

SHAREPOINT 1, 2, 3!

PART OF THE TRAINING 1, 2, 3! SERIES



LEVEL 300 COURSE MATERIALS HANDS-ON LAB LESSON GUIDE

SPONSORED BY



CONTENT PRODUCED BY
THE ATLANTA MICROSOFT PROFESSIONALS USER GROUP



**TRAINING MATERIALS
PRODUCED BY**

THE ATLANTA MICROSOFT PROFESSIONALS
[HTTP://WWW.ATLANTAMSPROS.COM/](http://www.atlantamsp.com/)

DAN ATTIS, SENIOR DEVELOPER
MATT RANLETT, SENIOR SYSTEMS ANALYST
KEITH ROME, CONSULTANT
BRENDON SCHWARTZ, COLLABORATION EVANGELIST

© 2005 ATLANTA MICROSOFT PROFESSIONALS

TABLE OF CONTENTS

TABLE OF CONTENTS 3

HANDS-ON LAB SETUP 4

 WARM-UP EXERCISE: GETTING STARTED WITH HANDS ON LAB 300 5

LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY 6

 EXERCISE 1: EXPLORING THE SHAREPOINT OBJECT MODEL 7

 EXERCISE 2: USING THE SHAREPOINT OBJECT MODEL FROM EXTERNAL APPLICATIONS 10

 EXERCISE 3: IMPLEMENTING DOCUMENT LIBRARY EVENT HANDLERS 12

LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS 14

 EXERCISE 1: CREATE A SHAREPOINT WORKSPACE FROM WITHIN EXCEL USING VISUAL BASIC FOR APPLICATIONS 15

 EXERCISE 2: SAVE AN EXCEL DOCUMENT TO AN EXISTING WORKSPACE WITH VBA 17

 EXERCISE 3: CREATE A SHARED LIST IN SHAREPOINT USING EXCEL AND VBA 19

 EXERCISE 4: USE A SHAREPOINT LIST IN AN EXCEL DOCUMENT WITH VBA 21

 EXERCISE 5: RESPOND TO SHAREPOINT DOCUMENT UPDATES IN EXCEL USING VBA 24

LESSON 3: INFOPATH FORMS AND WORKFLOW LITE 26

 EXERCISE 1: INFOPATH FORMS AND FORMS LIBRARIES 27

 EXERCISE 2: INSTALLING AND CONFIGURING WORKFLOW LITE 33

LESSON 4: ADVANCED WEB PART DEVELOPMENT 38

 EXERCISE 1: CREATING A PROVIDER WEB PART 39

 EXERCISE 2: CREATING A CONSUMER WEB PART 44

 EXERCISE 3: CREATING A SECOND CONSUMER WEB PART 48

LESSON 5: SHAREPOINT WEB SERVICES 52

 EXERCISE 1: USING SHAREPOINT WEB SERVICES 53



HANDS-ON LAB SETUP

Objectives After completing this lab, you will understand the steps required to achieve the goals of this Hands on Lab

1. Setting up a SharePoint server for the remaining lessons covered in this Hands-On Lab.

Setup The lab machine requires the following software be pre-installed before beginning this lab:

- Windows Server 2003 + Service Pack 1
 - Install IIS and create an application server.
 - Remove FrontPage 2000 server extensions
 - Windows SharePoint Services + Service Pack 1
- SQL Server 2000 + Service Pack 4
- Reporting Services + Service Pack 2
- Microsoft Office 2003 Professional

Estimated time to complete this lab: 10 minutes



HANDS-ON LAB SETUP

WARM-UP EXERCISE: GETTING STARTED WITH HANDS ON LAB 300

Scenario

All of the level 300 labs will use the same team portal off of the main SharePoint site. Let's create that portal before we begin any of the level 300 lessons.

Tasks	Detailed Steps
1. Create a new portal	<ol style="list-style-type: none"> 1. Open top level site (http://localhost/) 2. Click Create in top navigation 3. Click Sites and Workspaces 4. Enter HOL300 in the Title, Description and URL textboxes 5. Select same permissions as parent and click Create 6. Select Blank Site as your template and click OK
2. Add site as a trusted site.	<ol style="list-style-type: none"> 1. Open Internet Explorer and browse to http://localhost 2. Click Tools → Internet Options 3. Select the Security Tab 4. Select Trusted Sites in site options 5. Click the Sites button 6. Uncheck the SSL check box 7. Make sure http://localhost is in the site text box 8. Click Add, then click Close, and finally OK
3. Add a database connection string for AdventureWorks 2000	<ol style="list-style-type: none"> 1. Open a Windows Explorer window by double-clicking on My Computer 2. Navigate to C:\inetpub\wwwroot 3. Make a backup of the web.config file by selecting the file and clicking Copy then clicking Paste. A new file, Copy of web.config will appear. 4. Right-click the file and select Open with... Pick Notepad from the list and click OK. 5. Search for the <appSettings> section in the file 6. If the section does not exist yet, add it immediately following the </configSections> closing element. <configSections> must always be the first section in a config file. 7. Add the following values between the <appSettings> and </appSettings> tags: <pre><add key="DB:AdventureWorks" value="Server=(local);Database=AdventureWorks2000; Trusted_Connection=True"/></pre> 8. Save and close the config file
4. Run IISReset to reload the web.config file	<ol style="list-style-type: none"> 1. Click Start → Run 2. Type "iisreset" and click OK 3. A command window will open and the IIS Web Service will restart



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

Objectives After completing this lab, you will understand the steps required to achieve the goals of this Hands on Lab

- Understand how to add the SharePoint Object Model to Visual Studio .Net projects
- Learn how the SharePoint Object Model allows remote interaction with SharePoint

Setup The lab machine requires the following software be pre-installed before beginning this lab:

- Windows Server 2003 + Service Pack 1
 - Install IIS and create an application server.
 - Remove FrontPage 2000 server extensions
 - Windows SharePoint Services + Service Pack 1
- SQL Server 2000 + Service Pack 4
- Reporting Services + Service Pack 2
- Microsoft Office 2003 Professional
- Visual Studio

Estimated time to complete this lab: 120 minutes



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

EXERCISE 1: EXPLORING THE SHAREPOINT OBJECT MODEL

Scenario

It is difficult to understand how a Windows SharePoint Server has its team sites and workspaces laid out. To help us understand what we are working with, we need a web part that shows us exactly what sites have been built off of the root.

Tasks	Detailed Steps
1. Open Visual Studio	1. Start → All Programs → Microsoft Visual Studio .Net 2003 → Microsoft Visual Studio .Net 2003
2. Create a new empty solution	1. In Visual Studio, click File → New → Blank Solution 2. Name the solution " Sharepoint123 " 3. Save the solution under " C:\Projects " (Use the Browse button to easily find this directory)
3. Add a new Visual Basic project using the "Web Part Library" wizard.	1. Right click the solution in the Solution Explorer and select Add → New Project 2. Select Visual Basic Projects → Web Part Library 3. Name the project " Level300LabVB " 4. Save the project under " C:\Projects\Sharepoint123 "
4. Create a Code Signing Key Pair.	1. Run a Visual Studio Command Prompt by clicking Start → All Programs → Microsoft Visual Studio .Net 2003 → Visual Studio .Net Tools → Visual Studio .Net Command Prompt 2. Change Directory to C:\Projects\Sharepoint123 by typing " cd \projects\sharepoint123\Level300LabVB " 3. Execute the command " sn -k KeyPair.snk " to create the keys 4. Leave the command window open for now 5. Open the AssemblyInfo.vb file in Visual Studio 6. Add the following AssemblyKeyFile attribute after the AssemblyVersion attribute <Assembly: AssemblyKeyFile("C:\Projects\Sharepoint123\Level300LabVB\KeyPair.snk")> 7. Save the file
5. Rename the default files	1. In the Solution Explorer, Rename WebPart1.vb to Webs.vb by right clicking on the file and click Rename 2. In the Solution Explorer, Rename WebPart1.dwp to Webs.dwp by right clicking on the file and click Rename 3. In the Properties window, ensure its build action is set to Content. 4. Edit the Manifest.xml file, changing the DwpFile element's FileName attribute to " Webs.dwp " 5. Save the file
6. Update the default	1. Open the Webs.vb file by double clicking on it in the Solution



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

Web Part class file	<p>Explorer</p> <ol style="list-style-type: none"> Replace all instances of "WebPart1" with "Webs" by using the Edit → Find and Replace → Replace (CTRL+H) dialog. There should be 4 instances Delete all contents of the Webs class except for the RenderWebPart method Delete the content of the RenderWebPart method (leave only the empty shell) Remove the DefaultProperty attribute from the Webs class Save the file
7. Add imports statements to the class file.	<ol style="list-style-type: none"> Paste the contents of 300.1.1.Imports.txt into the top of the Webs class file <p><i>Note: this adds an import statement for the SharePoint.Administration namespace</i></p>
8. Add some helper methods to the web part	<ol style="list-style-type: none"> Paste the contents of 300.1.1.LoadSiteCollections.txt into the Webs class Paste the contents of 300.1.1.LoadChildSites.txt into the Webs class <p><i>Note: these methods recursively load all the sites in your portal into an easy to read list</i></p>
9. Implement the Rendering logic	<ol style="list-style-type: none"> Inside the RenderWebPart method, paste the contents of 300.1.1.RenderWebPart.txt Save the file <p><i>Note: This snippet adds the display logic for your web part</i></p>
10. Configure the Project Build Options	<ol style="list-style-type: none"> Open the Project Properties dialog for the Level300LabVB project Navigate to the Configuration Properties → Build page Change the Output Path configuration value to "C:\Inetpub\wwwroot\bin" Click OK to apply the changes
11. Test the project build	<ol style="list-style-type: none"> Select Build → Build Level300LabVB to initiate a project build Verify that the build was successful by viewing the Build Output window Using Windows Explorer, verify that the file was created in the C:\inetpub\wwwroot\bin folder
12. Extract the public key token	<ol style="list-style-type: none"> Return to the open Visual Studio .Net Command Prompt window Execute the command "sn -T \inetpub\wwwroot\bin\Level300LabVB.dll" Keep the public key token value available for steps 13 and 14 Example output – "Public key token is 98f9879e6f154af2" Do not close the command window as we will need this number in the next step
13. Update the dwp file	<ol style="list-style-type: none"> Open the Webs.dwp file



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

	<ol style="list-style-type: none"> Change the Title to "Webs" Change the Description to "Enumeration of Webs" Change the TypeName to "Level300LabVB.Webs" Change the Assembly to "Level300LabVB, Version=1.0.0.0, Culture=Neutral, PublicKeyToken=12dc9ace77001cbd" Substitute the token extracted in the previous step in the Assembly element Save the file
14. Mark the web part as a Safe Control	<ol style="list-style-type: none"> Open the web.config file in C:\inetpub\wwwroot Add a SafeControl element with the following attributes (use the snippet <i>300.1.1.SafeControl.txt</i>): <ul style="list-style-type: none"> Assembly="Level300LabVB, Version=1.0.0.0, Culture=Neutral, PublicKeyToken=12dc9ace77001cbd" Namespace="Level300LabVB" TypeName="*" Safe="True" Substitute the token extracted in a previous step for the PublicKeyToken attribute Save the file but leave the file open
15. Modify Trust Level	<ol style="list-style-type: none"> In web.config, replace the <code><trust level="WSS_Medium" originUrl="" /></code> with <code><trust level="Full" originUrl="" /></code> Save and close the file Perform an iisreset by clicking Start → Run, type "iisreset", and press Enter or click OK
16. Import the Web Part into SharePoint	<ol style="list-style-type: none"> Launch a browser window, and point it to http://localhost/HOL300 Click the "Modify Shared Page" menu at the top right Select the "Add Web Parts" sub menu Select "Import" Use the Browse button to find and select the file "C:\Projects\Sharepoint123\Level300LabVB\Webs.dwp" Click the Upload button There should now be a "Webs" web part listed under "Uploaded Web Parts"
17. Using the Web Part	<ol style="list-style-type: none"> Click and drag the "Webs" icon into any content area of the portal site The finished Web Part should be displayed in the target drop zone



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

EXERCISE 2: USING THE SHAREPOINT OBJECT MODEL FROM EXTERNAL APPLICATIONS

Scenario

To help improve our understanding of the SharePoint Object Model and how it can help us with our work, let's build a Winforms application that will expose several functions to us and demonstrate the power of the SharePoint Object Model.

