Best Practices for Developing SharePoint Web Parts

SPDEV370

Todd C. Bleeker, PhD.

Wednesday, 09:55AM-11:45AM

Simulcast Live

Note: Special lunch for attendees will follow this session

Clarity. Direction. Confidence.

BP

BEST PRACTICES SHAREPOINT CONFERENCE

About ME…

- **Todd C. Bleeker, PhD MVP**

CSA for English, Bleeker, and Associates

www.*EBAcompanies.com*

Instructor for Mindsharp

www.*Mindsharp.com*

DevelopersGuide.*Mindsharp.com*

tbleeker@mindsharp.com

facebook.com/toddbleeker

twitter.com/toddbleeker

SharePoint.*MindsharpBlogs.com*/Todd

Pictured here with youngest daughter Lexa

Agenda or Content Slide

- Poll of attendees expertise
- ABCs of Web Parts
- Standard SharePoint Development Lifecycle
- General Web Part Best Practices
- Demo: Visual Web Part

Your Experience

ABCs of Web Parts

- Appearance
  - CSS
  - HTML DOM
- Behavior
  - JavaScript
  - AJAX
  - Connections
- Content
  - SharePoint content database
  - Corporate LOB systems
  - External

Standard Development Lifecycle

01 • **Start** with a well-named VS.NET Library

02 • **Sign** or Strong Name the assembly

03 • **Set** values in the required XML files

04 • **Strike** <Ctrl-Shift-B> to build the solution output

05 • **Slap** the results into SharePoint

06 • **Sharpen** the logic and rendering

07 • **Secure** the assembly (if any) using CAS

08 • **Supply** a Solution deployment Manifest.xml

09 • **Specify** files to include in a Solution CAB

10 • **Store**/Deploy the Solution CAB

Standard Development Lifecycle

01 • **Start** with a well-named VS.NET Library

02 • **Sign** or Strong Name the assembly

03 • **Set** values in the required XML files

04 • **Strike** <Ctrl-Shift-B> to build the output

05 • **Slap** the results into

06 • **Sharpen** the logic and rendering

07 • **Secure** the assembly (if any) using CAS

08 • **Supply** a Solution deployment Manifest.xml

09 • **Specify** files to include in a Solution CAB

10 • **Store**/Deploy the Solution

Plumbing

Packaging

Plumbing

01

- **Start** with a well-named VS.NET Library

- Choose a Library Type:
  - Blank Library
  - Class Library
  - Web Control Library
  - VSeWSS Library
  - STSDEV (Class) Library
  - Other Community (CodePlex) Libraries

Choose a "Good" Name

- Meaningful
- Unique on the Internet
- CompanyProject.Contents.Category
- Used as default:
  - Solution Name
  - Project Name
  - Folder Name
  - Default Assembly Name
  - Default Namespace Name
- Example
  - Mindsharp.WebParts.Public

Library Pros and Cons

Factor	Blank	Web Control	Class	VSeWSS
x86/x64 support?	✓	✓	✓	✗
Useful defaults?	✗	✗	✓	✗
Nothing to cleanup?	✓	✗	✓	✗
Appropriate references?	✗	✗	✗	✗
XCopy deployment?	✗	✓	✓	✗
WSP CAB generation?	✗	✓	✓	✓
Customizable post-build events?	✗	✓	✓	✗

Library Pros and Cons

Factor	Blank	Web Control	Class	VSeWSS
Development consistency?	✓	✓	✓	✗
Foldering flexibility?	✓	✓	✓	✗
Multi-artifact solutions?	◉	✓	✓	✓
Upgradable to vNext?	✓	✓	✓	◉

Start with a Class Library

- A Class Library:
  - Works in all environments, today and tomorrow
  - Can be used consistently for all development projects
  - Supports both XCopy and WSP CAB deployment
- VSeWSS 1.3 brings significant improvements
- Adopt VS.NET 10 for SharePoint dev ASAP
- 'Round the clock housekeeping required:
  - Add reference to System.Web and WSS, if needed
  - Rename class (auto-refactor code)
  - Add required class constructors
  - Don’t forget to scope added classes

