



Building AJAX-Based Web Applications with ArcGIS Server and .NET

Art Haddad and Rex Hansen

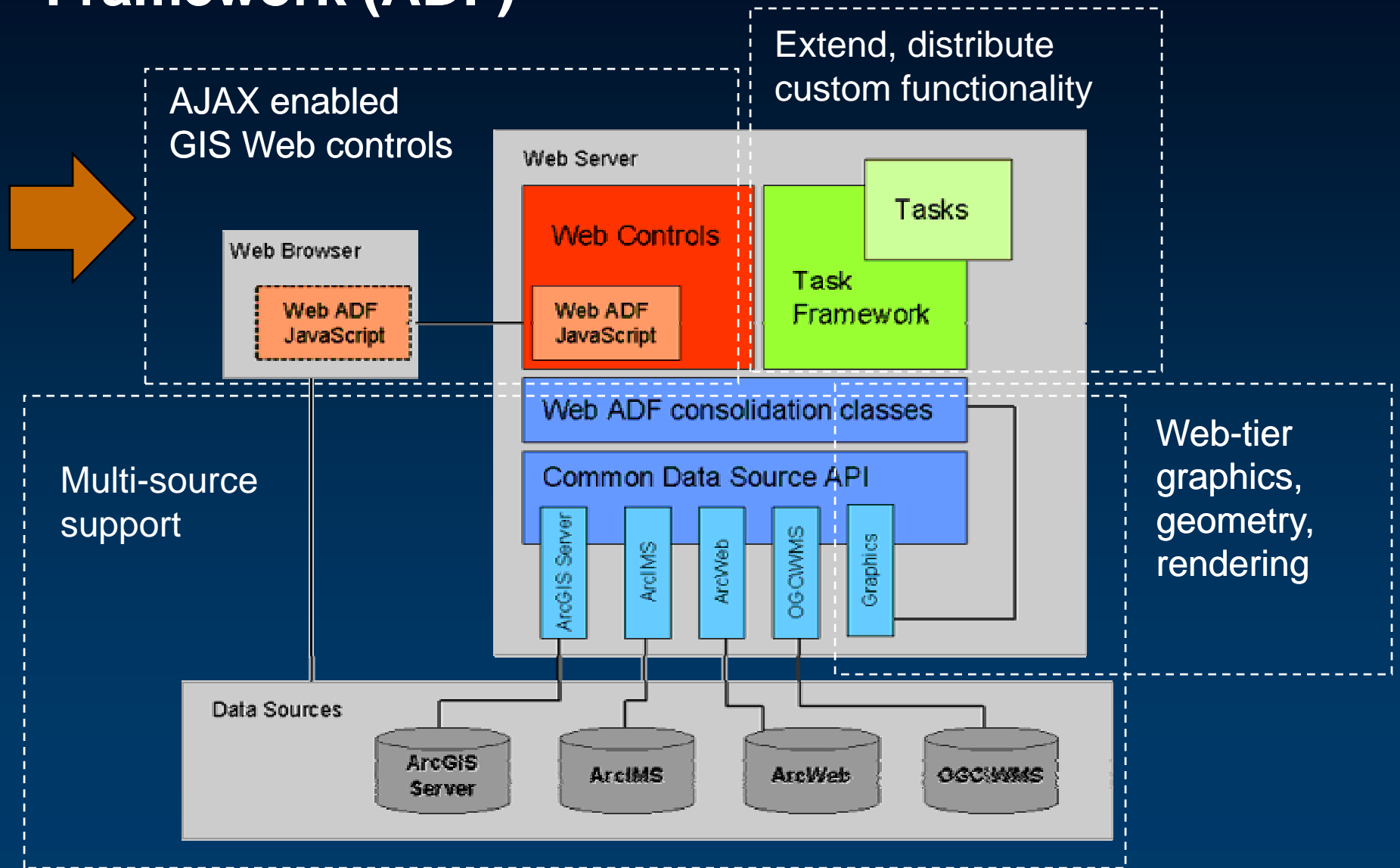
Session Topics

- **ASP.NET and AJAX**
- **Web ADF and AJAX**
 - Tools and Commands
 - Controls and Tasks
 - Working with Callbacks and Results
- **“Mash-ups” with the Web ADF**
- **Best Practices**
 - Building AJAX Web Applications
 - Web Application Design