Tasks	Detailed Steps
1. Add a Windows Application to your Solution	<ol style="list-style-type: none"> 1. Right click on your solution and select Add→New Project 2. Select “Visual Basic Projects, Windows Application” 3. Type “SPObjModelDemo” in the Name field 4. Click “OK”
2. Delete the default form and add a new one	<ol style="list-style-type: none"> 1. Delete Form1.vb by right clicking on the file in the Solution Explorer and selecting Delete. 2. Right click on your project and select Add→Add Windows Form 3. Type “frmMain” in the Name field 4. Click “Open” 5. Switch to the code view by right clicking the form designer and selecting View Code. 6. Replace all the contents of frmMain.vb with the contents of the included snippet code in 300.1.2.frmMain.vb.txt 7. Save the file.
3. Set up Startup object	<ol style="list-style-type: none"> 1. Right click project and select properties 2. Click “Common Properties” 3. Click “General” 4. Select “frmMain” as your startup object in the “Startup object” drop down list 5. Click “OK”.
4. Add reference to SharePoint assembly	<ol style="list-style-type: none"> 1. Right click “References” select “Add Reference” 2. Click Browse 3. Type “C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\ISAPI” into the filename textbox and click “Open” 4. Select “Microsoft.SharePoint.dll” 5. Click “Open” 6. Click “OK”
5. Create Helper methods	<ol style="list-style-type: none"> 1. Paste the contents of 300.1.2.cbParentWebDatabind.txt into the frmMain class 2. Paste the contents of 300.1.2.EnumerateListBox.txt into the



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

	frmMain class
6. Create Event Handlers	<ol style="list-style-type: none"> 1. Double click the form to create its “Load” event handler 2. Paste the contents of <i>300.1.2.frmMain_Load.txt</i> into the “Load” event handler 3. Double click the “cmdAddSite” button to create its “Click” event handler 4. Paste the contents of <i>300.1.2.cmdAddSite_Click.txt</i> into to the “Click” event handler 5. Double click the “cmdDeleteSite” button to create its “Click” event handler 6. Paste the contents of <i>300.1.2.cmdDeleteSite_Click.txt</i> into to the “Click” event handler 7. Double click “lbWebs” list box to create its “SelectedIndexChanged” event handler 8. Paste the contents of <i>300.1.2.lbWebs_SelectedIndexChanged.txt</i> into to the “SelectedIndexChanged” event handler
7. Build and test the application	<ol style="list-style-type: none"> 1. Right click the SPObjctModelDemo project in the Solution Explorer and select Set as Startup Project. 2. Select F5 to build and run the application 3. To browse existing sites, select a site from the large list box. This updates the rest of the form 4. To add a new site, select the parent web from the parent web drop down list. Type the name of the new site in the text box next to the drop down list and click the Add Site button. 5. To delete a site, click the site in the large list box and click the delete selected site button.



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

EXERCISE 3: IMPLEMENTING DOCUMENT LIBRARY EVENT HANDLERS

Scenario

While SharePoint excels at the presentation of custom lists, frequently the documents in SharePoint require some special handling because their content is not immediately apparent at a glance. To this end, SharePoint exposes some document event handlers which help us react better when a change to one document impacts a business process.

Tasks	Detailed Steps
1. Create Event Handler class file	<ol style="list-style-type: none"> 1. Right click on the Level300LabVB project and select Add → Add Class 2. Name the class “ProductPhotoEventHandler.vb”
2. Add namespace references	<ol style="list-style-type: none"> 1. Paste the contents of <i>300.1.3.Imports.txt</i> at the top of the “ProductPhotoEventHandler.vb” file
3. Implement the required IListEventSink Interface	<ol style="list-style-type: none"> 1. Place your cursor at the end of the class declaration line “Public Class ProductPhotoEventHandler” 2. Press the enter key 3. Type the following “Implements IListEventSink” 4. Press the enter key
4. Add Event Log helper method	<ol style="list-style-type: none"> 1. Paste the contents of <i>300.1.3.WriteToWindowsEventLog.txt</i> into the “ProductPhotoEventHandler” class
5. Implement OnEvent logic	<ol style="list-style-type: none"> 1. Paste the contents of <i>300.1.3.OnEvent.txt</i> into the OnEvent method stub <p><i>Note: this snippet adds the code to retrieve the file that has been uploaded to the picture library and puts it into the AdventureWorks database</i></p>
6. Build the project	<ol style="list-style-type: none"> 1. Select Build Level300LabVB from the Build menu to build the project
7. Enable Event Handlers for this Virtual Server	<ol style="list-style-type: none"> 1. Select Start → Administrative Tools → SharePoint Central Administration 2. Click “Configure virtual server settings” 3. Click “Default Web Site” 4. Click “Virtual server general settings” under Virtual Server Management 5. Scroll to the bottom of this page and select “On” beside Event Handlers 6. Click “OK”
8. Create a Picture Library in the HOL300 portal	<ol style="list-style-type: none"> 1. Browse to the http://localhost/HOL300 portal 2. Click “Create” in the top navigation 3. Click “Picture Library” under Picture Libraries” 4. Type “Adventure Works Product Photos” in the name and



LESSON 1: EXTENDING SHAREPOINT FUNCTIONALITY

	<p>description fields</p> <ol style="list-style-type: none"> 5. Select “Yes” to display on the “Quick Launch” under Navigation 6. Select “No” under Picture Versions 7. Click “Create”
9. Install Assembly to GAC	<ol style="list-style-type: none"> 1. Click Start→All Programs→Microsoft Visual Studio .NET 2003→Visual Studio.NET Tools→ Visual Studio .NET 2003 Command Prompt 2. Type the following at the command prompt: “gacutil /i c:\inetpub\wwwroot\bin\Level300LabVB.dll” 3. Press Enter
10. Enable Event Handlers for this Picture Library	<ol style="list-style-type: none"> 1. Click “Modify settings and columns” in the left navigation of the picture library 2. Click “Change advanced settings” 3. Both the class name and the assembly name values are located in the snippet <i>300.1.2.EventHandler.txt</i> 4. In the “Assembly Name” textbox type “Level300LabVB, Version=1.0.0.0, Culture=Neutral, PublicKeyToken=23a7d484d82be380” 5. Substitute the token extracted in the Exercise 1, step 12 of this lesson 6. In the Class Name textbox type “Level300LabVB.ProductPhotoEventHandler” 7. Click “OK”
11. View an existing photo in the Product Catalog	<ol style="list-style-type: none"> 1. Browse to the http://localhost/HOL200 portal 2. Type “91” into the textbox on the Product Catalog Photo Browser Smart Part to load the blue bicycle
12. Upload a new product photo	<ol style="list-style-type: none"> 3. Browse to the HOL300 portal 4. Click on the “Adventure Works Product Photos” picture library under “Pictures” 5. Click “Add Picture” 6. Click the “Browse” button 7. Browse to the Lesson 1\Exercise 3 directory on the content CD 8. Click “91.jpg” and click “Open” 9. Click “Save and Close”
13. View updated photo in the Product Catalog	<ol style="list-style-type: none"> 10. Browse to the HOL200 portal 11. Type 91 into the textbox on the Product Catalog Photo Browser Smart Part to see the updated photo



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

Objectives

After completing this lab, you will understand the steps required to achieve the goals of this Hands on Lab

2. Create a new SharePoint workspace using Visual Basic for Applications within Microsoft Excel
3. Save a workbook to an existing SharePoint workspace from within Microsoft Excel using VBA
4. Create a shared list in SharePoint using Microsoft Excel and VBA
5. Inserting a shared list from SharePoint into a Microsoft Excel document using VBA
6. Respond to document updates in SharePoint from within Microsoft Excel

Setup

The lab machine requires the following software be pre-installed before beginning this lab:

- Windows Server 2003 + Service Pack 1
 - Install IIS and create an application server.
 - Remove FrontPage 2000 server extensions
 - Windows SharePoint Services + Service Pack 1
- SQL Server 2000 + Service Pack 4
- Reporting Services + Service Pack 2
- Microsoft Office 2003 Professional

Estimated time to complete this lab:
120 minutes



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

EXERCISE 1: CREATE A SHAREPOINT WORKSPACE FROM WITHIN EXCEL USING VISUAL BASIC FOR APPLICATIONS

Scenario

In this exercise, you will create a new workspace off of a team SharePoint site and save a new Microsoft Excel workbook to the new workspace. For this exercise, we will use Visual Basic for Applications to automate portions of our task.

NOTE: for more information on VBA, check this website:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_xl2003_ta/html/odc_super.asp

Tasks	Detailed Steps
1. Create a blank Excel workbook	<ol style="list-style-type: none"> 1. Launch Microsoft Excel by pressing Start → All Programs → Microsoft Office → Microsoft Excel 2003 2. A new, blank workbook is opened automatically
2. Set the Macro security level to Medium. This setting allows macros to run.	<ol style="list-style-type: none"> 1. In Excel, go to Tools → Macros → Security. (Macros will not be visible unless you expand the menu by clicking the down arrow at the bottom of the drop down Tools menu) 2. On the security level tab, select Medium level security 3. Click OK
3. Open the Visual Basic for Applications editor for this workbook	<ol style="list-style-type: none"> 1. In the menu bar at the top of the screen, choose Tools → Macros → Visual Basic Editor or alternatively press Alt+F11 <p><i>Notice that the Visual Basic Editor opens with two of three windows open – the Project window containing Excel objects for each of the individual sheets and one for the entire workbook, the Properties window containing the properties for each of the Excel objects, and a blank space where the code window will be.</i></p>
4. Add a new Create_SPWorkspace macro to the workbook	<ol style="list-style-type: none"> 1. Right-click the Microsoft Excel Objects folder in the Project Window 2. Select Insert → Module 3. Notice the creation of the modules folder and a Module1 object in the Project window. The code window will also open. 4. Type at the top of the code window, type “Option Explicit” and press the ENTER key to ensure Visual Basic catches any spelling mistakes we might make. Notice that by hitting ENTER after typing, that the key words turn blue 5. Create a new public Sub called Create_Workspace by typing “Public Sub Create_Workspace” and pressing ENTER. Visual Basic will create the End Sub statement for you.



