



Building and Extending Tasks for ArcGIS Server Java Web Applications

David Cardella

James Gough



Introductions

Please!
Turn **OFF** cell phones
and paging devices



- **75 minute session**
 - 60 – 65 minute lecture
 - 10 – 15 minutes Q & A following the lecture
- **Who are we?**
 - ESRI Developer Network (EDN) Technology Lead
 - ArcGIS Server Instructor
- **Who are you?**
 - Current ArcIMS developers?
 - New to ArcGIS Server?
 - Current ArcGIS Server developers?
 - JSF Developers?

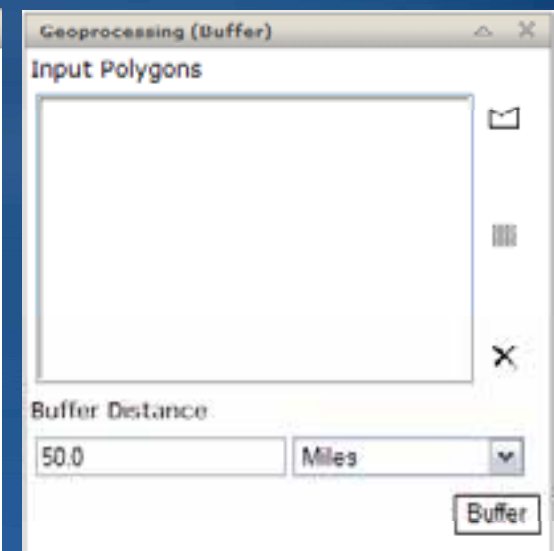
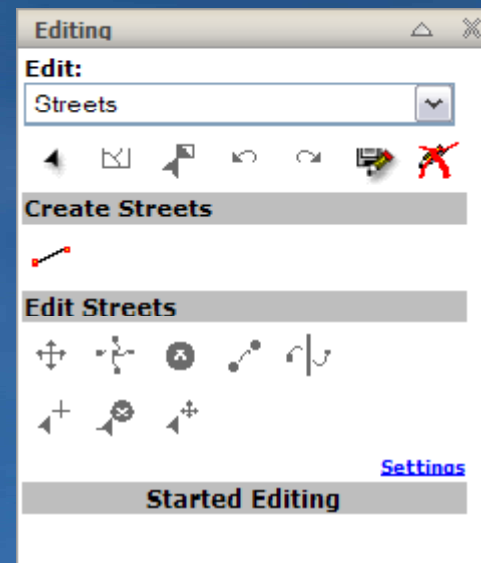
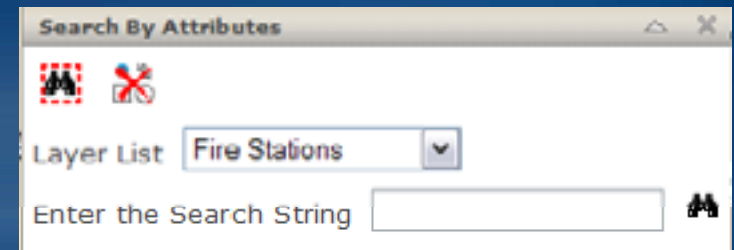
Please complete the session survey!

Session agenda

- **Overview of tasks and the task framework**
- **Customizing the tasks included with the ADF**
 - Map tools, Query
 - Edit
- **Build a custom task**
 - Implement a custom task
 - Implement parameters, commands (actions) and tools in the task
 - Displaying results to the user
 - Customize the look and feel of the task

Web ADF Tasks

- Tasks are objects that encapsulate business logic
- Configurable from Manager
- Out of the box tasks
 - MapTools
 - Geoprocessing
 - Search by attributes
 - Editing
 - Predefined query
 - Find place
 - Print (9.3)
- Custom tasks



Visual components of a task

Task menu links

Results panel

Result Details

The screenshot displays a web mapping application interface. On the left, there is a sidebar with a 'Results' panel and a 'Map Contents' panel. The 'Results' panel shows a list of search results for earthquakes, including 'Query Earthquakes (50)', 'Earthquake History (50)', and several specific locations like 'Near San Francisco, California'. The 'Map Contents' panel shows a list of map layers, including 'California', 'Earthquake History', 'Cities', 'Counties', 'Rivers', 'Lakes', 'Shaded Relief', and 'Land Background'. The main map area shows a map of California with numerous green pushpin markers representing earthquake locations. A 'Query Earthquakes' dialog box is open over the map, with a search criteria of 'Magnitude greater than: 6.0' and a 'Submit query' button. A 'Callout' box is also visible, showing details for 'Southern Sonoma County, California'. The application title is 'Web Mapping Application' and the status bar shows '100 50 0 100 Kilomet' and '© Copyright'.