Common Web Part Suffixes

- In the Project Root or Foo folder:
  - Web Part Class: **`FooPart.cs`**
- wpcatalog folder:
  - *.webpart File: **`FooPart.webpart`**
- wpresources folder:
  - All external resources
  - Also consider embedding WebResources
- TEMPLATE\CONTROLTEMPLATES folder:
  - User Controls*: **`FooPartControl.ascx`**

General Project Organization

- Uncheck Create directory for solution
- Project folders only one level deep
- Solution project shells for asset projects
  - Allows for mix and match
  - Good for source management
- wpcatalog folder for *.webpart files
- wpresources folder for external resources
- script/image/etc folders for embedded resources

.NET Web Part vs. WSS Web Part

.NET Web Part

- Runs on any website
- Future of Web Part development
- No cross-page connections
- No connecting Web Parts that aren't in zones
- No client-side connections

SharePoint Web Part

- Only runs on WSS sites
- Primarily available for backward compatibility
- Includes cross-page connections
- Allows connecting Web Parts that aren't in zones
- Supports client-side connections

Inherit from .NET Web Part

- A .NET Web Part:
  - Works in all environments, today and tomorrow
  - Used consistently for all Web Part projects
  - Works in all ASP.NET projects, not just SharePoint

- Housekeeping required:
  - Derive from

`System.Web.UI.WebControls.WebParts.WebPart`

CreateChildControls

- Initially output DateTime.Now.ToString()
- Never use Render
- Never use RenderControl
- Rarely use RenderContents

Plumbing

02

- **Sign** or Strong Name the assembly

- GAC rhymes with Flack, Hack, Sack, Smack, Whack; but operations should dictate deployment location. So, the Web Part must be signed.
- Sign the assembly using VS.NET
- No Password on SNK files
- Compile to embed Public Key Token
- Never deploy to _app_bin (it's a Full-trust code gen folder) SharePoint Designer won't find it there either

The GAC Isn't Crap

- First place the .NET Framework looks
- Pre-checked for tampering
- Can run multiple Version + Culture + PublicKeyToken versions of an assembly
- Always participates in CAS
- Always runs under Full Trust (pros and cons)
- Cached: Runs a shadow copy of the assembly (No DLL hell)
- IISRESET required to change

BIN rhymes with WIN

- Configured to run under WSS_Minimal Trust
- No need to recycle the Application Pool if the assembly is deployed to the bin
- Fastest iterative approach
  - Code
  - Compile
  - Refresh
- If your signed assembly runs in the BIN,
  it will likely run in the GAC,
  the opposite is not true (typically due to CAS)

Setup Get Public Key Option

- Tools > External Tools…
- Click the **Add** button
  - Title: *Get &Public Key*
  - VS.NET 2005 Command: *C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin\sn.exe*
  - VS.NET 2008 Command: *C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin\sn.exe*
  - Arguments: *-Tp "$(TargetPath)"*
  - Select **User Output Window** checkbox
  - Click the **OK** button to save
- Ensure project has focus before selecting option

Signing Best Practices

- Move SNK to Properties folder
- Inspect/Alter AssemblyInfo Class
- Chevy Chase Look to eliminate dynamic versioning: Version 1.0.*
- Set assembly directive (Yikes!):

```
System.Security.AllowPartiallyTrustedCallers()
```

SNK Management Options

- Developer
  - Each developer has their own key
  - Embed the key in VS.NET for development
  - Delay Signing
- Project
  - Each project has their own key
- Keys in development cannot be use in Production
- All code runs thru gatekeeper for deployment
- Setup a handful of permutations representing common CAS levels that developers can assign

Plumbing

03

- **Set** values in the required XML files

- In addition to the class, *.webpart is required
- Use 12 Hive to Organize
- Exposes Web Part to SharePoint
- importErrorMessage required Title not required
- AllowClose to False
- CatalogIconImageUrl
- Be sure to use the correct PublicKeyToken
- Assembly on five lines, Properties on one line

Use 12 Hive to Organize

- Flexible folder structure – a place for everything and everything in its place
- Easier to deploy (both XCopy and CAB)
- Anticipate others placement of project assets
- Supports the creation of large, complex solutions
- Interfaces with community tools