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

<p>5. Add code to the new macro that will create a new document workspace</p>	<p>1. Paste the code contained within the supplied snippet file <i>300.2.1.Create_Workspace.txt</i></p>
<p>6. Save and run the macro</p>	<p>1. In the Visual Basic for Applications editor, select File → Close and Return to Microsoft Excel 2. Save your Excel file as “E1-CreateSharePointWorkspaces.xls” in you’re my Documents folder (this step gives your file a recognizable name later) by clicking File → Save As and select the My Documents folder. 3. In Microsoft Excel, select Tools → Macro → Macros. 4. This brings up the Macros dialog where you can select and run or edit your macros 5. Select your Create_SPWorkspace and press Run. 6. Wait for the hourglass icon to disappear. 7. Close Microsoft Excel</p>
<p>7. Verify the creation of your new workspace</p>	<p>1. Open your team site at http://localhost/hol300. 2. Select Documents and Lists from the top menu 3. Select Document Workspaces from the left menu 4. Select SampleVBAWorkspace from the list</p>



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

EXERCISE 2: SAVE AN EXCEL DOCUMENT TO AN EXISTING WORKSPACE WITH VBA

Scenario

In this exercise, you will save a new Excel document to a pre-existing SharePoint workspace. We will automate a portion of our work with Visual Basic for Applications. We must do the following steps:

- a. Open an existing document from the SharePoint site
- b. Get a reference to that document's SharedWorkspace object
- c. Add our workbook to the SharedWorkspace object's Files collection
- d. Close the document opened in step a
- e. Close the workbook which has just been added and re-open it from the SharePoint site (to use the shared version)

Tasks	Detailed Steps
1. Create a blank Excel workbook	<ol style="list-style-type: none"> 1. Launch Microsoft Excel by pressing Start → All Programs → Microsoft Office → Microsoft Excel 2003 2. A new, blank workbook is opened automatically
2. Create a Blank.xls in your SharePoint workspace for use as a reference	<ol style="list-style-type: none"> 1. Save the blank workbook as Blank.xls to your My Documents folder by clicking File → Save As and select the My Documents folder. 2. Open your SharePoint team site in IE (http://localhost/hol300) 3. Click Documents and Lists on the top navigation bar 4. Click Document Workspaces on the left side of the screen to see the SampleVBAWorkspace workspace created in Exercise 1 5. Click on the link to SampleVBAWorkspace workspace 6. In the Shared Documents document library in the workspace, click Add New Document 7. Click the Browse button 8. Select the My Documents quick link 9. Select the newly created Blank.xls and click Open 10. Click the Save and Close button to save the empty Excel workbook to the SharePoint Workspace
3. Return to Excel and save the Blank.xls with a new name for future use	<ol style="list-style-type: none"> 1. Return to Microsoft Excel 2. Click File → Save As and navigate to the My Documents folder 3. Name the file "E2-ExistingSharePointWorkspace.xls" and click Save
4. Open the Visual Basic for Applications editor for this workbook	<ol style="list-style-type: none"> 1. In the menu bar at the top of the screen, choose Tools → Macros → Visual Basic Editor or alternatively press Alt+F11 <p><i>Notice that the Visual Basic Editor opens with two of three windows open – the Project window containing Excel objects for each of the individual sheets and one for</i></p>



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

	<p><i>the entire workbook, the Properties window containing the properties for each of the Excel objects, and a blank space where the code window will be.</i></p>
<p>5. Add a new Existing_SPWorks pace macro to the workbook</p>	<ol style="list-style-type: none"> Right-click the Microsoft Excel Objects folder Select Insert → Module <p><i>Notice the creation of the modules folder and a Module1 object in the Project window.</i></p> <ol style="list-style-type: none"> Type at the top of the Module1 code window “Option Explicit” and press ENTER to ensure Visual Basic catches any spelling mistakes we might make. <p><i>Notice that by pressing ENTER after typing, the key words turn blue</i></p> <ol style="list-style-type: none"> Create a new public Sub called Existing_Workspace by typing “Public Sub Existing_Workspace” and pressing ENTER. Visual Basic will create the End Sub statement for you.
<p>6. Add the code to detect an existing SharePoint workspace and add this document to it</p>	<ol style="list-style-type: none"> Add the following text contained in the supplied snippet file 300.2.2.Existing_Workspace.txt to the Sub Existing_Workspace
<p>7. Save and run the macro</p>	<ol style="list-style-type: none"> In the Visual Basic for Applications editor, select File → Close and Return to Microsoft Excel Save your Excel file by clicking on the Save icon In Microsoft Excel, select Tools → Macro → Macros. <p><i>This brings up the Macros dialog where you can select and run or edit your macros</i></p> <ol style="list-style-type: none"> Select your Existing_Workspace macro and click Run Click “Yes” on the “Workbook added to shared workspace” dialog box. <p><i>This closes Excel and opens Internet Explorer.</i></p>
<p>8. Verify the addition of your new workbook to the SampleVBA workspace</p>	<ol style="list-style-type: none"> Open your team site at http://localhost/hol300. Select Documents and Lists from the top menu Select Document Workspaces from the left menu Select SampleVBAWorkspace from the list



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

EXERCISE 3: CREATE A SHARED LIST IN SHAREPOINT USING EXCEL AND VBA

Scenario

In this exercise, we will publish some Excel spreadsheet data as a new list to a SharePoint site using Visual Basic for Applications. Once the list is created in SharePoint, we will need to share it so other users and documents can take advantage of our data.

1. Tasks	Detailed Steps												
2. Create a blank Excel workbook	<ol style="list-style-type: none"> 1. Launch Microsoft Excel by pressing Start → All Programs → Microsoft Office → Microsoft Excel 2003 2. A new, blank workbook is opened automatically 												
3. Save the workbook with an understandable name for future use	<ol style="list-style-type: none"> 1. Click File → Save As 2. Name the file “E3-SharingLists.xls” and click Save 												
4. Open the Visual Basic for Applications editor for this workbook	<ol style="list-style-type: none"> 1. In the menu bar at the top of the screen, choose Tools → Macros → Visual Basic Editor or alternatively press Alt+F11 <p><i>Notice that the Visual Basic Editor opens with two of three windows open – the Project window containing Excel objects for each of the individual sheets and one for the entire workbook, the Properties window containing the properties for each of the Excel objects, and a blank space where the code window will be.</i></p>												
5. Add some data to the spreadsheet to make up our list	<ol style="list-style-type: none"> 1. Add the following data to Sheet1 starting in cell A1 <table border="1" data-bbox="609 1123 1136 1297"> <thead> <tr> <th>Employee Rank</th> <th>vacation days</th> <th>sick days</th> </tr> </thead> <tbody> <tr> <td>Junior</td> <td>10</td> <td>5</td> </tr> <tr> <td>Senior</td> <td>15</td> <td>5</td> </tr> <tr> <td>Director</td> <td>20</td> <td>10</td> </tr> </tbody> </table> 2. Save the Excel document by clicking on the Save icon 	Employee Rank	vacation days	sick days	Junior	10	5	Senior	15	5	Director	20	10
Employee Rank	vacation days	sick days											
Junior	10	5											
Senior	15	5											
Director	20	10											
6. Add a new Create_SampleList macro to the workbook	<ol style="list-style-type: none"> 1. Right-click the Microsoft Excel Objects folder 2. Select Insert → Module <p><i>Notice the creation of the modules folder and a Module1 object in the Project window.</i></p> <ol style="list-style-type: none"> 3. Type at the top of the code window “Option Explicit” and press ENTER to ensure Visual Basic catches any spelling mistakes we might make. 4. Create a new public Sub called Publish_SampleList by typing “Public Sub Publish_SampleList” and pressing ENTER. 												



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

	<p>Visual Basic will create the End Sub statement for you.</p> <p>5. Create a new public Sub called Create_SampleList by typing “Public Sub Create_SampleList” and pressing ENTER. Visual Basic will create the End Sub statement for you.</p>
7. Add the code to publish a list to SharePoint	<p>1. Add the following text contained in the supplied snippet file 300.2.3.Publish_SampleList.txt to the Sub Publish_SampleList</p>
8. Add the code to create a list in code	<p>1. Add the following text contained in the supplied snippet file 300.2.3.Create_SampleList.txt to the Sub Create_SampleList</p>
9. Verify the presence of your list data online	<p>1. In the Visual Basic for Applications editor, select File → Close and Return to Microsoft Excel</p> <p>2. Save your Excel file by clicking on the Save icon</p> <p>3. In Microsoft Excel, select Tools → Macro → Macros.</p> <p>4. Select your Create_SampleList and click Run.</p> <p>5. Click OK when prompted that you have shared your list.</p> <p><i>The final line of the Publish_SampleList subroutine takes you to your site and shows you the list online. The list shows up as part of http://localhost/hol300 (not the SampleVBA workspace)</i></p>
10. Getting ahead for the next exercise: identify the list’s GUID	<p>1. While looking at your list online, select Modify Columns and Settings</p> <p><i>Notice the GUID in the URL in the Address Bar.</i></p> <p>2. Copy this GUID to a notepad file for future use in Exercise 4.</p> <p>3. Save the notepad file to the desktop as ListGUID.txt. <i>example GUID - 8422DAE6-9929-11CF-B8D3-004033373DA8</i></p>



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

EXERCISE 4: USE A SHAREPOINT LIST IN AN EXCEL DOCUMENT WITH VBA

Scenario

In this exercise, we will retrieve some online data from a shared SharePoint list for use within a Microsoft Excel document.

