



Developing Custom Web Tasks Using the .NET Web ADF

Sentha Sivabalan and Rex Hansen

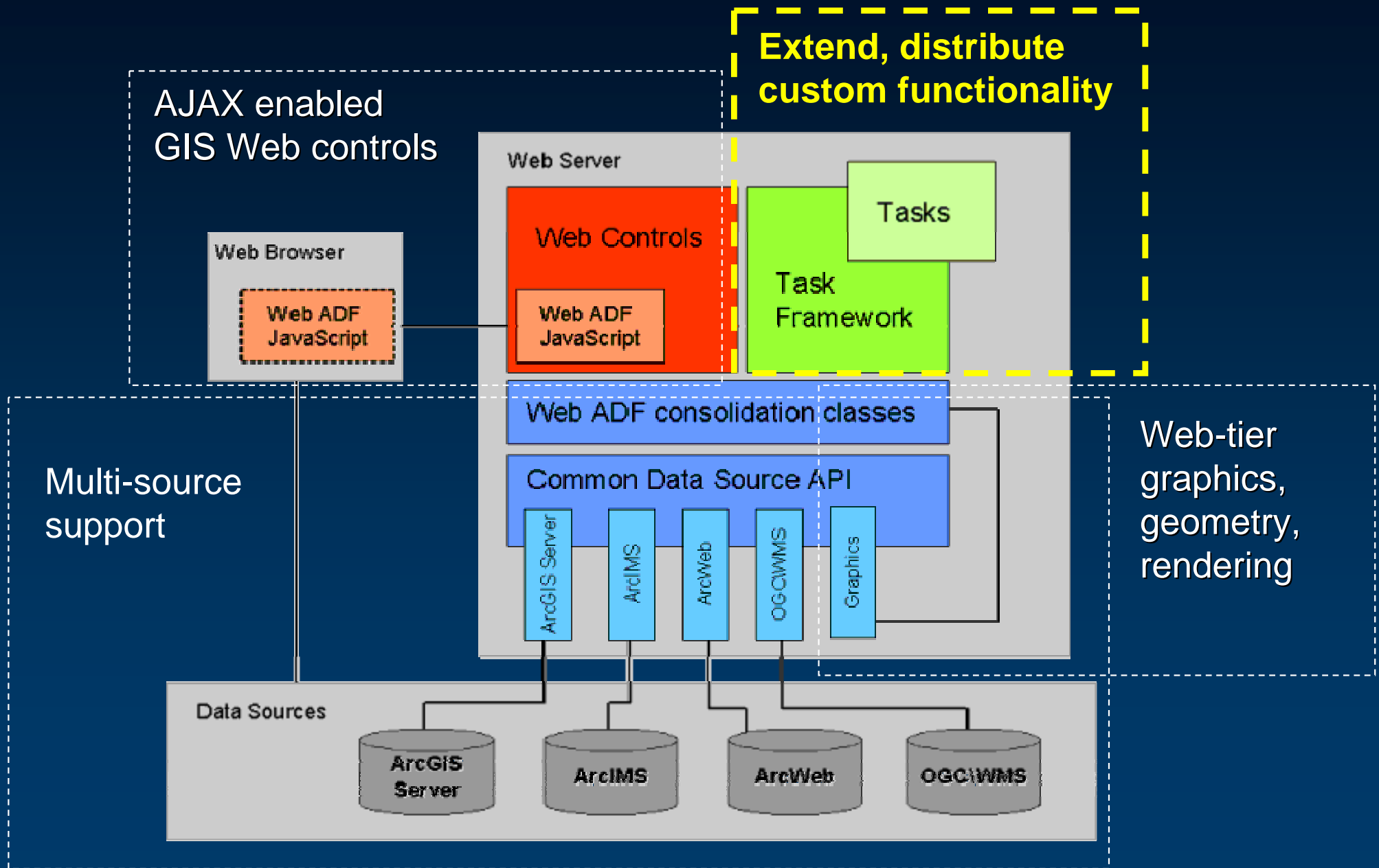
Session Topics

- **Overview of the Web ADF Task Framework**
- **Task Runtime Customization**
- **Visual Studio Integration**
- **Manager Integration**
- **Task Deployment**
- **Debugging**