Plumbing

04

- **Strike** <Ctrl-Shift-B> to build the solution

- XCopy deploy using one of the following:
  - Post-build events
  - Targets file
  - SDK deployment files
- Use MakeCab for creating WSP CAB
  - Manifest.xml
  - WSP.ddf
- SafeControl entry required for Web Part assembly

CAB Project vs. MakeCab.exe

CAB

- VS.NET Project Type
- Assets from projects can be tagged for inclusion
- Can only be used for Web Part projects
- Only outputs CAB
- No predefined limit

MakeCab.exe

- Command line tool
- Assets must be identified by name
- Used for all projects, including Web Parts
- Outputs CAB or WSP
- Defaults to 360K

Post Build vs. Targets

Post Build

- Developer environment
- Simple: Nine commands
- Easy to modify on the fly
- Defined in the project file
- May need to REM out before check-in

Targets

- Build environment
- Complex implementation
- Requires planning
- Defined in its own file
- Rarely modified

XCopy Commands

- **For developers only**, simple commands:

```
:: Change directory to the root of the project
cd "$(ProjectDir)"

:: Recycle the application pool
%systemroot%\system32\iisapp.vbs /a "SharePointAppPool" /r

:: Copy all files from the project's 12 folder to 12 Hive
xcopy "12" "%CommonProgramFiles%\Microsoft Shared\
    web server extensions\12\" /ys

:: Copy all files from the project's 80 folder to Web Application home directory
xcopy "80" "C:\Inetpub\wwwroot\wss\VirtualDirectories\[Site]\" /ys
```