Tasks	Detailed Steps
1. Create a blank Excel workbook	<ol style="list-style-type: none"> 1. Launch Microsoft Excel by pressing Start → All Programs → Microsoft Office → Microsoft Excel 2. A new, blank workbook is opened automatically
2. Save the workbook with an understandable name for future use	<ol style="list-style-type: none"> 1. Click File → Save As 2. Name the file E4-UsingSharedLists.xls, navigate to the My Documents folder and click Save.
3. Open the Visual Basic for Applications editor for this workbook	<ol style="list-style-type: none"> 1. In the menu bar at the top of the screen, choose Tools → Macros → Visual Basic Editor or alternatively press Alt+F11 <p><i>Notice that the Visual Basic Editor opens with two of three windows open – the Project window containing Excel objects for each of the individual sheets and one for the entire workbook, the Properties window containing the properties for each of the Excel objects, and a blank space where the code window will be.</i></p>
4. Add a new Retrieve_SampleList macro to the workbook	<ol style="list-style-type: none"> 1. Right-click the Microsoft Excel Objects folder 2. Select Insert → Module <p><i>Notice the creation of the modules folder and a Module1 object in the Project window.</i></p> <ol style="list-style-type: none"> 3. Type at the top of the code window “Option Explicit” and press ENTER to ensure Visual Basic catches any spelling mistakes we might make. 4. Create a new public Sub called Retrieve_SampleList by typing “Public Sub Retrieve_SampleList” and pressing ENTER. Visual Basic will create the End Sub statement for you.
5. Add the code to create a list in code	<ol style="list-style-type: none"> 1. Add the following text contained in the supplied snippet file 300.2.4.Retrieve_SampleList.txt to the Sub Retrieve_SampleList 2. Open the ListGUID.txt file from the desktop that you saved in Exercise 3, step 10 and replace the GUID in the line below and type it at the bottom of your Retrieve_SampleList subroutine ' use the GUID identified in Exercise 3, step 10



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

	<code>src(1) = "{BD318365-BFC3-4CFF-B497-A888E1638375}"</code>
6. Add a new DiscardLocalChanges_SampleList macro to the workbook	<ol style="list-style-type: none"> 1. Create a new public Sub in Module1 called DiscardLocalChanges_SampleList by typing “Public Sub DiscardLocalChanges_SampleList” and pressing ENTER. <i>Visual Basic will create the End Sub statement for you.</i>
7. Add the code to refresh the list with the SharePoint list	<ol style="list-style-type: none"> 1. Add the following text contained in the supplied snippet file 300.2.4.DiscardLocalChanges_SampleList.txt to the Sub DiscardLocalChanges_SampleList
8. Add a new UpdateSPWithLocalChanges_SampleList macro to the workbook	<ol style="list-style-type: none"> 1. Create a new public Sub in Module1 called UpdateSPWithLocalChanges_SampleList by typing “Public Sub UpdateSPWithLocalChanges_SampleList” and pressing ENTER. <i>Visual Basic will create the End Sub statement for you.</i>
9. Add the code to update a SharePoint list with local changes	<ol style="list-style-type: none"> 1. Add the following text contained in the supplied snippet file 300.2.4.UpdateSPWithLocalChages_SampleList.txt to the Sub UpdateSPWithLocalChages_SampleList
10. Save the macros	<ol style="list-style-type: none"> 1. In the Visual Basic for Applications editor, select File → Close and Return to Microsoft Excel 2. Save your Excel file by clicking on the Save icon
11. Run the Retrieve_SampleList macro	<ol style="list-style-type: none"> 1. In Microsoft Excel, select Tools → Macro → Macros <i>This brings up the Macros dialog where you can select and run or edit your macros</i> 2. Select your Retrieve_SampleList and press run <i>Notice that Excel now has your list data in cells A1:C4</i>
12. Make a change to the data locally then discard the update	<ol style="list-style-type: none"> 1. In Excel, on Sheet 1, modify cell D4 from 10 to 7 (days) 2. Select Tools → Macro → Macros. <i>This brings up the Macros dialog where you can select and run or edit your macros</i> 3. Select your DiscardLocalChages_SampleList and Click Run <i>Notice that cell D4 now reads 10 again.</i>



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

<p>13. Make a change to the data locally and use the changed data to update SharePoint</p>	<ol style="list-style-type: none">1. In Excel, on Sheet 1, modify the cell D4 from 10 to 7 (days)2. Select Tools → Macro → Macros. <i>This brings up the Macros dialog where you can select and run or edit your macros</i>3. Select your UpdateSPWithLocalChages_SampleList macro and Click Run4. Open IE and navigate to http://localhost/hol300/Lists/SharePointSharedList/AllItems.aspx <i>Notice that the published list has been updated to read 7 sick days allowed for executives instead of 10 sick days allowed.</i>
--	---



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

EXERCISE 5: RESPOND TO SHAREPOINT DOCUMENT UPDATES IN EXCEL USING VBA

Scenario

In this exercise, you will create an Excel spreadsheet which is self updating when documents are updated inside of a team SharePoint site. We will be using Visual Basic for Applications to accomplish this task

Tasks	Detailed Steps
1. Open the Excel workbook created in Exercise 4	<ol style="list-style-type: none"> 1. Launch Microsoft Excel by pressing Start → All Programs → Microsoft Office → Microsoft Excel 2. Select File → Open and browse to E4-UsingSharedList.xls in My Documents folder 3. Click Open 4. Click Enable Macros when the Security Warning dialog box appears
2. Open the Visual Basic for Applications editor for this workbook	<ol style="list-style-type: none"> 1. In the menu bar at the top of the screen, choose Tools → Macros → Visual Basic Editor or alternatively press Alt+F11 <p><i>Notice that the Visual Basic Editor opens with two of three windows open – the Project window containing Excel objects for each of the individual sheets and one for the entire workbook, the Properties window containing the properties for each of the Excel objects, and a blank space where the code window will be.</i></p>
3. Add a new Workbook_Sync macro to the workbook	<ol style="list-style-type: none"> 1. Double-click the ThisWorkbook icon to open the code window. 2. Change the top left dropdown that reads (General) to Workbook. 3. Change the top right dropdown that reads Open to Sync. 4. Type at the top of the code window “Option Explicit” to ensure Visual Basic catches any spelling mistakes we might make. <p><i>Notice that by hitting ENTER after typing that the key words turn blue</i></p> <p><i>Notice that by selecting Sync in the previous step, the VBA editor automatically created a Private Sub Workbook_Sync for you.</i></p>
4. Write the code that will refresh the list.	<ol style="list-style-type: none"> 1. Add the following text contained in the supplied snippet file 300.2.4.Workbook_Sync.txt to the Sub Workbook_Sync
5. Make a change to the data	<ol style="list-style-type: none"> 1. In Excel, on Sheet 1, modify cell D4 from 7 to 12 (days)



LESSON 2: USING SHAREPOINT TO CREATE AUTOMATED SYSTEMS

locally then discard the update	<ol style="list-style-type: none">2. Save your workbook3. Click OK when the Microsoft Excel dialog appears4. Inside of 10 minutes, the workbook will automatically sync with SharePoint and overwrite your changes to D4.
---------------------------------	---



LESSON 3: INFOPATH FORMS AND WORKFLOW LITE

Objectives After completing this lab, you will that understand that SharePoint has the ability to use third party workflow engines to automate work tasks. You'll also get a glimpse of the power of the integration between InfoPath and SharePoint

Prerequisites It is important that you understand the concepts of what SharePoint is and how to develop for SharePoint. You will be well prepared if you have completed the 100 level and 200 level Hands on Labs material before attempting this lesson.

Setup The lab machine requires the following software be pre-installed before beginning this lab:

- Windows Server 2003 + Service Pack 1
 - Install IIS and create an application server.
 - Remove FrontPage 2000 server extensions
- SQL Server 2000 + Service Pack 4
- Microsoft Office 2003 Professional
- Microsoft InfoPath 2003
- WSS
- Visual Studio 2003
- SharePoint templates for Visual Studio
- Workflow Lite downloaded from the GotDotNet workspace:
<http://www.gotdotnet.com/workspaces/workspace.aspx?ID=C07F64E8-8229-49A1-B160-B24C89122894>

Estimated time to complete this lesson: 30 minutes



EXERCISE 1: INFOPATH FORMS AND FORMS LIBRARIES

Scenario

InfoPath is a great new tool in the Microsoft Office Suite which, when combined with SharePoint, offers some real improvements to business processes. In this example, we will create a new InfoPath form to track purchases for our fictional Adventure Works company.

Tasks	Detailed Steps						
1. Create a new InfoPath form document	<ol style="list-style-type: none"> 1. Click Start → All Programs → Microsoft Office → Microsoft Office InfoPath 2003 2. In the Fill Out a Form dialog that pops up, select Design a Form on the left side 3. In the Design a Form task pane, select a New Blank Form 						
2. Design the layout for our form	<ol style="list-style-type: none"> 1. In the Design Tasks task pane, select Layout 2. In the Insert Layout Tables, select the Table with Title option 3. In the content window you will see a table with a title bar at the top. In this title bar, type “Adventure Works Purchase Order” 4. Move your cursor to beneath the table containing the form title. 5. Press Enter 6. From the menu bar, click Insert → Layout Table. 7. Use the popup menu to add a table with 3 columns and 2 rows. 8. In the cells of your table, type “Sales Representative” in the top left, “Location” in the top right, “E-mail” in the bottom left, and “Department” in the bottom right. 9. At this point, your form should look like this: <div style="border: 1px dashed gray; padding: 5px; margin: 10px 0;"> <p style="text-align: center;">Adventure Works Purchase Order</p> <hr style="border: 1px solid gray;"/> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%; padding: 2px;">Sales Representative</td> <td style="width: 33%; padding: 2px;">Location</td> <td style="width: 33%; padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">E-mail</td> <td style="padding: 2px;"></td> <td style="padding: 2px;">Department</td> </tr> </table> </div> 10. Modify the second table by dragging the middle columns closer together. Like the screen shot below 11. In the Layout task pane, select Controls at the top to reveal the controls which may be dragged onto the form 12. Drag a text box next to Sales Representative and another one next to E-mail 13. Double click the text box under Sales Representative and give the field the name “SalesRep” on the Text Box Properties dialog 14. Double click the text box under Email and give the field the name “Email” on the Text Box Properties dialog 15. Drag a drop-down list box next to Location and another one next to Department 	Sales Representative	Location		E-mail		Department
Sales Representative	Location						
E-mail		Department					



	<p>16. Double click the drop down list under Location and name it “Location”</p> <p>17. In the same drop down list properties dialog, change the List Box Entries radio selection to “Look up values in a data connection to a database, web service, file, or SharePoint library or list”</p> <ol style="list-style-type: none"> a. Click Add next to Data Connection b. Select the database radio button. Press Next c. Click Select Database d. Double click “New SQL Server connection.odc” e. Type (local) in the server name field and click Next f. Select the AdventureWorks2000 database from the database drop down list g. Select the SalesTerritory table and click Next h. Type “SalesTerritory” in the description box and click Finish i. Uncheck everything except Name (you will be unable to clear the TerritoryID field – this is OK) j. Click Next then Finish k. Click the Select XPath button next to the Entries field l. Expand the tree view and select Name from the list and click OK m. Click OK on the Drop Down List Properties dialog <p>18. Double click the drop down list under Department and name it “Dept”</p> <p>19. In the same drop down list properties dialog, change the List Box Entries radio button to “Look up values in a data connection to a database, web service, file, or SharePoint library or list”</p> <ol style="list-style-type: none"> a. Click Add next to Data Connection b. Select the database radio button. Press Next c. Click Select Database d. Double click “New SQL Server connection.odc” e. Type (local) in the server name field and click Next f. Select the AdventureWorks2000 database from the database drop down list g. Select the Department table and click Next h. Type “Departments” in the description box and click Finish i. Uncheck everything except Name (you will be unable to clear the DepartmentID field – this is OK) j. Click Next then Finish k. Click the Select XPath button next to the Entries field l. Expand the tree view and select Name from the list and click OK m. Click OK on the Drop Down List Properties dialog <p>20. At this point, your form should look like this:</p>
--	---