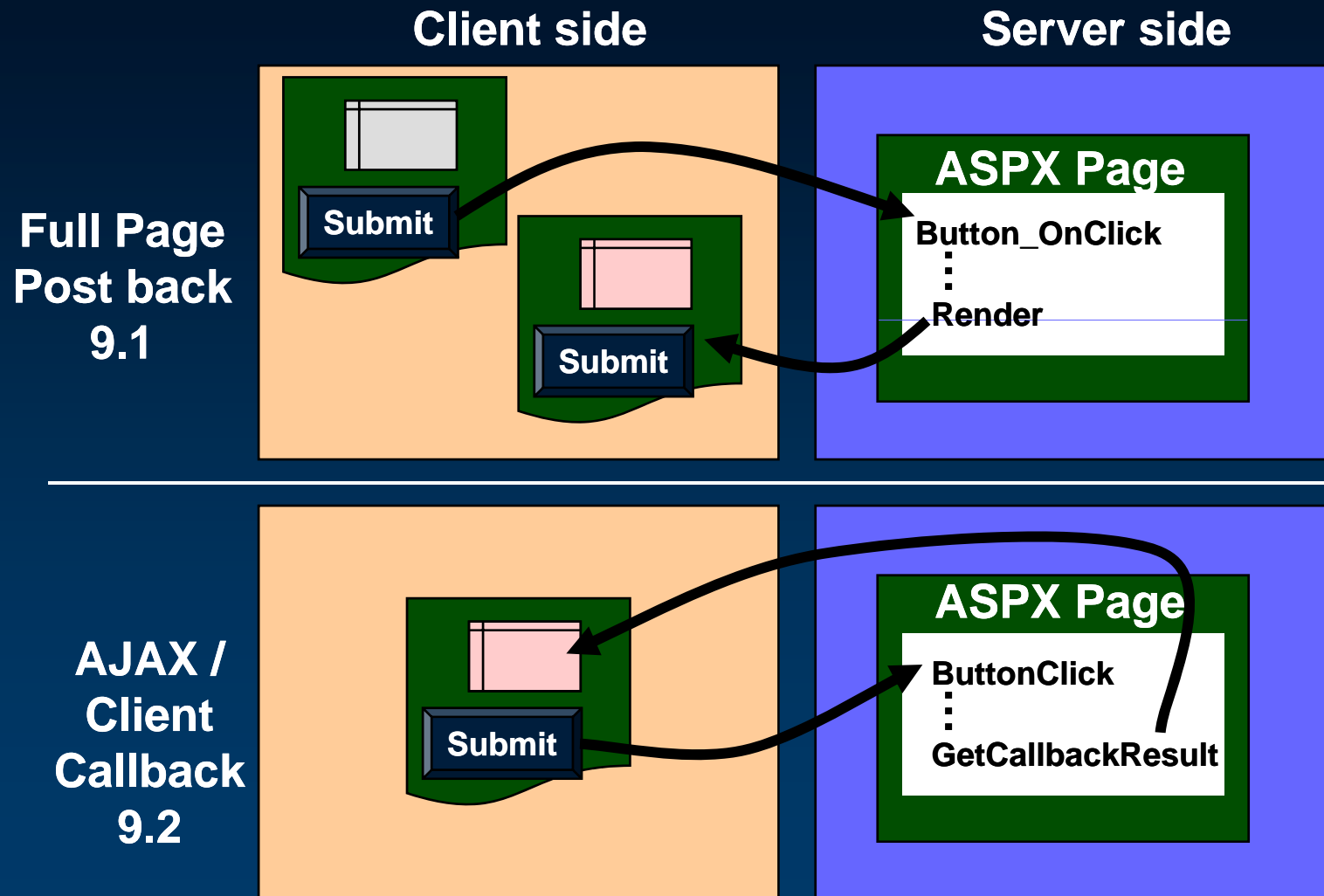
Our assumptions

- **Proficiency in Technology**
 - ASP.NET
 - Web Development techniques
 - HTML/JavaScript
- **Understanding of ESRI Server Products**
 - ArcGIS Server
 - Web ADF
- **Examples and discussion will use**
 - Visual Studio.NET 2005 – SP1
 - ASP.NET 2.0
 - C#
 - ArcGIS Server for Microsoft .NET Web ADF