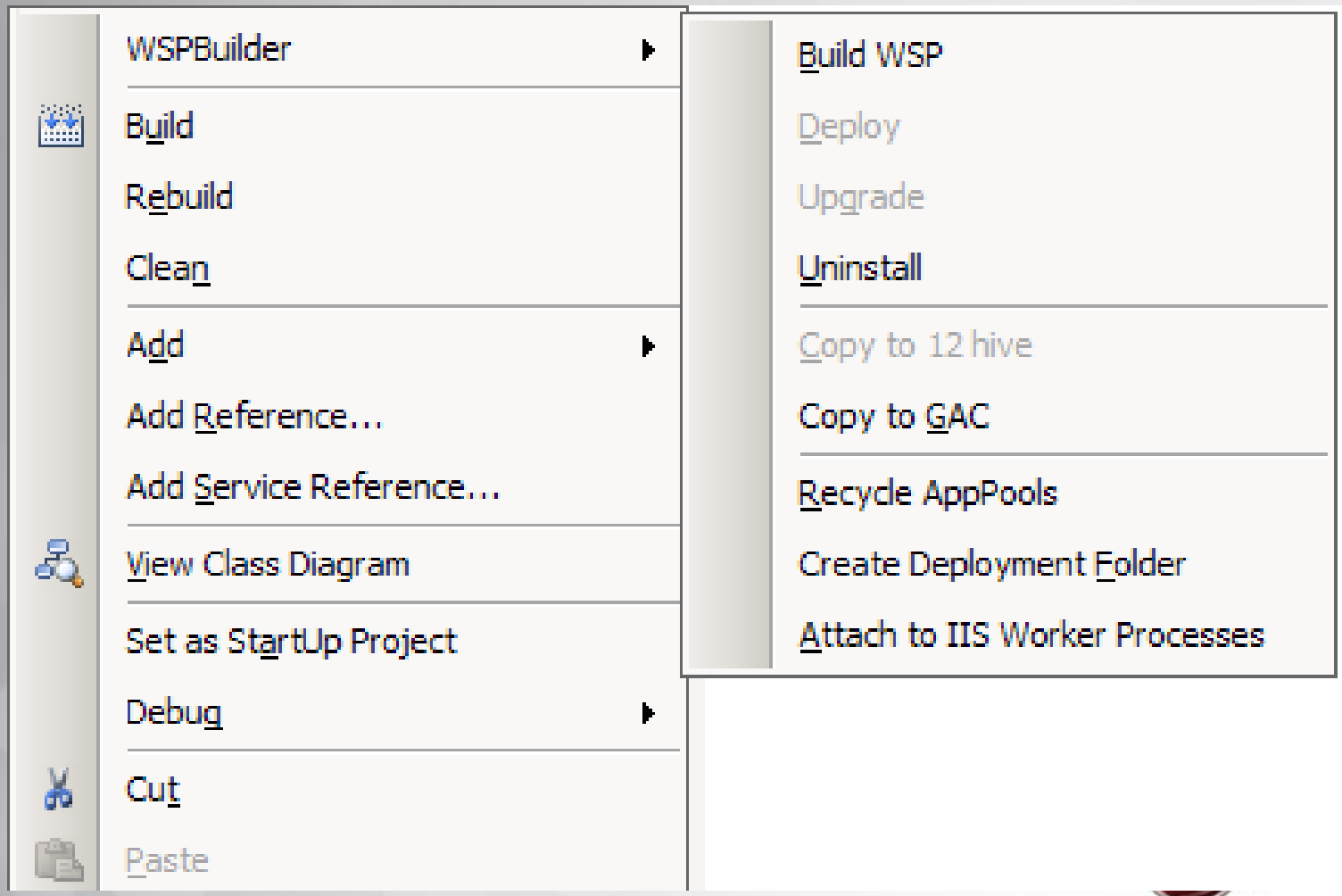
XCopy Commands

- Continued:

```
:: Copy DLLs to the BIN
xcopy "$(TargetDir)*.dll"
    "C:\Inetpub\wwwroot\wss\VirtualDirectories\[Site]\bin\" /ys

:: Install Force DLLs to the GAC (VS.NET 2005)
"%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\GacUtil.exe" /if
    "$(TargetPath)"

:: Install Force DLLs to the GAC (VS.NET 2008)
"%ProgramFiles%\Microsoft SDKs\Windows\v6.0A\Bin\GacUtil.exe" /if
    "$(TargetPath)"

:: Create a WSP CAB
MakeCAB /f "WSP.DDF"
```

Install WSPBuilder!

- Install WSPBuilder into your environment, TODAY



Plumbing

05

- **Slap** the results into SharePoint

- Plumb the Web Part with a bare bones initial solution (output DateTime)
- New Up the Web Part
- Web Application wpcatalog (Solution) vs. Site Collection Web Part Gallery (Solution/Feature)
- Install/Activate Feature
- Add the Web Part to a test page