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 ADF Overview



Web ADF Task Framework

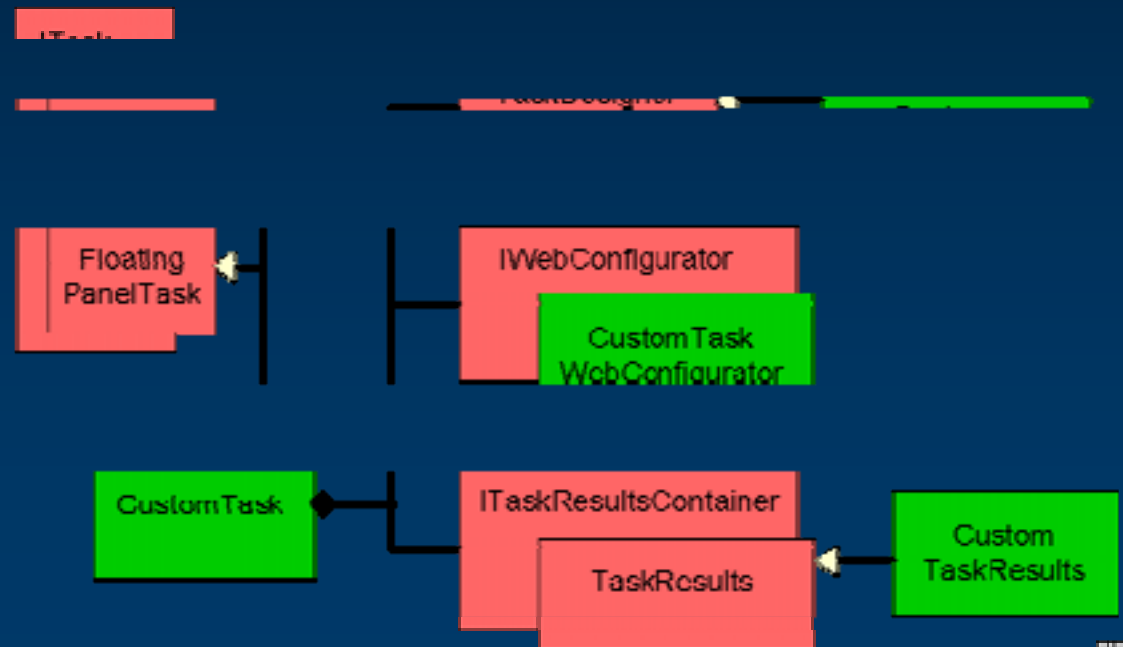
- **Build custom “Web Tasks”**

- Web tasks are visual components designed to perform a set of related actions and generate a common result

Out-of-the-box Tasks

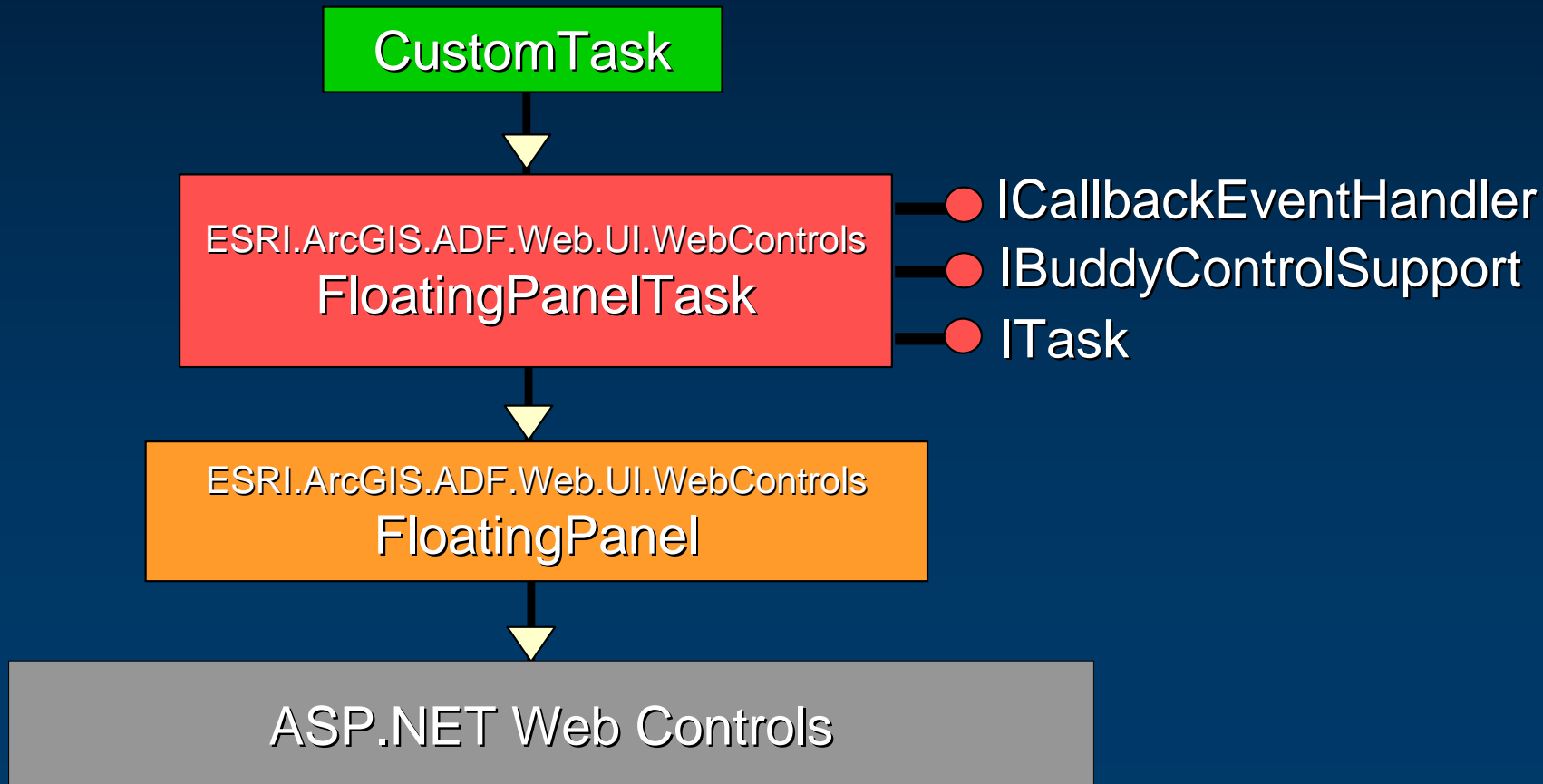
- SearchAttributes
- FindAddress
- FindPlace
- QueryAttributes
- Geoprocessing
- Editor