Demo

- **Demo**
 - **Search Attributes task**

Customizing out of the box tasks

- **Tasks have “getter” and “setter” methods that allow developers to**
 - **Customize the way the task performs**
 - **Change the look and feel of the task**
- **Use faces-config.xml to customize on startup**
- **Subclass task class to customize in run time**

MapToolsTask

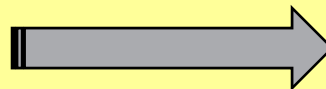
- MapToolsTask

- Common navigation and query tools

◆ `setTools()` ◆ `setIdentifiableLayers()`
◆ `setMapContinuousPan()` ◆ `setLayerDefinitions()`

faces-config.xml

```
<managed-bean-name>mapToolsTask</managed-bean-name>  
  
. . .  
<managed-property>  
  <property-name>tools</property-name>  
  <value>ZOOMIN,ZOOMOUT,PAN,IDENTIFY</value>  
  . . .
```

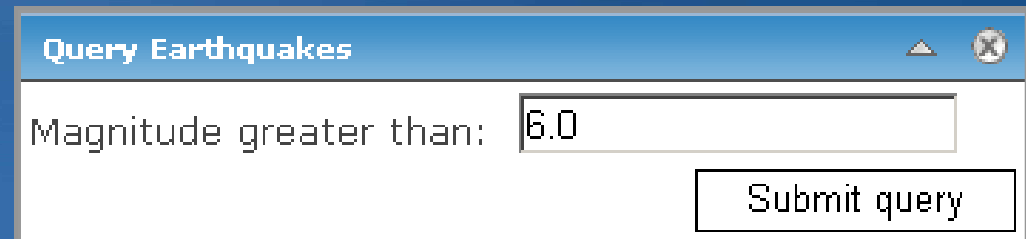
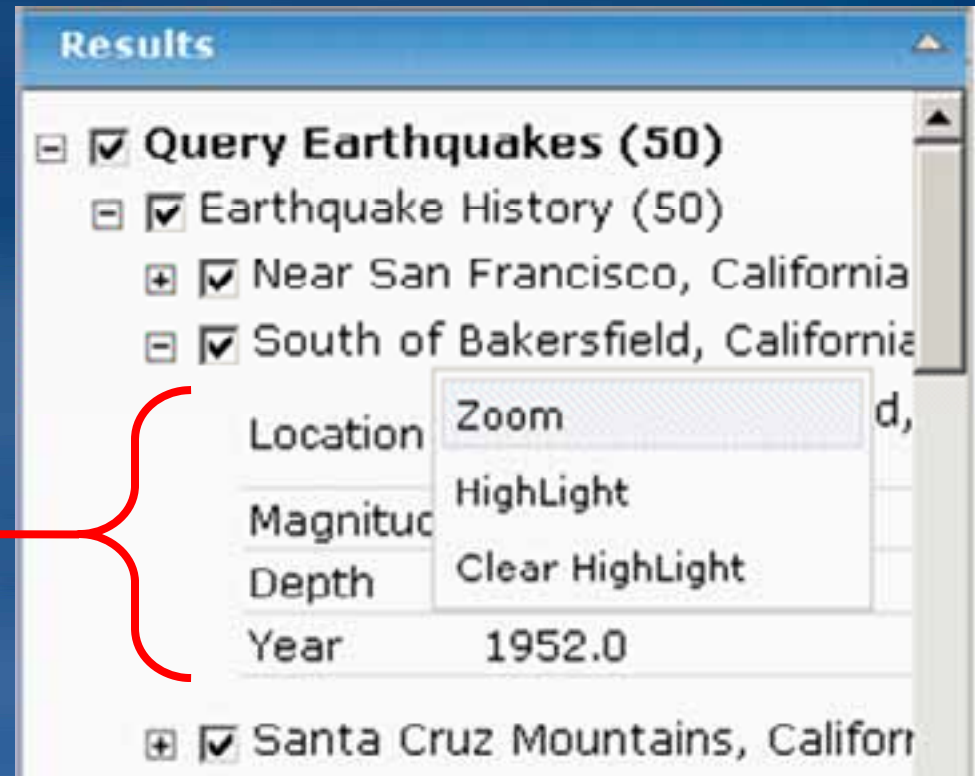