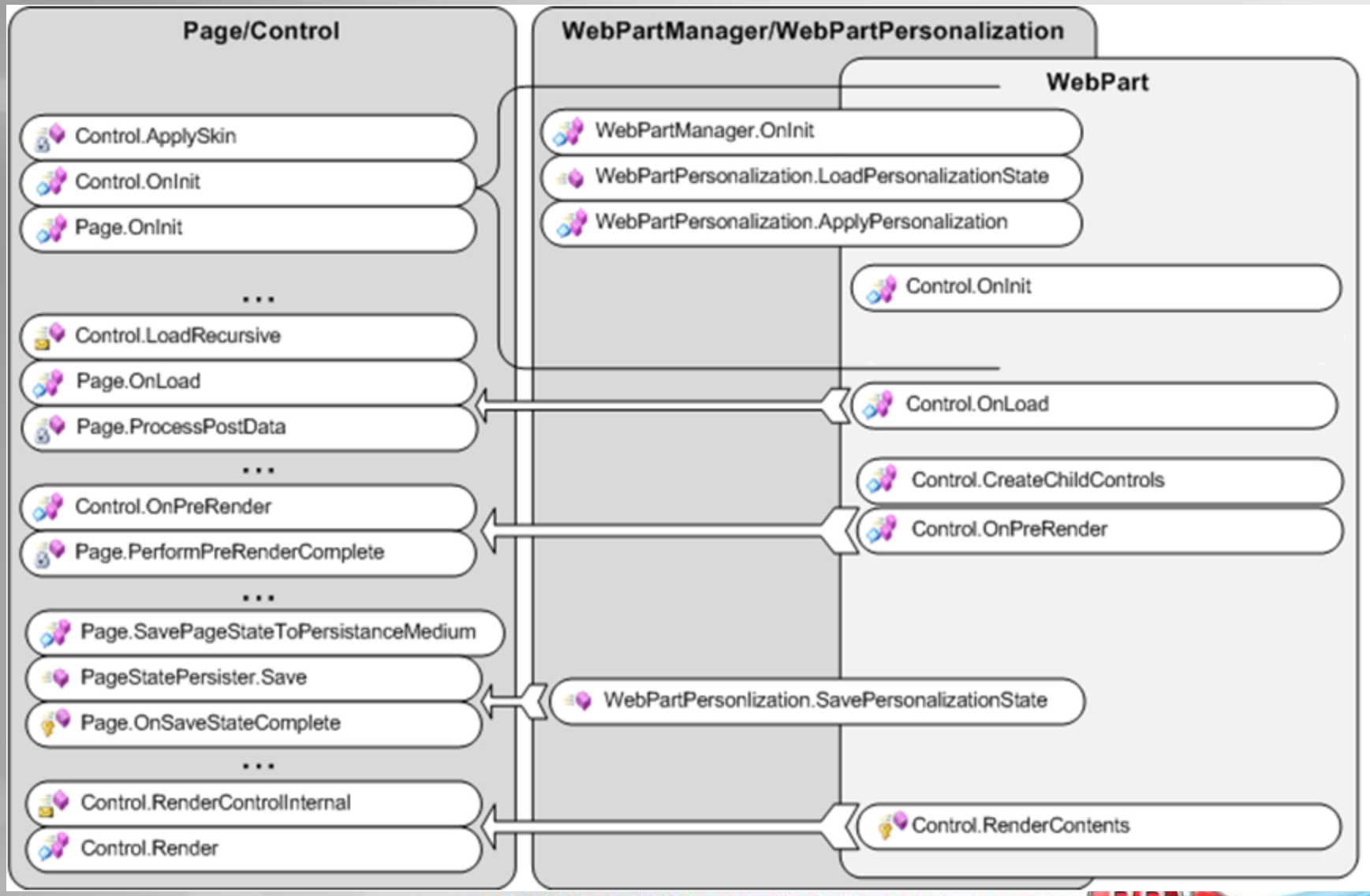
Sharpen the Logic and Rendering

06

- **Sharpen** the logic and rendering

- Two schools of thought on overriding methods
  - Page = Proxy Methods
  - Part = Direct Methods
- Update CreateChildControls()
- Add OnInit()
- Add OnLoad()
- Add OnPreRender()
- Add RenderContents()

Web Part Life Cycle



OnInit

- Use to initialize objects that would live for the duration of the life cycle
- Setup connection strings
- Page is not yet available
- Check for IsPostback and IsCallback

- Redirect forced move to OnPreRender

OnLoad

- Query the database (asynchronously is ideal)
- Load Datasets
- Use ClientScriptManager to inject external and embedded CSS and JavaScript
- Check for IsPostback and IsCallback

- Again, redirect forced move to OnPreRender

CreateChildControls

- Create User Interface structure as Server Control
- Four "Eye"s:
  - Instantiate
  - Initialize
  - wIre-up
  - Insert (Add)
- Create **Visual Web Parts** by moving the UI to a User Control and use .NETs **LoadControl()**
- May be called out of sequence using EnsureChildControls()

User Control vs. Server Control

Factor	User Control	Server Control
Documented in WSS SDK?	✗	✓
Great Intellisense?	✓	⊙
WYSIWYG Editing?	✓	✗
Manipulate programmatically?	✓	✓
Organize project using 12 Hive?	✓	✓
Easy to consume in Web Part?	✓	✓

User Control vs. Server Control

Factor	User Control	Server Control
Easy for the junior dev?	✓	✗
Can be debugged?	✓	✓
FindControl unnecessary?	✓	✗
In Custom folder?	✓	✗
More than one can be used?	✓	✗