ArcGIS Server Web Application Developer Framework (ADF)



Postbacks and callbacks



Web ADF and AJAX

- **Extends the ASP.NET 2.0 callback framework to generate and process asynchronous requests within a Web application**
- **The framework includes**
 - a set of tool/command interfaces accessed via a Toolbar control to process callbacks on the server
 - a set of JavaScript libraries to manage callbacks on the client,
 - public callback properties on each Web ADF control
 - a set of custom classes to manage callback messages

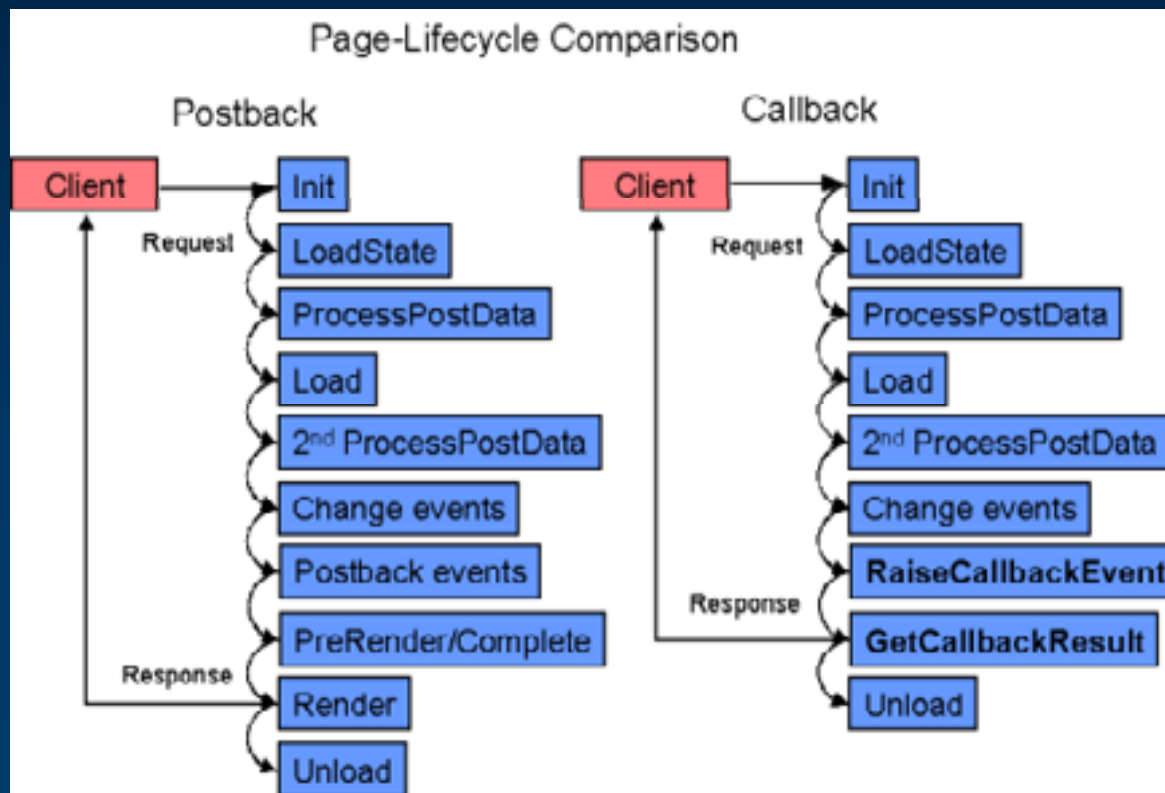
AJAX enables communication...

- **AJAX (Asynchronous JavaScript & XML)**
 - Refresh elements of Web page instead of full page
 - XML rarely used
- **XMLHttpRequest object in most recent browsers**
 - Enables sending requests without post back
 - Implementation details differ by browser
 - Not integrated into server-side code
- **Many implementations available**
 - Prototype
 - DOJO
 - Microsoft's ASP.NET AJAX 1.0
 - ASP.NET 2.0: ICallbackEventHandler

ASP.NET 2.0 Callback Framework

- **ASP.NET Callback Manager**

- Control handling the callback implements `ICallbackEventHandler`
- Client script components generate JavaScript used by the browser for AJAX communication



ASP.NET 2.0 - Working with callbacks:

Step 1 - Implementation

- Implement the `System.Web.UI.ICallbackEventHandler`
 - `RaiseCallbackEvent(eventArgs)`
 - Process custom string returned from browser
 - `GetCallbackResult()`
 - Construct custom string response sent to browser

```
public partial class Default : System.Web.UI.Page, System.Web.UI.ICallbackEventHandler
{
    string returnstring;

    string ICallbackEventHandler.GetCallbackResult()
    {
        return returnstring;
    }

    void ICallbackEventHandler.RaiseCallbackEvent(string eventArgument)
    {
        if (eventArgument == "getservertime")
        {
            returnstring = DateTime.Now.ToString();
        }
    }
}
```

ASP.NET 2.0 - Working with callbacks:

Step 2 - Control client scripts

- Call **GetCallbackEventReference**
 - Downloads WebForms.js to browser (embedded in System.Web.dll)
 - Constructs call to JavaScript function WebForm_DoCallback

```
public string sCallBackFunctionInvocation;  
  
protected void Page_Load(object sender, EventArgs e)  
{  
    sCallBackFunctionInvocation = Page.ClientScript.GetCallbackEventReference(this,  
        "message", "processMyResult", "context", "processMyError", true);  
}
```

GetCallbackEventReference arguments

Argument	Description
control	Control implementing ICallbackEventHandler
argument	JS variable with string to send to server
clientCallback	JS function to receive response
context	JS variable with context of the callback
clientErrorCallback	JS function for error response
useAsync	Synchronous or asynchronous call

ASP.NET 2.0 - Working with callbacks: Step 3 - HTML/JavaScript

- Add client-side components and JavaScript

```
<html>
<head>
  <title>Untitled Page</title>
  <script language="javascript" type="text/javascript">
    function getServerTime()
    (
      var message = 'getservertime';
      var context = 'Page1';
      WebForm_DoCallback('__Page', message, processMyResult, context, postMyError, true)
    )

    function processMyResult(returnmessage, context){
      var timediv = document.getElementById('timelabel');
      timediv.innerHTML = returnmessage;
    }

    function postMyError(returnmessage, context){
      alert("Callback Error: " + returnmessage + ", " + context);
    }
  </script>
</head>
<body>
  <form id="form1" runat="server">
    <input type="button" value="Get Server Time" onclick="getServerTime();" />
    <div id="timelabel"></div>
  </form>
</body>
</html>
```

Callback Request

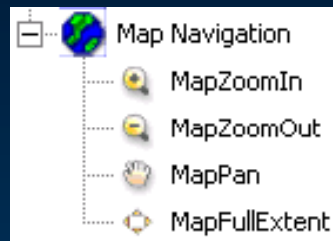
Callback Response

Get Server Time

5/22/2006 7:51:47 PM

Map Actions and the Toolbar control

- Web ADF Toolbar provides a framework to work with Map control using callbacks and Web ADF JavaScript
- Out-of-the-box map actions include:
 - Zoom In
 - Zoom Out
 - Pan
 - Full Extent
- Tool properties available in Visual Studio design-time dialog



Properties:

Appearance: Images	
DefaultImage	~/images/select_point_1.gif
DisabledImage	
HoverImage	~/images/select_point_1U.gif
SelectedImage	~/images/select_point_1D.gif

Appearance: Text	
Text	Add Point
ToolTip	

Client-side Action	
ClientAction	Point
JavaScriptFile	

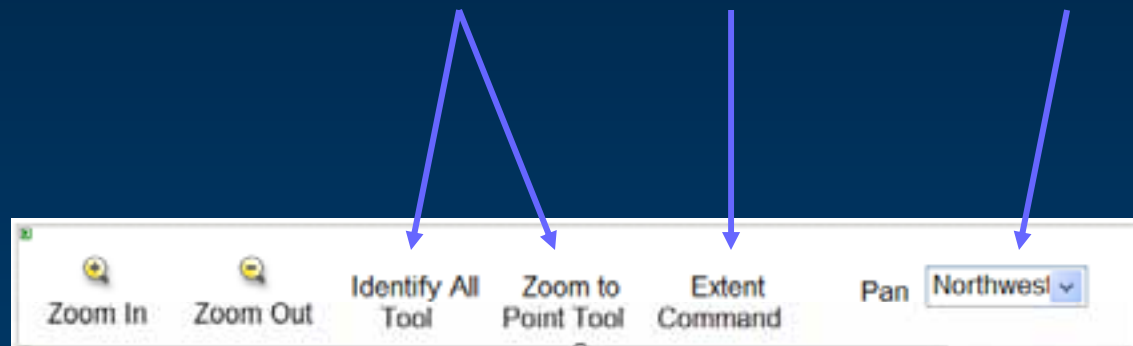
General	
BuddyItem	
Disabled	False
Name	PointTool
ShowLoading	True