QueryAttributesTask

- QueryAttributesTask
 - Query a specified layer and attribute field

◆ displayFieldName
◆ returnFields
◆ fieldAliases

◆ displayName
◆ fieldLabel
◆ fieldValue
◆ findButtonText

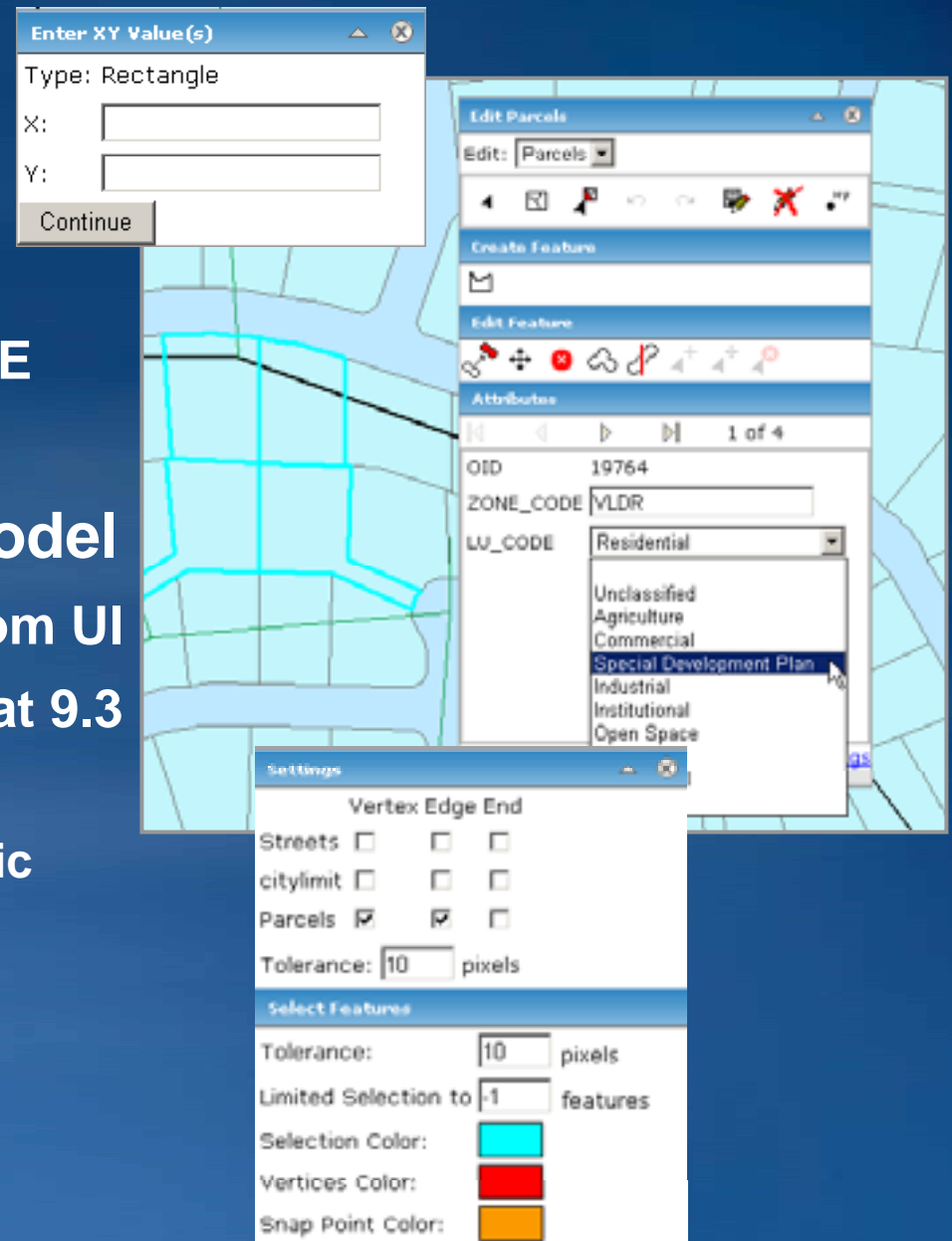


Demo

- Demo
 - MapToolsTask - changing default tools
 - queryAttributes – return fields, field alias, look and feel of Query UI