Extensible framework to support creating and deploying custom tasks



Custom Web Task Implementation

- Basic task implements ITask
- Create a custom Web control
 - Extend Task or FloatingPanelTask abstract base classes

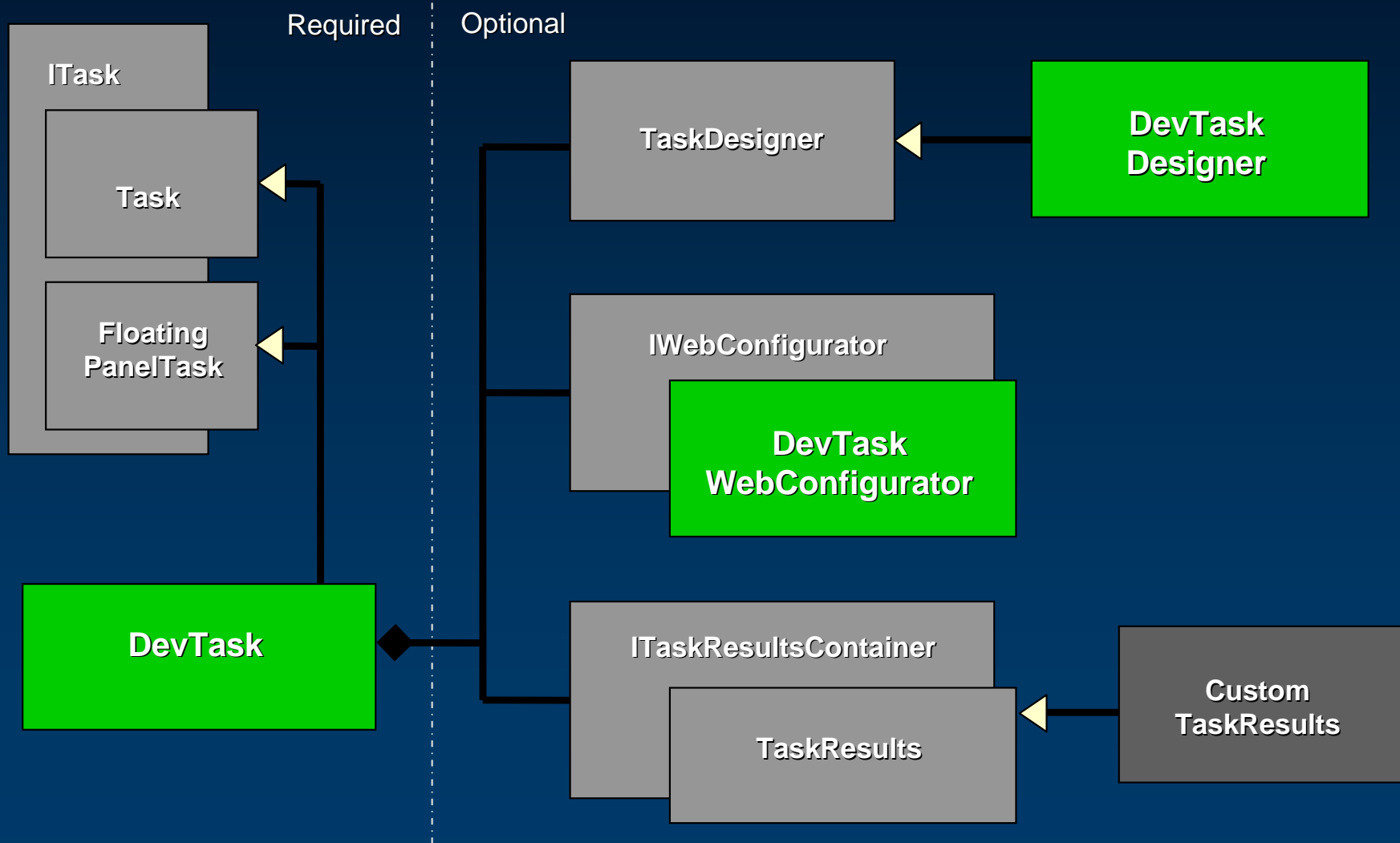


Why use the Task Framework?

- **Distributable UI components**
 - Can be used by any Manager User
- **Framework provides for:**
 - Organization of tasks
 - Feedback when a task is executing
 - Management of results
 - Display
 - Visualization on Map
 - Delete, Refresh, Re-run

Custom Task Classes in this Session

- .NET Class Library Project



Key methods and properties

Methods

- **CreateChildControls** – create the visual interface
- **GetCallbackResult** – process callback content
- **ExecuteTask** – execute the task and generate results

Properties

- **Input** – store input data from the user
- **Results** – after task execution, generate results for use in a TaskResults control
- **CallbackResults** – collection of results designed to update browser content using Web ADF JavaScript

Design Task Runtime Interface for user input

- **Override CreateChildControls**
 - Create content user will interact with in the browser
 - Called on every postback\callback during Page Load

