3 of 3

- Internet Explorer Toolbar for Developers
<http://www.microsoft.com/downloads/thankyou.aspx?familyId=e59c3964-672d-4511-bb3e-2d5e1db91038&displayLang=en>
- Liam Cleary's Blog entry on Http Modules
<http://www.helloitsliam.com/archive/2007/05/09/moss2007-quote-httpmodule-quote-investigation.aspx>
- SPD Docs Blog: <http://blogs.msdn.com/sharepointdeveloperdocs/>
- WSS Demo site: (a plethora of SP goodies) <http://www.wssdemo.com>
- MN SharePoint Users Group <http://www.sharepointmn.com/default.aspx>

QUESTIONS

&

ANSWERS

Thank You

James Ferguson

jferguson@nhmn.com

<http://nhmn.com/blogs/PointedMan.aspx>