Edit task

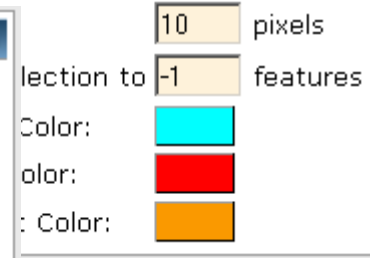
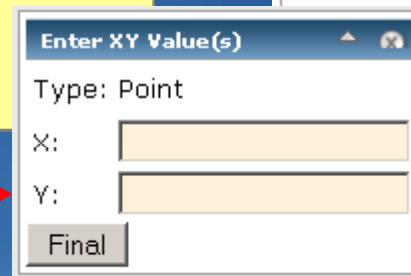
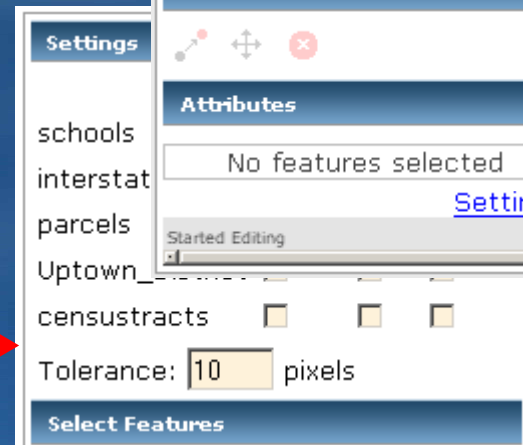
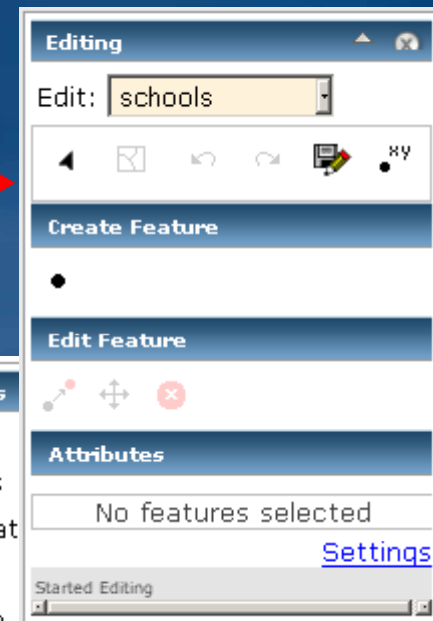
- Edit task
 - Edit layer served through SDE
 - Ability to set edit settings
- Different customization model
 - Business logic decoupled from UI
 - More customization options at 9.3
 - Customize UI
 - Extend existing business logic



Anatomy of the Editor Task

MapViewer.jsp

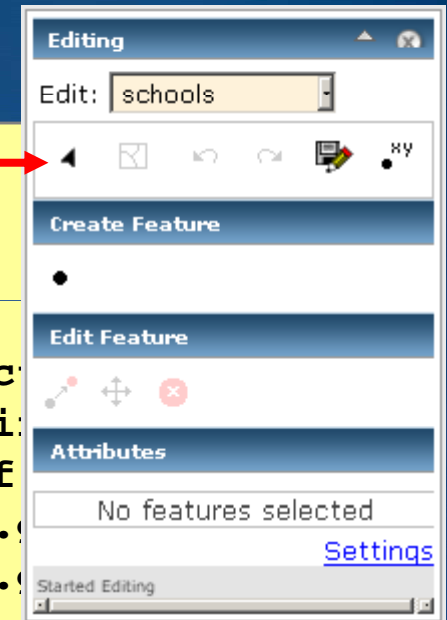
```
<div style="display:none">
  <div id="esri_editVersionDiv">
    <jsp:include page="version.jsp"
  /></div>
  <div id="esri_editDiv">
    <jsp:include page="edit.jsp"
  /></div>
  <div id="esri_editSettingsDiv">
    <jsp:include page="settings.jsp"
  /></div>
  <div id="esri_editXYDiv">
    <jsp:include page="xy.jsp"
  /></div>
</div>
```