DevTask.cs

```
[System.Web.UI.ToolboxData("<{0}:DevTask runat=\"server\" > </{0}:DevTask>")]
public class DevTask: ESRI.ArcGIS.ADF.Web.UI.WebControls.FloatingPanelTask
{
    private System.Web.UI.WebControls.TextBox textBox = null;

    protected override void CreateChildControls()
    {
        base.CreateChildControls();
        textBox = new System.Web.UI.WebControls.TextBox();
        textBox.ID = "textBox";
        Controls.Add(textBox);
    }
}
```

Utilize ADF JavaScript to execute task

- Task and FloatingPanelTask include Web ADF JavaScript to support task execution
- Manage events in browser to execute task

DevTask.cs

```
protected override void CreateChildControls()
{
    . . .
    string getArgumentJS = string.Format("'textBoxValue=' +
        document.getElementById('{0}').value", textBox.ClientID);
    string onClick = string.Format("executeTask({0},\"{1}\");",
        getArgumentJS, CallbackFunctionString);
    button.Attributes.Add("onclick", onClick);
}
```

display_task.js

```
function executeTask(...
function
    startActivityIndicator{...
function startJob{...
```

Task execution in the browser

- At runtime, a browser event triggers task execution
- User input is passed as argument\value pairs to the `executeTask` JavaScript function

`MyPage.aspx` (includes `DevTask` control)

DevTask1

My Input Text Execute Task

onClick

```
onclick="executeTask  
( 'textBoxValue=' + document.getElementById( 'DevTask1_textBox' ).value ,  
&quot;WebForm_DoCallback( 'DevTask1' , argument , processCallbackResult , context , postBackError  
, true )&quot; );"
```

`display_task.js`

```
function executeTask(...
```

Processing user input

- **Task and FloatingPanelTask designed to work with callbacks by implementing ICallbackEventHandler**
 - RaiseCallbackEvent captures event arguments (_callbackArg)
 - GetCallbackResult processes inputs, executes task
- **Input property stores user input**
 - Storage format determined by developer

DevTask.cs

```
public override string GetCallbackResult()  
{  
    System.Collections.Specialized.NameValueCollection keyValColl =  
        CallbackUtility.ParseStringIntoNameValueCollection(_callbackArg);  
    Input = keyValColl["textBoxValue"];  
    return base.GetCallbackResult();  
}
```

Executing the Task

- Implement the ExecuteTask method
- FloatingPanelTask calls ExecuteTask from GetCallbackResult method

DevTask.cs

```
GetCallbackResult()
```

```
ExecuteTask()
```

FloatingPanelTask.cs

```
public override string GetCallbackResult()  
{  
    . . .  
    string eventArg = keyValColl["EventArg"];  
    . . .  
    else if (eventArg == "executeTask")  
    {  
        ExecuteTask();  
        DisplayResults(taskJobID, Input, Results);  
    }  
    return CallbackResults.ToString();  
}
```

Generating Results

- **Results property stores content to be displayed in a TaskResults control**
- **TaskResults.DisplayResults method works with:**
 - SimpleTaskResult
 - Dataset
 - TaskResultNode

DevTask.cs

```
public override void ExecuteTask()  
{  
    string textBoxValue = Input as string;  
  
    ESRI.ArcGIS.ADF.Web.UI.WebControls.SimpleTaskResult simpleresult =  
        new ESRI.ArcGIS.ADF.Web.UI.WebControls.SimpleTaskResult(heading, detail);  
  
    Results = simpleresult;
```

Working with Task Results

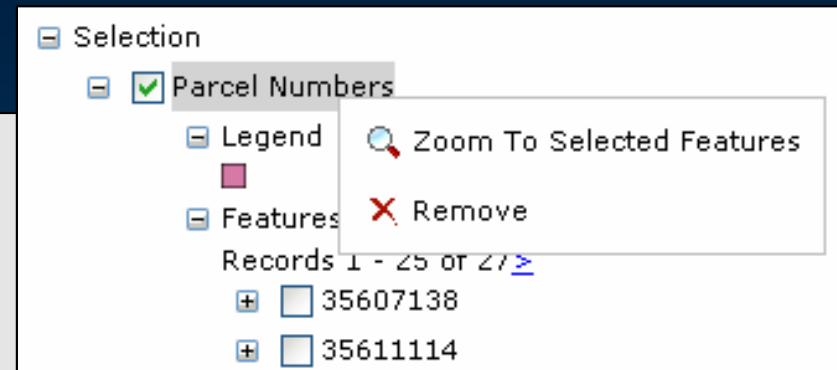
- Use methods on the TaskResults container to prepare content
- Use TaskResultNodes to package Web ADF graphics results

DevTask.cs

```
public override void ExecuteTask()  
{  
    System.Data.DataSet dataset =  
        new System.Data.DataSet();  
    dataset.Tables.Add(graphicslayer);
```

```
    TaskResultNode trn = new TaskResultNode(dataset.DataSetName);  
    taskresults.SetupTaskResultNode(null, null, null, trn);
```

```
    TreeViewPlusNode tvpn = taskresults.CreateDataTableNode(graphicslayer,  
        true, true, true, "graphicsid", "Parcel Numbers");  
    trn.Nodes.Add(tvpn);  
    Results = tvpn;
```