Visual Web Parts

- Whenever possible, move your user interface to a User Control
- User Control's code beside and designer classes are compiled into the Web Part's DLL
- Create strongly typed variable to the User Control within the Web Part
- Create strongly typed variable to the Web Part within the User Control

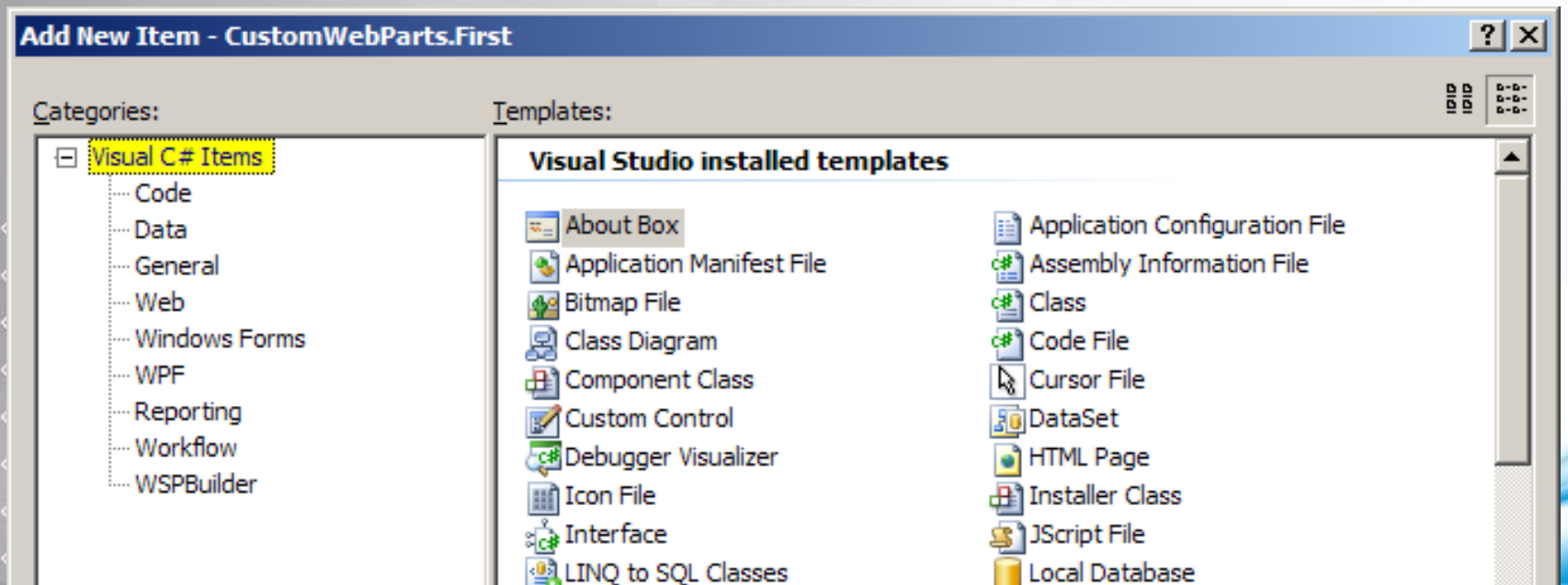
Add VS.NET Web Item Templates

- Add the following to your *.csproj file:

<ProjectTypeGuids>{349c5851-65df-11da-9384-00065b846f21};
{**fae04ec0-301f-11d3-bf4b-00c04f79efbc**}</ProjectTypeGuids>

- Add the following to your *.vbproj file:

<ProjectTypeGuids>{349c5851-65df-11da-9384-00065b846f21};
{**f184b08f-c81c-45f6-a57f-5abd9991f28f**}</ProjectTypeGuids>



OnPreRender

- Last Opportunity to influence the View State that will be sent to the client
- Move OnInit and OnLoad code to this event when the code may be run unnecessarily

RenderContents

- Only use RenderContents to update user interface for programmatically set properties
- The base class essentially calls:
  - EnsureChildControls()
  - RenderChildren()

Secure the Web Part using CAS

07

- **Secure** the assembly using CAS

- Similar to User Access Security
- This is a matter of TRUST
- Most attacks come from within