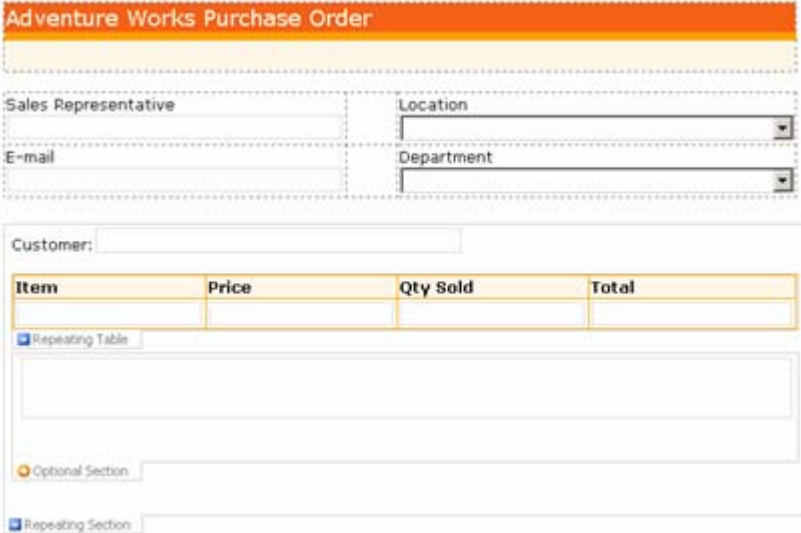


	<p>Adventure Works Purchase Order</p> <p>Sales Representative <input type="text"/></p> <p>E-mail <input type="text"/></p> <p>Location <input type="text" value="Select..."/></p> <p>Department <input type="text" value="Select..."/></p> <p>21. Double click on the text box under Sales Representative to open the Text Box properties. On the Data tab, check the box that says “Cannot be blank”</p> <p><i>This is to specify that the Sales Representative field is required.</i></p> <p>22. Specify that the E-mail text box must contain an e-mail address by adding some data validation</p> <ol style="list-style-type: none"> Double click the text box under E-mail to get to the Text Box properties dialog On the Data tab, click Data Validation Click Add in the Data Validation dialog box Fill out the first two dialog boxes so that it reads “If this condition is true, [email] [does not match pattern] To fill in the last dialog box, pull down the drop down and click “Select a pattern” Select “Custom pattern” from the list In the Insert special character drop down, select “Any Letter” In the Insert special character drop down, select “One or more: +” In the Custom pattern text box, type @adventureworks at the end of your current pattern In the Insert special character drop down, select “Period” In the Custom pattern text box, type com at the end of your current pattern. <p><i>At this point your custom pattern should look like this,</i> “\p{L}+@adventureworks\.com”</p> <ol style="list-style-type: none"> Click OK on this dialog Enter “Please enter a valid AdventureWorks.com e-mail address” in the Screen Tip text box Close any open dialog boxes by clicking “OK” <p>23. Place your cursor underneath the Layout table with the Sales Representative fields and press Enter.</p> <p><i>This is to add a place for some sales details.</i></p>
--	--



	<p>24. In the Controls task pane, drag a Repeating Section onto the form</p> <p>25. Inside the repeating section, type “Customer:”</p> <p>26. Drag a text box next to the customer text just added.</p> <p>27. Double click this text box and go to the Size tab of the Text Box Properties dialog. Specify a width of 350 and click OK on this dialog.</p> <p>28. From the Controls task pane, drag a Repeating Table onto the form inside the Repeating Section, under the Customer text box. Specify 4 columns and press OK</p> <p>29. Give each column a title, “Item”, “Price”, “Qty Sold”, “Total” respectively</p> <p>30. Double click the Price column’s input field and change the field name to “Price” and click OK</p> <p>31. Double click the Qty Sold column’s input field and change the field name to “Qty” and click OK</p> <p>32. Double click the Total column’s input box, name the field “Total” then click Apply and on the Data tab, click the formula button (Σ)</p> <ol style="list-style-type: none"> a. In the Insert Formula dialog, click the Insert Field or Group button and select the Price field and click OK. b. Type a “*” after the Price field c. Click the Insert Field or Group button again and double click Qty field. d. Click OK to close any open dialogs <p>33. Drag an Optional Section to underneath your repeating table inside the repeating section.</p> <p><i>To add an optional location for notes,</i></p> <p>34. Inside the Optional Section, drag a Rich Text Box</p> <p>35. Double click the Optional Section tab (located under the section) to bring up it’s properties dialog.</p> <p>36. At the bottom of the dialog, change the Show insert button and hint text to read “Click here to insert optional notes” and click OK</p> <p>37. Give some color to your form design by selecting Format → Color Schemes → Orange</p> <p>38. Your form should now look like this:</p>
--	---



	 <p>39. Save your InfoPath form as AdvWorksPO by pressing File→Save. Click the Save button. Click the My Documents shortcut and press Save.</p>
<p>3. Test your form</p>	<ol style="list-style-type: none"> 1. In the Standard toolbar, click the Preview Form button 2. Enter data into all the fields 3. Test adding optional notes, multiple items, and multiple customers 4. Click the Close Preview button from the Standard toolbar
<p>4. Publish the form to SharePoint</p>	<ol style="list-style-type: none"> 1. Select File → Publish to open the Publishing Wizard click Next 2. Select that you want to publish this form to a SharePoint library and click Next 3. Select that you want to create a new SharePoint form library and click Next 4. Type “http://localhost/hol300” in the URL input box and click Next 5. Give the library the name “Adventure Works Purchase Orders” and type “Adventure Works Purchase Order InfoPath form” into the description box click Next 6. In the next screen, click Add and select the Sales Rep click OK repeat for the Dept field to add these values as custom columns to our form library view 7. Click Finish and allow InfoPath to create your form library 8. On the final dialog box, check the box to “Open this form from its published location” and you will be taken to your new Adventure Works Purchase Orders form library by Close
<p>5. Fill out a sample form</p>	<ol style="list-style-type: none"> 1. In the form library, click “Fill out this form” to open a new Purchase Order InfoPath form. 2. Fill out your form and Click Save to submit it back to SharePoint 3. Save the document as PO1 4. Click Fill Out a Form from the Standard toolbar menu



	<ol style="list-style-type: none"> 5. Fill out your form and Click Save to submit it back to SharePoint 6. Save the document as PO2 <p><i>We will use both forms in our next step</i></p>
<ol style="list-style-type: none"> 6. Merge two forms together 	<ol style="list-style-type: none"> 1. Open Internet Explorer and navigate to http://localhost/hol300/AdventureWorksPurchaseOrders/forms/AllItems.aspx 2. In the form library click form PO1 and click Open 3. Inside this form, in InfoPath, select File → Merge Forms 4. Open PO2 by typing “http://localhost/hol300” into the File name field and clicking Enter. Double click the AdventureWorks Purchase Orders forms library and select PO2. Click Merge <p><i>Your InfoPath form now has all the sales data in it from both forms</i></p> <ol style="list-style-type: none"> 5. Save the form to your My Documents folder as “SamplePOForm” by clicking File → Save
<ol style="list-style-type: none"> 7. Export the contents to Microsoft Excel 	<ol style="list-style-type: none"> 1. Select File → Export to → Microsoft Excel to start the Export to Excel Wizard <p><i>As it is frequently easier to deal with tabular data in Microsoft Excel, this allows you to export the purchase order details to Excel.</i></p> <ol style="list-style-type: none"> 2. Click Next 3. Select Form fields and this table or list 4. Highlight group 4 (the longer entry which represents the merged fields) and click Finish 5. Microsoft Excel will open with your merged data.



EXERCISE 2: INSTALLING AND CONFIGURING WORKFLOW LITE

Scenario

We are going to examine the concepts behind workflows by looking at a free Workflow Lite product. We're going to be working with two document libraries to demonstrate our workflow.

Tasks	Detailed Steps
1. Install the Workflow Lite engine	<ol style="list-style-type: none"> 1. Run the <i>Workflow Lite for SharePoint RC1.c.msi</i> located on the Content CD in the Tools and Downloads folder by double-clicking on it 2. Click Next, accept the license agreement and click Next again 3. Accept the default install location 4. Install the package for everyone and click Next 5. Click Next to begin the installation 6. Click Next after having seen the directions <p><i>Note: these directions have been restated in this lab document for the purposes of this HOL so you don't need to save them at this point</i></p> <ol style="list-style-type: none"> 7. Click Close to complete and close the MSI installer
2. Enable the SharePoint Event Handlers in the Central Administrator	<ol style="list-style-type: none"> 1. Start → Administrative Tools → SharePoint Central Administration 2. Click "Configure virtual server settings" 3. Select Default Web Site 4. Choose "Virtual server general settings" under the Virtual Server Management category 5. Ensure Event handlers are: "On" (bottom of the page) 6. Click OK 7. Click the Home button on IE to return to http://localhost/hol300
3. Set up configuration settings for the Workflow product	<ol style="list-style-type: none"> 1. Open a Windows Explorer window by double-clicking on My Computer 2. Navigate to C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\CONFIG 3. Make a backup of the machine.config file by selecting the file and clicking Copy then clicking Paste. A new file, Copy of machine.config will appear. 4. Right-click the file and select Open with... Pick Notepad from the list 5. Search for the <appSettings> section in the file Add the contents of code snippet 300.3.2.MachineConfig.txt between the <appSettings> and </appSettings> tags 6. Remove comment markers from around the <appSettings> section to enable your changes.



	7. Save and close the config file
4. Create a new Forms library	<ol style="list-style-type: none"> 1. Browse to http://localhost/hol300 2. Click the Create link at the top of the page 3. Select Document Library 4. Type “POReview” in the name and description boxes. Accept the default choices for all the other options 5. Click the Create button 6. Click the Home button to go back to the team site 7. Click Modify Shared Page → Add Web Parts → Browse 8. Drag the POReview document library from the list into the left web part zone. 9. Drag the AdventureWorks Purchase Orders form library created in Exercise 1 to the left web part zone. 10. Close the web part gallery to finish designing the page by clicking the Home button in IE
5. Add a Status column to the AdventureWorks Purchase Orders form library	<ol style="list-style-type: none"> 1. Click on the title bar for AdventureWorks Purchase Orders to enter the form library. 2. Click the link to modify columns and settings on the left side of the screen 3. Click the link Add a new column in the Columns section 4. Name the column “Status” 5. Set the new field to be a Choice field and add the following values to the choices listbox: “Ready for Review”, “Please Revise” 6. Set these values to be presented as radio buttons 7. Set Ready for Review to be the default selection. 8. Click OK to add the column.
6. Add a document event handler to your AdventureWorks Purchase Orders form library	<ol style="list-style-type: none"> 1. Still in the Customize AdventureWorks Purchase Orders screen, choose Change Advanced Settings 2. In the Event Handler section fill out following values located in the snippet <i>300.3.2.EventHandler.txt</i>: Assembly Name: Leadit.SharePoint.Workflow, Version=0.2.0.0, Culture=neutral, PublicKeyToken=3948f234bbbabe18 Class Name: Leadit.SharePoint.Workflow.EventHandler 8. Click OK to accept your changes
7. Add a Status column to the POReview document library	<ol style="list-style-type: none"> 1. Click on the title bar for POReview to enter the document library. 2. Click the link to modify columns and settings on the left side of the screen 3. Click the link “Add a new column” 4. Name the column “Status”, 5. Set the new field to be a Choice field and add the following values to the choices listbox: “Awaiting Review”, “Approved”,