Working with CallbackResults

- Upon task execution, callback response string is returned from custom task `GetCallbackResult` method
- `CallbackResults` are callback messages formatted to be processed by Web ADF JavaScript
- All Web ADF controls maintain a `CallbackResults` collection

Push Web ADF control and custom callback results into task callback results

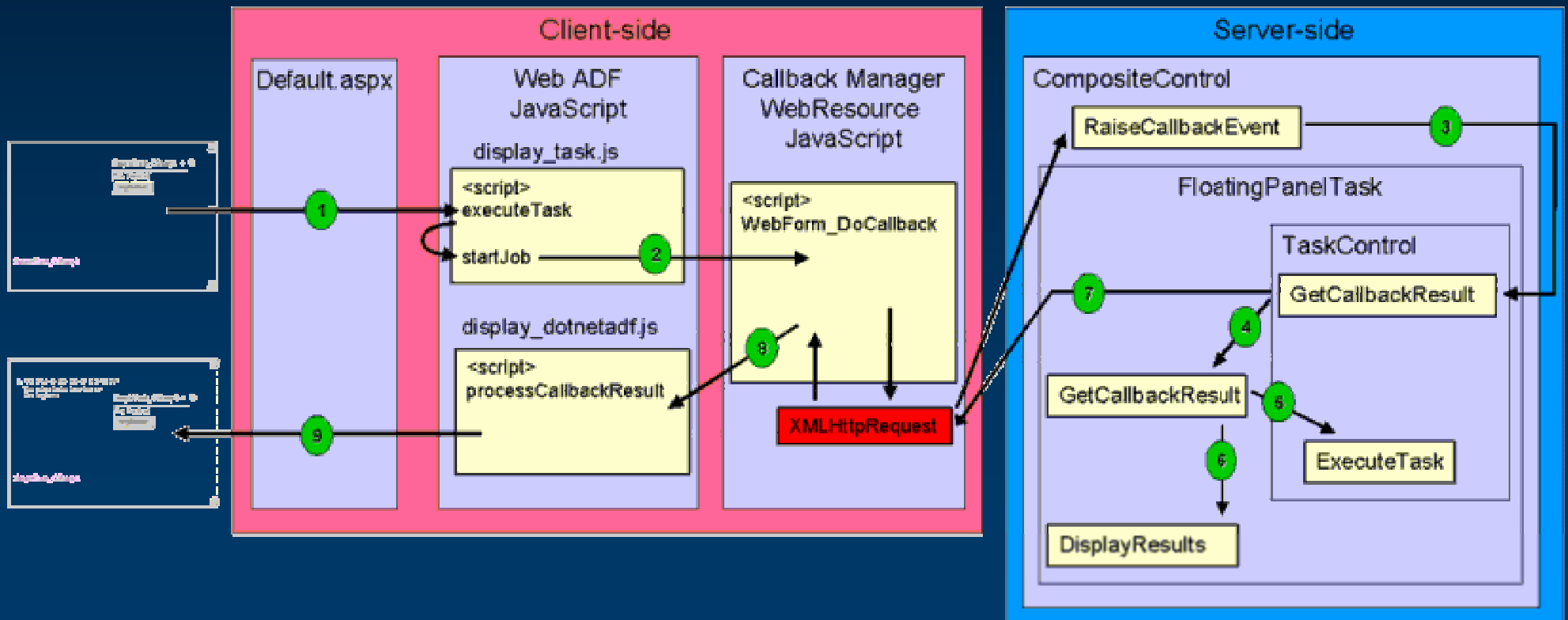
`DevTask.cs`

```
public override void ExecuteTask()  
{  
    map.CenterAt(MyPoint);  
    CallbackResults.CopyFrom(map.CallbackResults);  
}
```



Task Runtime Workflow

- Task Execution
 - Start Activity Indicator
 - Execute Task



Enhance Task Usability in the Browser

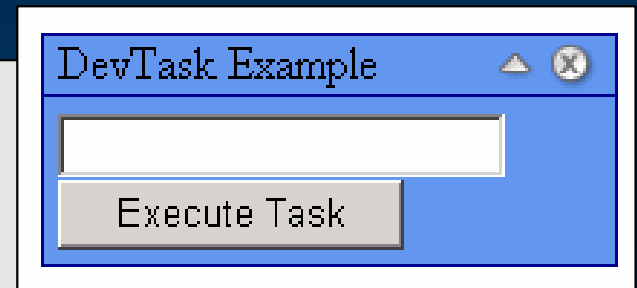
- Include multiple arguments and values in callback request

```
string getArgumentJS = string.Format("
    'textBoxValue=' + document.getElementById('{0}').value +
    '&dropDownListValue=' + document.getElementById('{1}').value",
    textBox.ClientID, dropDownList.ClientID);
```

- Set FloatingPanel properties

```
protected override void OnPreRender(EventArgs e)
{
    base.OnPreRender(e);

    Title = "DevTask Example";
    BackColor = System.Drawing.Color.CornflowerBlue;
    BorderColor = System.Drawing.Color.DarkBlue;
    BorderStyle = System.Web.UI.WebControls.BorderStyle.Solid;
    BorderWidth = System.Web.UI.WebControls.Unit.Pixel(1);
    Style[System.Web.UI.HtmlTextWriterStyle.Width] = "200px";
```