Misc	
Type	ExtendedTool

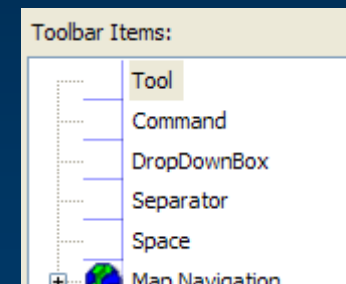
Server-side Action	
EnablePostBack	False
ServerActionAssembly	App_Code
ServerActionClass	GraphicPointTool

Custom tools: handle callbacks easily

- Using Web ADF Toolbar control
- Callback mechanism handled behind the scenes
- Only works for actions on Map
- Can be tool, command or drop-down



Toolbar with custom tools



Toolbar Designer in Visual Studio

Creating a custom tool

- **To create a tool/command:**
 1. Write server-side code to handle map action
 2. Add a new tool to toolbar
 3. Set tool's server action to the server-side code
 4. For tool, choose client action (point, rectangle, etc.)
 5. Set tool images



Custom tool:

1. Server code for map action

- Create public class that implements action interface
 - IMapServerToolAction for tool
 - Implement required ServerAction method
- ToolEventArgs has user geometry (point, circle, etc.)
- Perform action on the map
- Web ADF handles callbacks to apply map changes

```
public class CustomTool : IMapServerToolAction
{
    public void ServerAction(ToolEventArgs args)
    {
        Map map = (Map) args.Control;
        PointEventArgs pargs = (PointEventArgs) args;
        if (map != null && pargs != null)
        {
            map.CenterAt(pargs.ScreenPoint);
        }
    }
}
```

No callback
handling
code!

Custom tool:

1. Server code for map action (continued)

- Implement appropriate interface for toolbar item

Toolbar item	Implement interface	Event argument
Tool	IMapServerToolAction	ToolEventArgs – cast for client action: <ul style="list-style-type: none">• PointEventArgs• RectangleEventArgs• VectorEventArgs• CircleEventArgs• ...
Command	IMapServerCommandAction	ToolbarItemInfo
DropDownBox	IMapServerDropDownBoxAction	ToolbarItemInfo

Custom tool: Add tool to Toolbar and set properties

2. Add new tool to toolbar
3. Set tool's server action to the server-side code
4. For tool, choose client action (point, rectangle, etc.)
5. Set tool images

The screenshot shows the 'ToolbarCollectionEditorForm' window. On the left, the 'Toolbar Items' list includes 'Tool'. On the right, the 'Current Toolbar Contents' list includes 'Tool'. A green circle with the number '2' is placed over the 'Add >>' button. Below the toolbar editor, the 'Properties' window is open, showing the configuration for the 'Tool'.

Appearance: Images

DefaultImage	~/images/select_point_1.gif
DisabledImage	
HoverImage	~/images/select_point_1U.gif
SelectedImage	~/images/select_point_1D.gif