	<p>“Rejected”</p> <ol style="list-style-type: none"> Set these values to be presented as radio buttons Set Awaiting Review to be the default selection. Click OK to add the column.
8. Create a new view for the POReview document library	<ol style="list-style-type: none"> Still in the Customize POReview screen, choose Create a new view at the bottom of the screen Click Standard View Name your view “StatusView” Check the box to make this view the default view In the columns section, leave only the following columns checked: Type, Name, Modified By, and Status Change the first drop down in the Sort category to Status Click OK Click the Home link at the top of the page Click the down arrow on the title bar of POReview and select Modify Shared Web Part Set the Current View drop down on the right side of the screen to StatusView Click OK on the warning pop up message about changing the views potentially disabling web part connections Click OK at the bottom of the pane on the right
9. Add a document event handler to your POReview form library	<ol style="list-style-type: none"> Still in the Customize POReview screen, choose Change Advanced Settings In the Event Handler section fill out following values located in the code snippet <i>300.3.2.EventHandler.txt</i>: Assembly Name: Leadit.SharePoint.Workflow, Version=0.2.0.0, Culture=neutral, PublicKeyToken=3948f234bbbabe18 Class Name: Leadit.SharePoint.Workflow.EventHandler Click OK to accept your changes
10. Update the configuration file for Workflow Lite to move	<ol style="list-style-type: none"> Navigate to the Workflow Lite directory with Windows Explorer(“C:\Program Files\Workflow Lite for SharePoint RC1”) Locate config.xml and make a backup by selecting the file and clicking Copy then clicking Paste. A new file, Copy of config.xml will appear. Select Edit → paste to create a backup copy of the default configuration Replace the contents of config.xml with the <i>300.3.2.SampleConfig.txt</i> supplied with this Hands on Lab to get the correct settings:
11. Run IISReset to refresh the config files	<ol style="list-style-type: none"> Start → Run Type “cmd”



<p>in memory</p>	<p>3. In the command prompt window, type, “iisreset” and click OK.</p> <p><i>This will reload the machine.config and the config.xml files you have just changed</i></p>
<p>12. Use the workflow</p>	<ol style="list-style-type: none"> 1. Browse to http://localhost/hol300. 2. Click the Add new form link under the AdventureWorks Purchase Order form library 3. Click Browse and select the “SamplePOForm.xml” from your’re my Documents folder 4. Ensure the Status field is set to “Ready for Review” and click Save and Close. 5. Click the Home link at the top of the page. <p><i>Once the page reloads, the form should now be in the POReview document library with a status of Awaiting Review</i></p> <ol style="list-style-type: none"> 6. Click on the header link to the POReview document library to enter the POReview document library 7. Click the pull-down menu next to your SamplePOForm.xml and select Edit Properties 8. Change the Status property to “Rejected” 9. Click the Home link at the top of the page <p><i>Once the page reloads, the form should now have moved back to the AdventureWorks Purchase Order library</i></p> <ol style="list-style-type: none"> 10. Click the header link to the AdventureWorks Purchase Order form library to enter the AdventureWorks Purchase Order form library 11. Click the pull-down menu next to your SamplePOForm.xml and select Edit Properties 12. Change the Status property to “Ready for Review” 13. Click the Home link at the top of the page <p><i>Once the page reloads, the form should now have moved back to the POReview library with a status of Awaiting Review</i></p> <ol style="list-style-type: none"> 14. Click on the header link to the POReview document library to enter the POReview document library 15. Click the pull-down menu next to your SamplePOForm.xml and select Edit Properties 16. Change the Status property to “Accepted” 17. Click the Home link at the top of the page



	<p><i>Once the page reloads, the form should have stayed in the POReview library with a status of Accepted</i></p>
--	--



LESSON 4: ADVANCED WEB PART DEVELOPMENT

Objectives

After completing this lab, you will understand the steps required to achieve the goals of this Hands on Lab.

1. Understand the mechanics of Connectable Web Parts
2. Create a Provider Web Part
3. Create a Consumer Web Part

Setup

The lab machine requires the following software be pre-installed before beginning this lab:

- Windows Server 2003 + Service Pack 1
 - Install IIS and create an application server.
 - Remove FrontPage 2000 server extensions
- SQL Server 2000 + Service Pack 4
- Microsoft Office 2003 Professional
- Windows SharePoint Services
- AdventureWorks 2000 Sample Database
- Microsoft Visual Studio .Net 2003

Estimated time to complete this lesson: 30 minutes



EXERCISE 1: CREATING A PROVIDER WEB PART

Scenario

It is often desirable to have Web Parts that are capable of communication with each other. Fortunately, the SharePoint Web Part model has features in place that support this functionality.

In this example, we wish to display a listing of the Special Discount Offer promotions in a web part, and when a particular Offer is selected, secondary web parts should display further details.

Tasks	Detailed Steps
1. Add a new C# Web Part Library to the Solution	<ol style="list-style-type: none"> 1. Open Visual Studio .NET 2003 2. Create or Open the Solution named C:\Projects\SharePoint123 3. Right-click the solution in the Solution Explorer, and select “New Project” from the “Add” menu. 4. Under “Visual C# Projects” select “Web Part Library” 5. Enter “Level300LabCs” for the Project name, and click OK to create the project.
2. Add a needed assembly reference	<ol style="list-style-type: none"> 1. Right-click the new project and select “Add Reference” from the context menu. 2. On the “.NET” tab, select “System.Data.Dll”, and click the “Select” button to add it to the “Selected Components” list. 3. Click “OK” to finish adding the assembly reference.
3. Delete the default web part files	<ol style="list-style-type: none"> 1. Right-click the “WebPart1.cs” file in the solution explorer, and select “Delete”. Click “OK” to confirm the permanent deletion of the file. 2. Right-click the “WebPart1.dwp” file in the solution explorer, and select “Delete”. Click “OK” to confirm the permanent deletion of the file.
4. Add a constants file containing the SQL queries that will be used by the Web Parts in this project	<ol style="list-style-type: none"> 1. Right-click the Project in the solution explorer and select “Add Class” from the “Add” menu. 2. Name the new class file “SQLStatements.cs” and click “Open” to create and open the new file. 3. Replace all of the contents of this new file with the text contained in the supplied snippet file 300.4.1.SQLStatements.cs.txt 4. Save the file and close it. <p><i>Note: These SQL queries are used to populate the databound grids that are used by the web parts.</i></p>
5. Add a utility class to handle grid data-	<ol style="list-style-type: none"> 1. Right-click the Project in the solution explorer and select “Add Class” from the “Add” menu



LESSON 4: ADVANCED WEB PART DEVELOPMENT

<p>binding, paging, and sorting features as well as UI configuration</p>	<ol style="list-style-type: none"> 2. Name the new class file “WebGridUtility.cs” and click “Open” to create and open the new file 3. Replace all of the contents of this new file with the text contained in the supplied snippet file 300.4.1.WebGridUtility.cs.txt 4. Save the file and close it. <p><i>Note: This helper class is designed to interact with a DataGrid object. It applies a consistent UI as well as handling common grid functionality such as Paging and Sorting. These tasks are somewhat more complex when done inside of a Web Part, since you need to take into account the control hierarchy when attaching event handlers and dealing with viewstate.</i></p>
<p>6. Create a Strong-Naming Key Pair file to sign the web part assembly</p>	<ol style="list-style-type: none"> 1. Launch a Visual Studio Command Prompt (found at Start → All Programs → Microsoft Visual Studio .Net 2003 → Visual Studio .Net Tools → Visual Studio .Net Command Prompt) 2. Change the current directory by executing the command “cd C:\Projects\Sharepoint123\Level300LabCs” 3. Create the Strong Name Key Pair file by executing the command “sn -k KeyPair.snk” <p><i>If the command was successful, you should see a message stating “Key pair written to KeyPair.snk”</i></p>
<p>7. Assign the Key Pair to the web part library assembly</p>	<ol style="list-style-type: none"> 1. Double-click the “AssemblyInfo.cs” project file in the solution explorer to open it in the code editor window. 2. Locate the “AssemblyKeyFile” attribute and replace it with the attribute text contained in the supplied snippet file 300.4.1.AssemblyInfo.cs.txt 3. Save the file and close it.
<p>8. Add the new Web Part using the template wizard</p>	<ol style="list-style-type: none"> 1. Right-click the Project in the solution explorer and select “Add New Item” from the “Add” submenu. 2. Select the Provider Web Part template from the Local Project Items category. 3. Name the new file “SpecialOffers.cs”
<p>9. Remove unnecessary default code</p>	<ol style="list-style-type: none"> 1. Remove the “DefaultProperty” attribute from the SpecialOffers class. 2. Remove the “defaulttext” string constant. 3. Remove the “text” private string member variable. 4. Remove the “Text” property and its associated attributes.
<p>10. Alter default web part communication interface definition code</p>	<ol style="list-style-type: none"> 1. Change the “cellName” private string member variable string value to the value “SpecialOfferID”. 2. Update the call to RegisterInterface within the EnsureInterfaces method by changing the menuLabel parameter from “Provides a



	<p>cell to” to “Provides a Special Offer ID to”.</p> <p>3. Also update the description parameter from “Provides a cell of data” to “Provides the ID of a Special Offer Promotion”.</p> <p>4. Update the interfaceName parameter from “CellProvider_WPQ_” to “SpecialOfferProvider_WPQ_”.</p> <p>5. Update the interfaceClientReference parameter from “CellProvider_WPQ_” to “SpecialOfferProvider_WPQ_”.</p>
11. Add a property to track the currently selected Offer ID in ViewState	1. Paste the code contained within the supplied snippet file 300.4.1.SpecialOffers_OfferID.txt in the class.
12. Add an instance of the GridUtil class, and initialize it	<p>1. Paste the code contained within the supplied snippet file 300.4.1.SpecialOffers_GridUtil.txt into the class.</p> <p><i>Note: This helper class handles the details of databinding, paging, and sorting of grids as well as applying a consistent “look and feel” to the grids in our web parts. We create the DataGrid itself as well as the GridUtil in the web part’s OnInit method because this is the point where we need to create any objects that have postback event handlers attached to them.</i></p>
13. Add an event handler to respond to selection of Special Offer rows	<p>1. Paste the code contained within the supplied snippet file 300.4.1.SpecialOffers_Selected.txt in the class</p> <p><i>Note: This simply updates the internal tracking property, which is used later to broadcast the current Special Offer ID to any listening consumer web parts.</i></p>
14. Broadcast changes in the selected Special Offer to any listening Consumer Web Parts	<p>1. Find the PartCommunicationMain method.</p> <p><i>This method is invoked by the SharePoint Web Part framework every time the web part is processed, in order to broadcast any data to connected peer web parts.</i></p> <p>2. Modify the Cell assignment from “cellReadyArgs.Cell = Text” to “cellReadyArgs.Cell = OfferID”</p>
15. Add code to DataBind and display the currently selected row during the web part’s PreRender stage	<p>1. Paste the code contained within the supplied snippet file 300.4.1.SpecialOffers_PreRender.txt</p> <p><i>Note: Data binding is postponed until the PreRender stage in case we need to alter the source query that drives the data grid in our web part.</i></p>
16. Render the Web Part contents	<p>1. Find the RenderWebPart method.</p> <p>2. Replace the call to “output.Write()” with “GridUtil.RenderToWebPart(output)”</p>