Embedded Resources

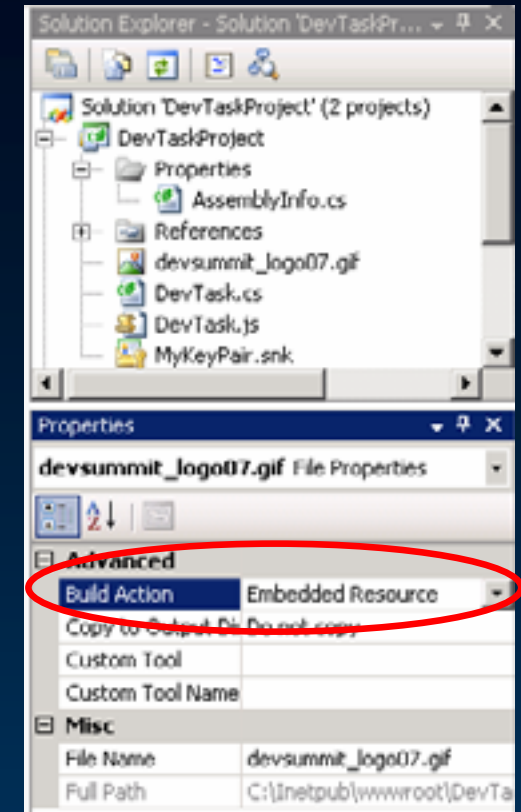
- Embed resources used by the task
 - Images and JavaScript
 - Easy to deploy, included in the task assembly

AssemblyInfo.cs

```
[assembly: WebResource("DevTaskProject.devsummit_logo07.gif", "image/gif")]  
[assembly: WebResource("DevTaskProject.DevTask.js", "text/javascript")]
```

DevTask.cs

```
protected override void CreateChildControls()  
{  
    . . .  
    System.Web.UI.HtmlControls.HtmlImage image =  
        new System.Web.UI.HtmlControls.HtmlImage();  
    image.Src =  
        Page.ClientScript.GetWebResourceUrl(typeof(DevTask),  
        "DevTaskProject.devsummit_logo07.gif");  
}
```



Manage Non-executing callbacks

- Initiate callbacks to update task control in preparation to execute
- May use embedded JavaScript

DevTask.cs

```
protected override void OnPreRender(EventArgs e)
{
    string scriptLocation =
        Page.ClientScript.GetWebResourceUrl(typeof(DevTask),
        "DevTaskProject.DevTask.js");
    Page.ClientScript.RegisterClientScriptInclude("DevTaskProject",
        scriptLocation);
    . . .

protected override void CreateChildControls()
{
    . . .
    string onClick = string.Format("sendRequest({0}, \"{1}\");",
        getArgumentJS, CustomCallbackFunction);
```

DevTask.js

```
function sendRequest(...
```

Accessing Web ADF Resources at Runtime

- Only needed during initial load?
 - Use PreRender
- Needed during every callback?
 - Use CreateChildControls or Load

DevTask.cs

```
protected override void OnPreRender(EventArgs e)
{
    base.OnPreRender(e);
    UpdateResources();
}

private UpdateResources()
{
    MapResourceManager rm = (MapResourceManager)Page.FindControl(mrmID);
    if (!rm.Initialized)
        rm.Initialize();
}
```

Refine custom task control

- Add a custom tag prefix
- Define a version

AssemblyInfo.cs

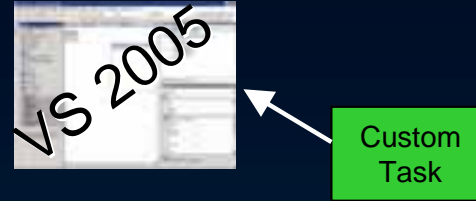
```
[assembly: System.Web.UI.TagPrefix("DevTaskProject", "devTask")]

[assembly: AssemblyVersion("1.1.0.0")]
[assembly: AssemblyFileVersion("1.1.0.0")]
```

MyPage.aspx or Web.Config

```
<%@ Register Assembly="DevTaskProject, Version=1.1.0.0,
    Culture=neutral, PublicKeyToken=a284737434b9d17c"
    Namespace="DevTaskProject"
    TagPrefix="devTask" %>
```

Visual Studio Integration



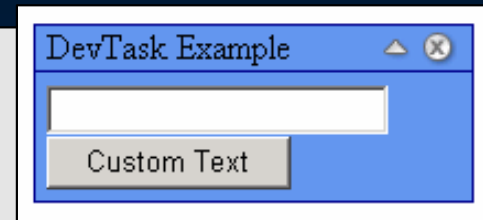
- Custom attributes
- Extend TaskDesigner to manage smart tags
 - Enable Web ADF resource access
- Custom toolbox image

Custom task attributes

- Set properties at design-time, change runtime behavior

DevTask.cs

```
public string ButtonText
{
    get
    {
        object o = StateManager.GetProperty("buttonText");
        return (o == null) ? "Execute Task" : o as string;
    }
    set
    {
        StateManager.SetProperty("buttonText", value);
    }
}
```



MyPage.aspx (design-time)

```
<devTask:DevTask ID="DevTask1" runat="server" ButtonText="Custom Text">
</devTask:DevTask>
```