Appearance: Text

Text	Add Point
ToolTip	

Client-side Action

ClientAction	Point
JavaScriptFile	

General

BuddyItem	
Disabled	False
Name	Point Tool
ShowLoading	True

Misc

Type	ExtendedTool
------	--------------

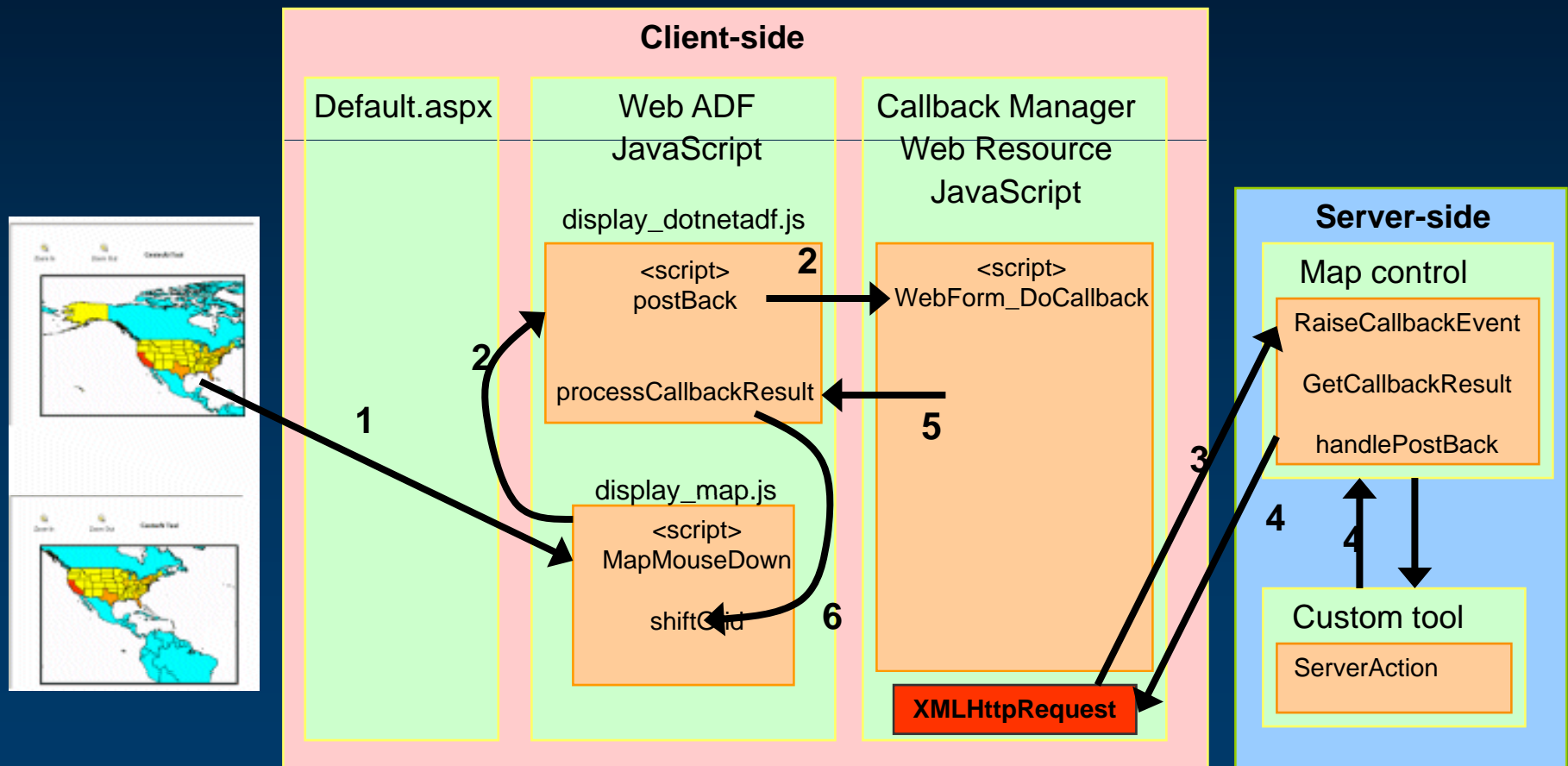
Server-side Action

EnablePostBack	False
ServerActionAssembly	App_Code
ServerActionClass	GraphicPointTool

A green circle with the number '5' is placed over the 'Appearance: Images' section, a green circle with the number '4' is placed over the 'Client-side Action' section, and a green circle with the number '3' is placed over the 'Server-side Action' section.