	<p>3. Save the file.</p> <p><i>Note: The GridUtil class will output a simple replacement string if there is no results data to display.</i></p>
<p>17. Change the Project Output Directory</p>	<ol style="list-style-type: none"> 1. Right-click the project in the solution explorer and click “Properties” from the context menu. 2. Select the “Build” node under “Configuration Properties” in the project property dialog. 3. Change the “Output Path” option to “C:\Inetpub\wwwroot\bin” and click the “OK” button.
<p>18. Build the Web Part project</p>	<ol style="list-style-type: none"> 1. Select “Build Level300LabCs” from the Build menu. 2. Verify that the project is compiled successfully. <p>** If there are any build errors, compare your code to the contents of the completed source code file included in “300.4.1.SpecialOffers.cs.txt”</p>
<p>19. Extract the Public Key Token from the compiled assembly</p>	<ol style="list-style-type: none"> 1. Launch a Visual Studio Command Prompt (found under the Visual Studio .NET 2003 start menu) 2. Extract the Public Key Token by executing the command “sn -T C:\Inetpub\wwwroot\bin\Level300LabCs.dll” <p><i>If the command was successful, you should see a message stating “Public key token is xxxxxxxxxxxxxxxx”</i></p> <p>Keep this value readily available, it will be needed in future steps.</p>
<p>20. Create a DWP file for the web part installation information</p>	<ol style="list-style-type: none"> 1. Right-click the project in the solution explorer and select “Add New Item” from the “Add” menu. 2. Select the Web Part Dwp template from Local Project Items. 3. Name the new file “SpecialOffers.dwp” and click the “Open” button to create and open the dwp file. 4. Replace the contents of the new file with the contents contained within the supplied snippet file 300.4.1.SpecialOffers.dwp.txt. 5. Update the PublicKeyToken value with the actual token extracted in the previous step (example “9c3ec2f3cb594a8b”). 6. Save the file.
<p>21. Update the web.config SafeControls element to mark this web part library as “safe”</p>	<ol style="list-style-type: none"> 1. Open “C:\inetpub\wwwroot\web.config” in a text editor. 2. Locate the <SafeControls> element. 3. Add a new SafeControl entry by pasting the contents of the supplied snippet file 300.4.1.SafeControl.txt



	<ol style="list-style-type: none"> 4. Update the PublicKeyToken value with the actual token extracted in a previous step (example “9c3ec2f3cb594a8b”).
<p>22. Verify the site’s trust level</p>	<ol style="list-style-type: none"> 1. Find the <trust> element in the web.config file. 2. Verify that the “level” attribute is set to either “Full” or “WSS_Medium”. <p><i>In order to execute SQL commands from a web part (without implementing custom CAS policy or installing to the GAC), we need to grant a minimum of WSS_Medium trust to the SharePoint web application.</i></p> <ol style="list-style-type: none"> 3. Save the file. 4. Click Start → Run, then type “iisreset” and press Enter
<p>23. Install the Web Part into SharePoint</p>	<ol style="list-style-type: none"> 1. Open Internet Explorer and navigate to the root of the Lab portal at http://localhost/hol300/default.aspx 2. Click “Modify Shared Page” and then “Import” from the “Add Web Parts” submenu. 3. Browse to the DWP file that was created in an earlier step which should be located at “C:\Projects\Sharepoint123\Level300LabCs\SpecialOffers.dwp” 4. Click the “Upload” button to upload the web part definition into SharePoint. 5. Verify that there is now a “Special Offer List” entry listed under “Uploaded Web Part”. 6. Drag the uploaded web part from the “Uploaded Web Part” section into the “Left” web part zone. <p><i>If all steps have been performed correctly, the Special Offer List should now be displayed.</i></p> <ol style="list-style-type: none"> 7. Click the “Home” link at the upper left of the portal page to quickly exit the customization mode. There will be records in the datagrid of the webpart. <p><i>Sorting and Paging should work, as well as Selection. However, no Consumer Web Parts have been built yet, so there is nothing to send data to as the selection changes.</i></p>



EXERCISE 2: CREATING A CONSUMER WEB PART

Scenario

Provider Web Parts are responsible for pushing information to other web parts, but we need a Consumer Web Part in order to listen for that information.

In this example, we will build a Consumer Web Part that listens for selected Offers and displays the list of Products that are a part of that promotion.

Tasks	Detailed Steps
1. Add the new Web Part using the template wizard	<ol style="list-style-type: none"> 1. Right-click the Project in the solution explorer and select “Add New Item” from the “Add” submenu. 2. Select the Consumer Web Part template from the Local Project Items category. 3. Name the new file “SpecialOfferDetails.cs”
2. Remove unnecessary default code	<ol style="list-style-type: none"> 1. Remove the “DefaultProperty” attribute from the SpecialOffersDetails class. 2. Remove the “defaulttext” string constant. 3. Remove the “text” private string member variable. 4. Remove the “Text” property and its associated attributes.
3. Alter default web part communication interface definition code	<ol style="list-style-type: none"> 1. Change the “cellName” private string member variables string value to the value “SpecialOfferID”. 2. Update the call to RegisterInterface within the EnsureInterfaces method by changing the menuLabel parameter from “Consumes a cell from” to “Consumes a Special Offer ID from”. 3. Also update the description parameter from “Consumes a cell of data” to “Consumes the ID of a Special Offer Promotion”. 4. Also update the interfaceName parameter from “CellConsumer_WPQ_” to “SpecialOfferConsumer_WPQ_” 5. Update interfaceClientReference parameter from “CellConsumer_WPQ_” to “SpecialOfferConsumer_WPQ_” 6. Find the PartCommunicationConnect method and replace the string constant “CellConsumer_WPQ_” with “SpecialOfferConsumer_WPQ_” 7. Find the GetInitEventArgs method and replace the string constant “CellConsumer_WPQ_” with “SpecialOfferConsumer_WPQ_”
4. Add a property to track the currently selected Offer ID in ViewState	<ol style="list-style-type: none"> 1. Paste the code contained within the supplied snippet file “<i>300.4.2.SpecialOfferDetails_OfferID.txt</i>” into the class.
5. Add an instance of the	<ol style="list-style-type: none"> 1. Paste the code contained within the supplied snippet file



<p>GridUtil class, and initialize it</p>	<p>“300.4.2.SpecialOfferDetails_GridUtil.txt” into the class.</p> <p><i>Note: This helper class handles the details of databinding, paging, and sorting of grids as well as applying a consistent “look and feel” to the grids in our web parts. We create the DataGrid itself as well as the GridUtil in the web part’s OnInit method because this is the point where we need to create any objects that have postback event handlers attached to them.</i></p>
<p>6. Add the event handler to respond to incoming data from a connected provider web part</p>	<ol style="list-style-type: none"> 1. Find the CellReady event handling method. 2. Replace the contents of this method with the code contained within the supplied snippet file “300.4.2.SpecialOfferDetails_CellReady.txt” <p><i>Note: This method is called by the SharePoint Web Part framework every time a connected provider web part transmits new cell data to the consumer web part.</i></p> <p><i>In this case, we simply update our local OfferID property and reset the SQL query used during data binding.</i></p>
<p>7. Add code to DataBind and display the currently selected row during the web part’s PreRender stage</p>	<ol style="list-style-type: none"> 1. Paste the code contained within the supplied snippet file “300.4.2.SpecialOfferDetails_PreRender.txt” <p><i>Note: Data binding is postponed until the PreRender stage in case we need to alter the source query that drives the data grid in our web part.</i></p>
<p>8. Render the Web Part contents</p>	<ol style="list-style-type: none"> 1. Find the RenderWebPart method. 2. Replace the call to “output.Write()” with “GridUtil.RenderToWebPart(output)” 3. Save the file. <p><i>Note: The GridUtil class will output a simple replacement string if there is no results data to display.</i></p>
<p>9. Build the Web Part project</p>	<ol style="list-style-type: none"> 1. Select “Build Level300LabCs” from the Build menu. 2. Verify that the project is compiled successfully. <p>** If there are any build errors, compare your code to the contents of the completed source code file included in “300.4.2.SpecialOfferDetails.cs.txt”</p>
<p>10. Create a DWP file for the web part installation</p>	<ol style="list-style-type: none"> 1. Right-click the project in the solution explorer and select “Add New Item” from the “Add” menu.



<p>information</p>	<ol style="list-style-type: none"> 2. Select the Web Part Dwp template from Local Project Items. 3. Name the new file “SpecialOfferDetails.dwp” and click the “Open” button to create and open the dwp file. 4. Replace the contents of the new file with the contents contained within the supplied snippet file “300.4.2.SpecialOfferDetails.dwp.txt”. 5. Update the PublicKeyToken value with the actual token extracted in the previous step (example “9c3ec2f3cb594a8b”). 6. Save the file.
<p>11. Install the Web Part into SharePoint</p>	<ol style="list-style-type: none"> 1. Open Internet Explorer and navigate to the root of the Lab portal at http://localhost/hol300/default.aspx 2. Click “Modify Shared Page” and then “Import” from the “Add Web Parts” submenu. 3. Browse to the DWP file that was created in an earlier step which should be located at “C:\Projects\Sharepoint123\Level300LabCs\SpecialOfferDetails.dwp” 4. Click the “Upload” button to upload the web part definition into SharePoint. 5. Verify that there is now a “Special Offer Details” entry listed under “Uploaded Web Part”. 6. Drag the uploaded web part from the “Uploaded Web Part” section into the “Left” web part zone. <p><i>If all steps have been performed correctly, the Special Offer Details web part should now be displayed. It will not display any records yet however, since it hasn't been connected to a provider web part.</i></p>
<p>12. Connecting the Consumer Web Part to a Provider</p>	<ol style="list-style-type: none"> 1. While still in Page Design Mode, click the small down-arrow icon in the header bar for the Special Offer Details web part that was just added to the page. 2. Because this is a connectable web part, a submenu titled “Connections” will be available. 3. Click the “Connections” submenu to display the list of available connections to this web part. <p><i>We only implemented one connection interface, so only one option will be given. We could have implemented a number of incoming and outgoing connection interfaces if we had needed to.</i></p> <ol style="list-style-type: none"> 4. Select the “Consumes a Special Offer ID from” option. <p><i>Note that this is the value we provided for the “menuLabel” parameter in the call to RegisterInterface earlier.</i></p>



	<p><i>SharePoint will then display a list of available Provider interfaces. Since we currently only have one Provider web part on the page at this time, only that one option will be presented.</i></p> <ol style="list-style-type: none">5. Select the “Special Offer List” option to complete the web part connection process.6. Click the “Home” link at the upper left of the portal page to quickly exit the customization mode.7. Sorting and Paging should work in the new web part.8. Selecting a new record in the Special Offer List connected web part should affect this new web part accordingly.
--	--



EXERCISE 3: CREATING A SECOND CONSUMER WEB PART

Scenario

The power on connectible web parts lies in the flexible architecture provided by SharePoint. While Providers and Consumers are aware of the actual structures and data elements being passed through connections, they do not necessarily care which remote web parts are being conversed with (or how many).

In this example, we will build a second Consumer Web Part that listens for selected Offers and displays the list of Orders that took advantage of that promotion.

This exercise is strikingly similar to the previous exercise, with only minor differences in the corresponding code. This purposefully illustrates how simple developing connected web parts can be, once you master the key concepts and gain a little practice.