- For Web Part CAS details, see http://tinyurl.com/SharePointCAS

Code Access Security

- CAS IS NOT HARD
- CAS IS NOT HARD
- CAS IS NOT HARD
- CAS IS NOT HARD
- CAS IS NOT HARD
- CAS IS NOT HARD
- CAS IS NOT HARD

Code Access Security

- Use CAS it IS NOT HARD
- Test Web Parts using Anonymous and Readers
- Add to or create a custom CAS policy; Consider implementing half a dozen permutations
- Deploy CAS using a WSP CAB
- Use .NET Framework 2.0 Configuration Wizard to generate:
  - SecurityClass (Condition, Permission, and Construct)
  - NamedPermissionSet
  - CodeGroup

Packaging

08 • **Supply** a Solution deployment Manifest.xml

09 • **Specify** files to include in a Solution CAB

10 • **Store**/Deploy the Solution CAB

- Use community tools like WSPBuilder to generate the Manifest.xml and WSP.ddf
- ALWAYS use a WSP Solution CAB for deployment into production
- For packaging details, see other talks this week

Solution Deployment

- Add a Manifest.xml file to the VS.NET project
- Provide an inventory of files that will be in the CAB
- Maximize use of the RootFiles tag
- Utilize the DwpFiles tag for *.webpart files

Web Part Features

- Pros for Web Part Features
  - Only way to "group" Web Part in Add dialog
  - Only way to permission Web Parts
  - May be activated by end users on decentralized Site Collections rather than centrally on Web Applications
- Cons for Web Part Features
  - Orphaned in Web Part Gallery on deactivation
  - Must be activated by end users on decentralized Site Collections rather than centrally on Web Applications

Avoid Web Part Features*

- Only need a WSP CAB , not a Feature, to deploy a custom Web Part
- Feature is a tremendous overhead to provide the three pros listed on the previous slide

*This is a minority opinion

Use Web Part Properties

- State Management
- Finite Presentation/Validation
  - String (textbox)
  - Integer (textbox)
  - DateTime (textbox not calendar)
  - Boolean (checkbox)
  - Color (dropdown list)
  - Enumeration (dropdown list)
- Property Builders
- Personalization Management
- Categories

Use Editor Parts

- Validate User Input
- Abstraction Layer
- Custom User Interface Presentation
  - Password
  - Calendar
  - Dependant Lists

Use Web Part Connections

- Custom Interface
- IWebPartTable
- IWebPartRow*
- IWebPartCell
- IWebPartFilter

- Give each connection a unique ID, don’t use the default ID called "Default"
- Leverage Transformers

*Most Transformable

Test, Test, Test Web Parts

- Anonymous and Reader Users
- Code Access Security (CAS)
- Test Connections
- Check out the details for how to test these in this MSDN article:

  msdn.microsoft.com/en-us/library/ms916830.aspx

  www.21apps.com/agile/beginners-guide-to-test-driven-web-part-development/

Other Considerations

- Verbs = Embedded functionality
- Web Part Cache - Who doesn't love cache
  - Can substantially improve performance
  - Use for non-volatile, frequently accessed, finite data that can easily fit into memory
- Call EnsureChildControls() before using child controls
- Customization (shared) vs. Personalization (individual)
- Use Properties to avoid hard coded values
- HTMLEncode everything that the user enters when you render it out to prevent script/SQL injection hacks

Other Considerations

- Don't build your entire application in a single Web Part
- Do build solutions that can be added to SharePoint nearly anywhere

Want More?

- Get your SharePoint project OnPath™ with Mindsharp's unique educational approach

- Get Todd's SharePoint v3 (2007) Developer Training: http://www.Mindsharp.com/?top=TRAINING

- Get Todd's SharePoint Developer book: http://www.amazon.com/dp/1584505001

DEMO

VISUAL WEB PART

Use a User Control as a strongly typed, tightly coupled design surface that manages the user interface (UI) for your Web Part

Thank you for attending!

Please fill out your evaluation and turn it in on the back table!

Did you miss something? There will be a Conference DVD shipped directly to you, sponsored by HIGH MONKEY CONSULTING

Clarity. Direction. Confidence.