Extend TaskDesigner

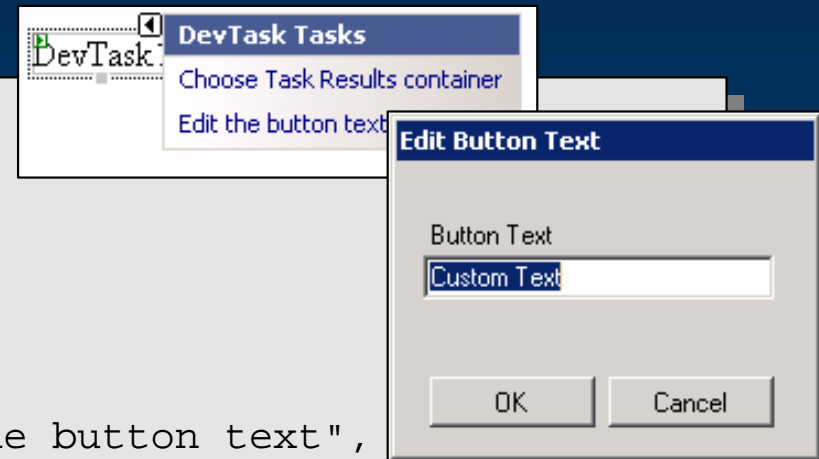
- Visual interface for modifying task properties at design-time

DevTask.cs

```
[System.ComponentModel.Designer(typeof(DevTaskDesigner))]  
public class DevTask: ESRI.ArcGIS.ADF.Web.UI.WebControls.FloatingPanelTask  
{
```

DevTaskDesigner.cs

```
public class DevTaskDesigner : TaskDesigner  
{  
    public DevTaskDesigner()  
        : base()  
    {  
        verbs.Add(new DesignerVerb("Edit the button text",  
            new EventHandler(OnEditButtonText)));  
    }  
}
```

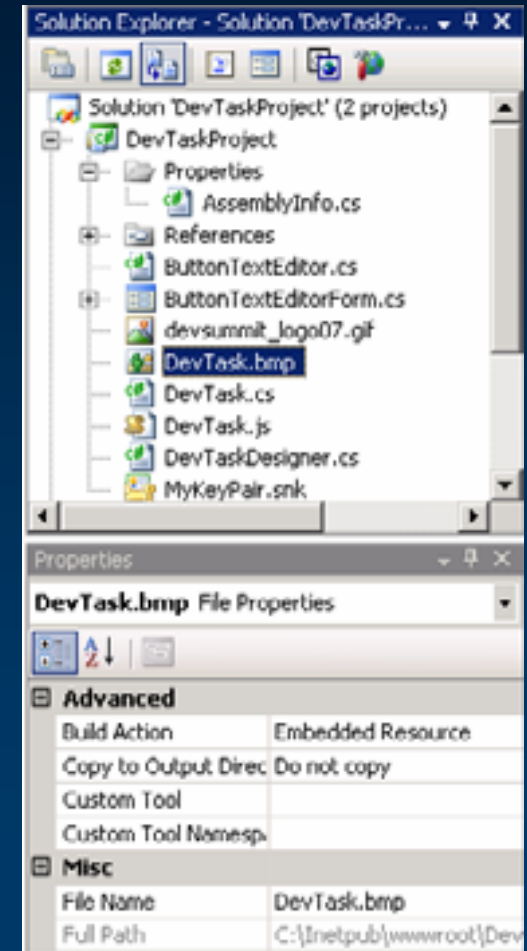
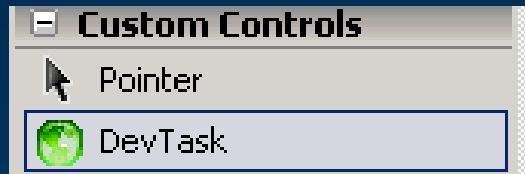


Refine visual display of control in Visual Studio

- **Toolbox image for custom task Web control**

DevTask.cs

```
[System.Drawing.ToolboxBitmap(typeof(DevTask))]
public class DevTask:
    ESRI.ArcGIS.ADF.Web.UI.WebControls.FloatingPanelTask
    {
```



Manager Integration

Custom
Task



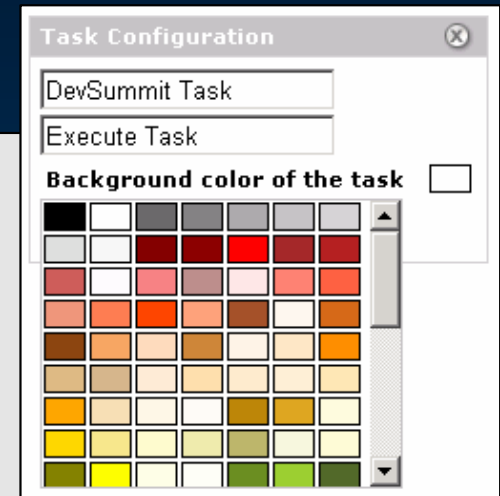
- Runtime rendering in Manager
- Implement IWebConfigurator interface
- Resource dependencies and validation
- Managing output to Web Mapping Application

Design Web control to modify custom task

- To configure a task in Manager requires a custom Web control

DevTaskWebConfigurator.cs

```
public class DevTaskWebConfigurator:  
    ESRI.ArcGIS.ADF.Web.UI.WebControls.CompositeControl  
{  
    private System.Web.UI.WebControls.Button okButton;  
  
    protected override void CreateChildControls()  
    {  
        okButton = new System.Web.UI.WebControls.Button();  
        okButton.Click += new EventHandler(okButton_Click);  
    }  
}
```