Tasks	Detailed Steps
1. Add the new Web Part using the template wizard	<ol style="list-style-type: none"> 1. Right-click the Project in the solution explorer and select “Add New Item” from the “Add” submenu. 2. Select the Consumer Web Part template from the Local Project Items category. 3. Name the new file “SpecialOfferOrders.cs”
2. Remove unnecessary default code	<ol style="list-style-type: none"> 1. Remove the “DefaultProperty” attribute from the SpecialOfferOrders class. 2. Remove the “defaulttext” string constant. 3. Remove the “text” private string member variable. 4. Remove the “Text” property and its associated attributes.
3. Alter default web part communication interface definition code	<ol style="list-style-type: none"> 1. Change the “cellName” private string member variables string value to the value “SpecialOfferID”. 2. Update the call to RegisterInterface within the EnsureInterfaces method by changing the menuLabel parameter from “Consumes a cell from” to “Consumes a Special Offer ID from”. 3. Also update the description parameter from “Consumes a cell of data” to “Consumes the ID of a Special Offer Promotion”. 4. Update the interfaceName parameter from “CellConsumer_WPQ_” to “SpecialOfferOrder_WPQ_” 5. Update the interfaceClientReference parameter from “CellConsumer_WPQ_” to “SpecialOfferOrder_WPQ_” 6. Find the PartCommunicationConnect method and replace the string constant “CellConsumer_WPQ_” with “SpecialOfferOrder_WPQ_” 7. Find the GetInitEventArgs method and replace the string



	constant "CellConsumer_WPQ_" with "SpecialOfferOrder_WPQ_"
4. Add a property to track the currently selected Offer ID in ViewState	1. Paste the code contained within the supplied snippet file "300.4.3.SpecialOfferOrders_OfferID.txt".
5. Add an instance of the GridUtil class, and initialize it	1. Paste the code contained within the supplied snippet file "300.4.3.SpecialOfferOrders_GridUtil.txt" <i>Note: This helper class handles the details of databinding, paging, and sorting of grids as well as applying a consistent "look and feel" to the grids in our web parts. We create the DataGrid itself as well as the GridUtil in the web part's OnInit method because this is the point where we need to create any objects that have postback event handlers attached to them.</i>
6. Add the event handler to respond to incoming data from a connected provider web part	1. Find the CellReady event handling method. 2. Replace the contents of this method with the code contained within the supplied snippet file "300.4.3.SpecialOfferOrders_CellReady.txt" <i>Note: This method is called by the SharePoint Web Part framework every time a connected provider web part transmits new cell data to the consumer web part.</i> <i>In this case, we simply update our local OfferID property and reset the SQL query used during data binding.</i>
7. Add code to DataBind and display the currently selected row during the web part's PreRender stage	1. Paste the code contained within the supplied snippet file "300.4.3.SpecialOfferOrders_PreRender.txt" <i>Note: Data binding is postponed until the PreRender stage in case we need to alter the source query that drives the data grid in our web part.</i>
8. Render the Web Part contents	1. Find the RenderWebPart method. 2. Replace the call to "output.Write()" with "GridUtil.RenderToWebPart(output)" 3. Save the file. <i>Note: The GridUtil class will output a simple replacement string if there is no results data to display.</i>
9. Build the Web Part project	1. Select "Build Level300LabCs" from the Build menu. 2. Verify that the project is compiled successfully.



	<p>** If there are any build errors, compare your code to the contents of the completed source code file included in "300.4.3.SpecialOfferOrders.cs.txt"</p>
<p>10. Create a DWP file for the web part installation information</p>	<ol style="list-style-type: none"> 1. Right-click the project in the solution explorer and select "Add New Item" from the "Add" menu. 2. Select the Web Part Dwp template from Local Project Items. 3. Name the new file "SpecialOfferOrders.dwp" and click the "Open" button to create and open the dwp file. 4. Replace the contents of the new file with the contents contained within the supplied snippet file "300.4.3.SpecialOfferOrders.dwp.txt". 5. Update the PublicKeyToken value with the actual token extracted in the previous step (example "9c3ec2f3cb594a8b"). 6. Save the file.
<p>11. Install the Web Part into SharePoint</p>	<ol style="list-style-type: none"> 1. Open Internet Explorer and navigate to the root of the Lab portal at http://localhost/hol300/default.aspx 2. Click "Modify Shared Page" and then "Import" from the "Add Web Parts" submenu. 3. Browse to the DWP file that was created in an earlier step which should be located at "C:\Projects\Sharepoint123\Level300LabCs\SpecialOfferOrders.dwp" 4. Click the "Upload" button to upload the web part definition into SharePoint. 5. Verify that there is now a "Special Offer Orders" entry listed under "Uploaded Web Part". 6. Drag the uploaded web part from the "Uploaded Web Part" section into the "Right" web part zone. <p><i>If all steps have been performed correctly, the Special Offer Orders web part should now be displayed. It will not display any records yet however, since it hasn't been connected to a provider web part.</i></p>
<p>12. Connecting the Consumer Web Part to a Provider</p>	<ol style="list-style-type: none"> 1. While still in Page Design Mode, click the small down-arrow icon in the header bar for the Special Offer Orders web part that was just added to the page. <p><i>Because this is a connectable web part, a submenu titled "Connections" will be available.</i></p> <ol style="list-style-type: none"> 2. Click the "Connections" submenu to display the list of available connections to this web part.



	<p><i>We only implemented one connection interface, so only one option will be given. We could have implemented a number of incoming and outgoing connection interfaces if we had needed to.</i></p> <p>3. Select the “Consumes a Special Offer ID from” option.</p> <p><i>Note that this is the value we provided for the “menuLabel” parameter in the call to RegisterInterface earlier.</i></p> <p><i>SharePoint will then display a list of available Provider interfaces. Since we currently only have one Provider web part on the page at this time, only that one option will be presented.</i></p> <p>4. Select the “Special Offer List” option to complete the web part connection process.</p> <p>5. Click the “Home” link at the upper left of the portal page to quickly exit the customization mode.</p> <p>6. Sorting and Paging should work in the new web part.</p> <p>7. Selecting a new record in the Special Offer List connected web part should affect this new web part accordingly.</p>
--	--



LESSON 5: SHAREPOINT WEB SERVICES

Objectives	After completing this lab, you will that understand that SharePoint has web services exposed for developers to use in applications. <ol style="list-style-type: none">4. Create a windows app that pulls all the catalog data via ADO.net5. Upload each item into a list in SharePoint via Web Services6. * attachment of images
Prerequisites	It is important that you understand the concepts of what SharePoint is and how to develop for SharePoint. You will be well prepared if you have completed the 100 level and 200 level Hands on Labs material before attempting this lesson.
Setup	The lab machine requires the following software be pre-installed before beginning this lab: <ul style="list-style-type: none">• Windows Server 2003 + Service Pack 1<ul style="list-style-type: none">○ Install IIS and create an application server.○ Remove FrontPage 2000 server extensions• SQL Server 2000 + Service Pack 4• Microsoft Office 2003 Professional• WSS• Visual Studio 2003• SharePoint templates for Visual Studio

Estimated time to complete this lesson: 30 minutes



EXERCISE 1: USING SHAREPOINT WEB SERVICES

Scenario

For the purposes of this demo, we need to create a Windows forms application that pulls the bicycles out of the AdventureWorks database. We will be using ADO.Net to accomplish this task. Once we have the catalog data in our Winform application we will be publishing the data to a new SharePoint list.

Tasks	Detailed Steps
1. Create a new, blank C# project in our Visual Studio solution	<ol style="list-style-type: none"> 1. Open Visual Studio .NET 2003 and Create or Open the Solution named C:\Projects\SharePoint123 2. Right-click the solution in the Solution Explorer, and select “New Project” from the “Add” menu. 3. Select C# Projects → Windows application 4. Give your project the name HOL300AdvWorks and save it in the C:\Projects folder 5. A new, blank C# project is created for you with a blank Form1 ready for your code
2. Add a web reference to the Lists	<ol style="list-style-type: none"> 1. Right-click on the References folder in the Solution Explorer window 2. Select Add Web Reference 3. In the dialog that pops up, click the link Web services on this local machine 4. Click Lists in the list that pops up 5. In the web reference name field, type “SharePoint.WebServiceModel” 6. Click the Add Reference button to create the web reference proxy class for the Lists web service 7. Select your SharePoint.WebServiceModel web reference in the Solution Explorer 8. In the Properties window, change the URL behavior to Dynamic. <p><i>This causes the application to read app.config to find the web reference URL</i></p>
3. Add a reference in the app.config to the SharePoint.WebServiceModel	<ol style="list-style-type: none"> 1. Open app.config by double clicking it in the Solution Explorer 2. Replace the contents of this file with the contents of the code snippet 300.5.1.AppConfig.txt. <p><i>This config changes the Lists web reference URL to point to our HOL300 team site.</i></p>



LESSON 5: SHAREPOINT WEB SERVICES

<p>4. Add a drop down list, list box, and button to our form</p>	<ol style="list-style-type: none"> 1. Open the Form1.cs form by double clicking it in the Solution Explorer 2. Switch to the code view by clicking View→Code (F7) in the menu bar 3. Replace the contents of this file with the code snippet 300.5.1.EntireForm.txt. <p><i>This snippet adds the form elements in their proper locations</i></p>
<p>5. Wire up the form’s constructor to wire the drop down list to the AdventureWorks database</p>	<ol style="list-style-type: none"> 1. Locate the constructor for Form1 (public Form1()) and replace it with the contents of 300.5.1.Constructor.txt <p><i>This snippet populates the drop down list with a list of bicycle categories from the AdventureWorks2000 database</i></p>
<p>6. Add an enumeration of list types to our project</p>	<ol style="list-style-type: none"> 1. Right click on the project HOL300AdvWorks and select Add → Add New Item. 2. Add a class file called Enums.cs and paste the contents of the snippet file 300.5.1.Enums.txt into the new class file.
<p>7. Add an event handler for the drop down list to display all the bicycles of the selected type</p>	<ol style="list-style-type: none"> 1. Copy the contents of the code snippet 300.5.1.BikeSelectChange.txt into the Form1 class. <p><i>This snippet handle’s the combo box’s SelectedIndexChanged event and puts a list of all the appropriate bicycles into the list box</i></p>
<p>8. Add a handler for the button click to upload our listed bicycles to the SharePoint team site</p>	<ol style="list-style-type: none"> 1. Copy the contents of the code snippet 300.5.1.UploadList.txt into the Form1 class. <p><i>This snippet creates a SharePoint list, adds a column to the list, and puts our loaded list of bicycles into this SharePoint list</i></p>
<p>9. Run our application</p>	<ol style="list-style-type: none"> 1. Build and run the application from the debugger by pressing F5 2. Give your SharePoint list a name or accept the default “WebServicesList” 3. Select a type of bike from the drop down list. The listbox will fill with bicycle names 4. Click the upload button to send the list to SharePoint.
<p>10. Navigate to our team site and verify the list was created successfully</p>	<ol style="list-style-type: none"> 1. Browse in IE to http://localhost/hol300 2. Click the Lists hyperlink in the QuickLaunch bar on the left 3. Select the WebServiceList link in the list of SharePoint lists. This list contains the bicycles we selected.