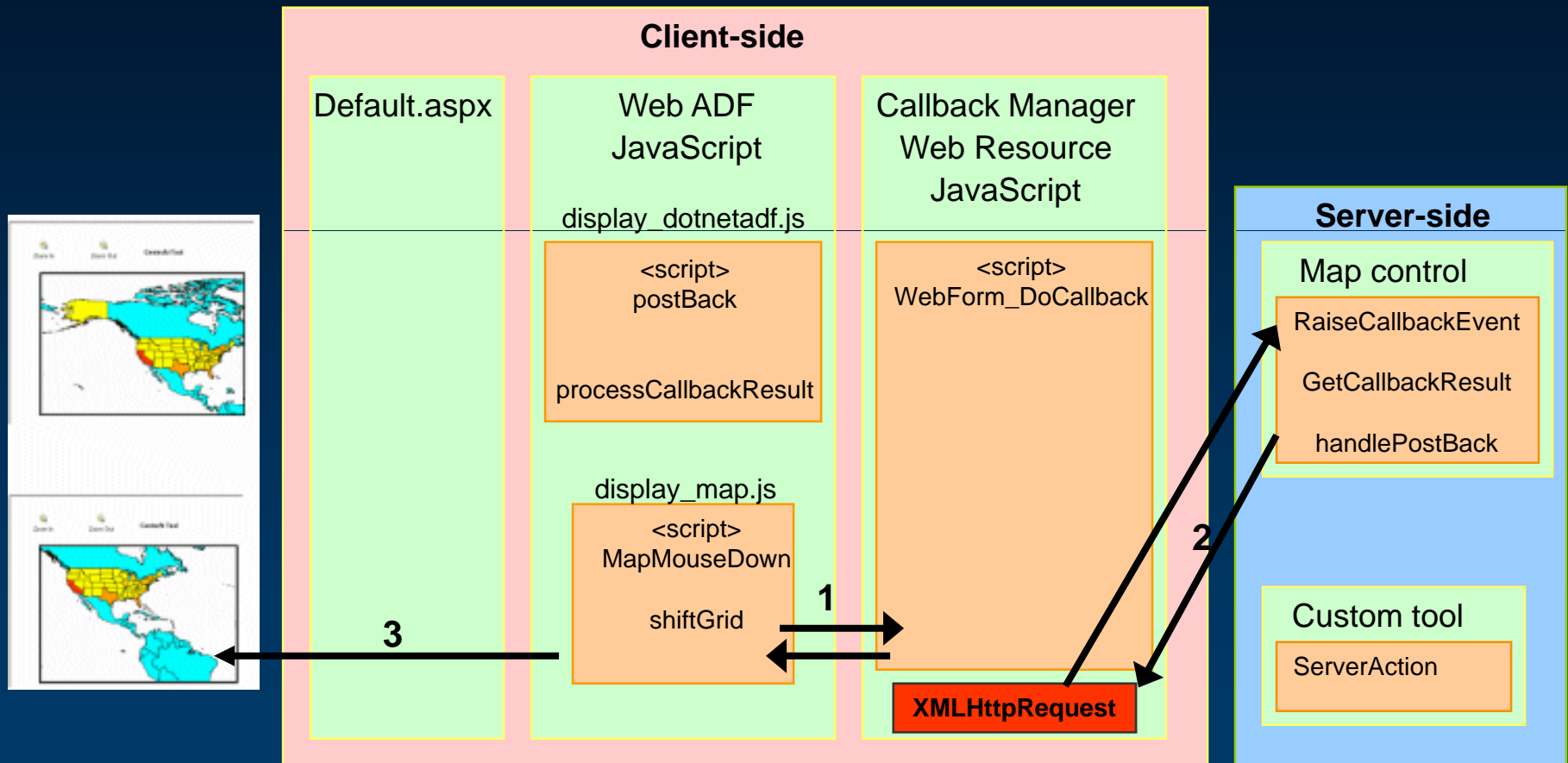
Custom tool callback workflow

- Initial user action and callback



Custom tool callback workflow

- Subsequent callbacks to process action



Toolbar considerations

- **Page can have multiple toolbars**
 - Buddy each toolbar to a Map
 - If buddy to same Map, set **ToolbarGroup** property to same value
 - Tools will then activate/deactivate across toolbars
- **Custom tasks may have toolbars**
 - Add in code to create task (**CreateChildControls**)
 - Add class to handle toolbar action
- **Tools can be directly managed from Map**
 - Map has **CurrentToolItem** property
 - Would need to manage tool properties with custom code

Callbacks and the Web ADF Controls

- **What about callbacks if outside the toolbar?**
 - **Examples:**
 - **Label that displays map scale as user zooms map**
 - **Text boxes to enter coordinates to zoom map to**
- **One model is...**
 - **Utilize the ASP.NET 2.0 callback framework in a page, custom control or custom task and role your own**