IWebConfigurator implementation

- Manager events and properties available by implementing interface
- Properties
 - **ControlToConfigure**: returns an instance of the Task
 - **AdditionalControls**: returns other controls in same Page as the Task
 - **ValidateResources**: determine if a required resource is available
- Event Handlers
 - **WebConfigurationCompleteEventHandler**: use to handle events when finished configuring the Task
 - **WebConfigurationCancelledEventHandler**: use to handle events when configuring the Task is cancelled

Configure custom task properties

- **ControlToConfigure** returns an instance of the custom task
- **Handle OnWebConfigurationComplete** event

DevTaskWebConfigurator.cs

```
public class DevTaskWebConfigurator:
    ESRI.ArcGIS.ADF.Web.UI.WebControls.IWebConfigurator
{
    private DevTask TaskInstance = null;

    private void okButton_Click(object sender, EventArgs e)
    {
        TaskInstance.ButtonText = buttonText.Text;

        OnWebConfigurationComplete(new WebConfigurationCompleteEventArgs
            (TaskInstance, getDesignTimeTag()));
    }
}
```

Resource validation in Manager

- **Implement ValidateResources to determine if a valid resource is available**

DevTaskWebConfigurator.cs

```
public bool ValidateResources(out string message)
{
    ArrayList mrms = new ArrayList();
    Utility.FindControls(typeof(MapResourceManager),
        AdditionalControls, ref mrms);
}
```

- **Implement GetGISResourceItemDependencies to reset task when dependent resource is removed**

DevTask.cs

```
public override List<GISResourceItemDependency> GetGISResourceItemDependencies()
{
    List<GISResourceItemDependency> list = new List<GISResourceItemDependency>();
    list.Add(new GISResourceItemDependency(typeof(MapResourceItem), mritem,
        typeof(MapResourceManager), mrmanager));
}
```

Generating Web Mapping Application output

- **Create output string of declarative content for the Web Mapping Application**

`DevTaskWebConfigurator.cs`

```
private string getDesignTimeTag()
{
    string openTag = string.Format("<devTask:DevTask ID=\"{0}\"
runat=\"server\" , TaskInstance.ID);

    StringBuilder trcTag = new StringBuilder();
    trcTag.Append("<TaskResultsContainers>");
    trcTag.Append("<esri:BuddyControl Name=\"TaskResults1\" />");
    trcTag.Append("</TaskResultsContainers>");

    string endTag = "</devTask:DevTask>";
    . . .
    return alltags.ToString();
}
```

Register with custom task and Manager

- Add attribute to custom task

DevTask.cs

```
[WebConfigurator(typeof(DevTaskWebConfigurator))]  
public class DevTask: ESRI.ArcGIS.ADF.Web.UI.WebControls.FloatingPanelTask  
{
```

- Modify Tasks.xml

- Located in <ArcGIS Instance>\Manager\Applications\App_Data

Tasks.xml

```
<Task Name="DevTask" DisplayName="DevSummit Task"  
Type="DevTaskProject.DevTask, DevTaskProject, Version=1.1.0.0,  
Culture=neutral, PublicKeyToken=a284737434b9d17c"  
TagPrefix="devTask" />
```



Task Deployment – Best Practices

- **Start with Task or FloatingPanelTask base classes**
 - Already provides some implementation code
- **Define a standard and unique tag prefix**
 - Arbitrary tag prefix may conflict with other custom tasks
- **Embed resources - images, JavaScript**
- **Sign and register (GAC)**
- **Use an Installer!**

Debugging

- Web Application runtime



IIS - ASP.NET worker process
XP: **aspnet_wp.exe**
2003: **w3wp.exe**

- Manager runtime

- Visual Studio design-time

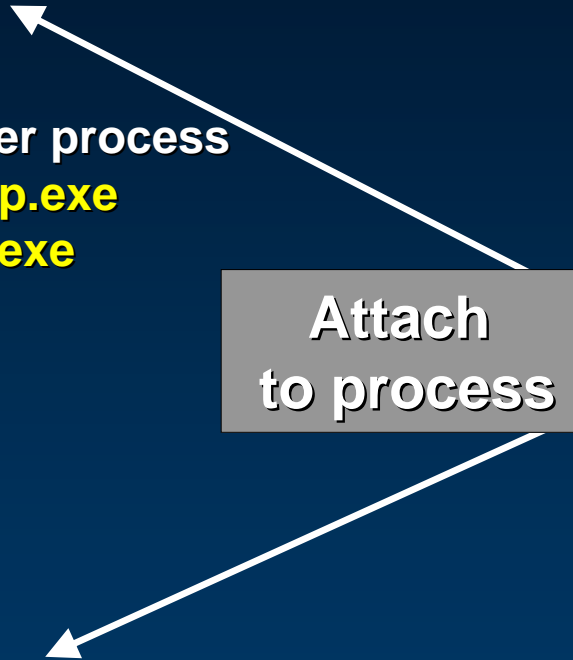


VS2005
Web Application
devenv.exe

Attach
to process



VS2005
Task Source



Custom Task Demo



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