Web Controls and Tasks – AJAX Enabled

- **Use callback handling built into Web ADF controls**
 - Makes dealing with callbacks simpler
 - Web ADF controls implement **ICallbackEventHandler**
 - Inherited from base classes
 - ESRI.ArcGIS.ADF.Web.UI.WebControl
 - ESRI.ArcGIS.ADF.Web.UI.CompositeControl
 - **CallbackResults** property that stores callback results
 - Utilize Web ADF JavaScript to process callback response

Web ADF JavaScript Libraries

- **Provide a rich client-side environment for interacting with Web ADF controls.**
 - Prepackaged as WebResources for Web ADF controls.
 - Source files installed on disk
- **processCallbackResult function**
 - From a callback perspective, is most important.
 - in `display_dotnetadf.js`
 - Function is designed to parse and process callback messages from Web ADF controls and update their content, at runtime, on the client.
- **Custom asynchronous callback solutions can be developed**
 - For Example: initiate changes in map display and update other non-Web ADF components on a page

Web ADF Control Callbacks

- **Web ADF controls are designed to handle callback requests and generate callback responses.**
- **Each Web ADF control has a `CallbackResults` property.**
 - **Stores the callback messages the control will send to the client (`processCallbackResult` function).**
 - **Custom solutions can use the `GetCallbackresult` method**
 - ***For Example:* Updating a Web ADF control on the client, you will need to retrieve the callback response message from the `CallbackResults` property and return it to the client (e.g. using the `GetCallbackResult` method).**

Callback Classes

- **CallbackResultCollection**

- This collection can be modified to include custom **CallbackResult** objects

- **CallbackResult**

- Custom **CallbackResult** objects can be used to interact with non-Web ADF content in the Web page (e.g. HTML table, GridView, images, text).

- *Value Add: you can use the existing Web ADF JavaScript libraries to process the callback response on the client instead of writing your own JavaScript*

Working with Custom Callback Events

- To work with custom elements in the browser, create a custom **CallbackResult** object
- Example: display simple alert upon map callback

```
string jsfunction = "alert('Hello');";  
CallbackResult cr = new CallbackResult(null, null, "javascript", jsfunction);  
Map1.CallbackResults.Add(cr);
```

```
CallbackResult(string controlType, string controlId, string eventArg, object[] parameters)
```

Options for eventArg

eventArg	Description
“content”	Sets outerHTML of element
“innercontent”	Sets innerHTML of element
“image”	Sets src of image
“javascript”	Executes JavaScript (function)

Putting it all together

- Mash-ups and the Web ADF



Best Practices in building Web Applications

- **Keep it simple**
 - The best web application is the one that people use
- **AJAX – Myth or Mystery or ???**
 - by itself, will not solve all problems
 - Apply OO development principals to web development
 - Using XML is nice, but is slow
 - Use strings or JSON
 - AJAX is chatty
 - Look for caching opportunities
 - Batch requests

Best Practices in building Web Applications

- **Understand the Network**
 - Be aware that the network is unreliable at best
 - Failures and users leaving the page before operation completes
 - At most, browser uses 2 simultaneous connections per domain
 - Leverage services with sub-domains
- **Must design for proper feedback**
 - Prioritize and control order
 - More interactivity == more code == slower site
- **Scripts tag stops the world**
 - Send scripts dynamically

Best Practices in Web Application Design

- **Always provide immediate feedback that shows something is working**
- **Preserve user state**
 - Try to come back to where the user left off
- **Controls**
 - Build UI components for re-use by others
- **Divide & Conquer**
 - Separate presentation and implementation
- **Use Services**
 - Web Services allow the sharing of business logic in a generic way

Presentation materials

- PowerPoint presentation and code are posted on the conference web site
 - <http://www.esri.com/events/devsummit/index.html>
- EDN – downloads and videos



Microsoft Keynote

Wednesday 6:00pm
Oasis 2

Eddie Amos

Senior Director, Developer & Platform Evangelism

Further questions?

- **TECH-TALK AREAS**

- **What:** Further opportunity to discuss questions and concerns with presenters and subject matter experts

- **Where:**

- **When:** during the next 30 minutes

- **ESRI Showcase**

- **Meet the teams**

- **ESRI Developers Network (EDN) website**

- **<http://edn.esri.com>**



Thank you