Anatomy of the Editor Task

Edit.jsp

```
<a:button  
  id="select"  
  mapId="map1"  
  clientAction="EsriMapRectangle"  
  serverAction="#{mapContext.attributes.mapEditor.select"  
  defaultImage="./images/tasks/editing/selectfeature.gif"  
  hoverImage="./images/tasks/editing/selectfeatureU.gif"  
  selectedImage="./images/tasks/editing/selectfeatureD.  
  disabledImage="./images/tasks/editing/selectfeatureX.  
  tooltip="EditTask.TaskInfo.Tip.selectfeatures"  
/>
```



- How can we customize the editing task?

Demo

- **Customize the editing task**
 - Show the out-of-the box Editing task
 - Customize the UI
 - Extend the business logic of the Select tool

Task framework

- **Allows custom functionality to be implemented**
- **Tasks**
 - Objects that encapsulate business logic
 - Can contain one or many commands or tools
- **Advantages**
 - Tight integration with the ADF
 - Event handling with commands (actions) and tools
 - UI is implemented for you
 - Good way to encapsulate related functionality
 - **Task development is similar to implementing a standard JavaBean**
- **What are we going to do?**
 - Extend the Task Framework by implementing a custom task (action and tool)

Steps to implement a custom task

1. Create a standard Java Class

- Signature of method dictates a command or tool

2. Register the Java class as a managed bean in the faces-config

- Makes it available in the ADF

3. Add a task tag to the jsp, with reference to the managed bean

- Visually displays the UI to the user

Example: Implement a custom task (1)

① Create a standard Java Class

MyTask.java

```
public class MyTask {  
  
}
```

② (2A) Register Java class as a managed bean

faces-config.xml

```
<managed-bean>  
  <managed-bean-name>myTask</managed-bean-name>  
  <managed-bean-class>myPackage.MyTask</managed-bean-class>  
  <managed-bean-scope>none</managed-bean-scope>  
</managed-bean>
```

Example: Implement a custom task (2)

②(2B) Add Java class as an attribute of web context

faces-config.xml

```
<managed-bean-name>mapContext</managed-bean-name>
. . .
<property-name>attributes</property-name>
. . .
<map-entry> <key>myTask</key>

<value>#{myTask}</value>
```

③ Add the control to the jsp

- Value attribute = value of managed-bean-name in faces-config

xxx.jsp

```
<a:task value="#{mapContext.attributes.myTask}" mapId="map1" />
```

Anatomy of a task

- **Parameter**

- Provides inputs to a task
- Examples: Layer name, zoom factor

- **Command (action)**

- Executes business logic without user interaction with the map
- Example: Zoom to full extent

- **Tool**

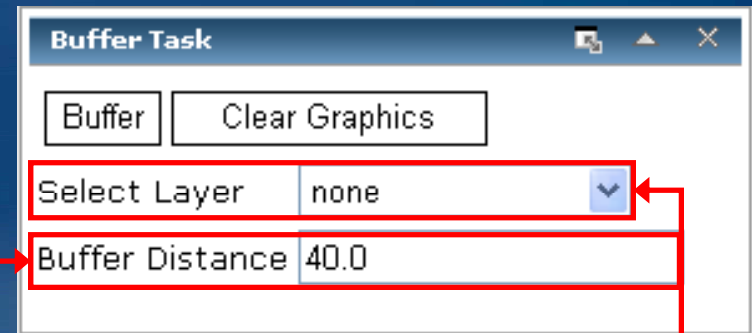
- Requires user interaction with the map
- Requires client-side action
- Examples: Identify, dynamic navigation (e.g., Zoom In/Out)

The screenshot shows a dialog box titled "Demo Task". At the top, there are four buttons: "Tool 1", "Tool 2", "Action 1", and "Action 2". Below these buttons are two parameter fields: "Param1" with a dropdown menu showing the value "1", and "Param2" with an empty text input field. Red rectangular boxes highlight the "Tool 1" and "Tool 2" buttons, and the "Param1" and "Param2" fields.

Example: Add parameters to a custom task

- Parameters

- Provide inputs for the task

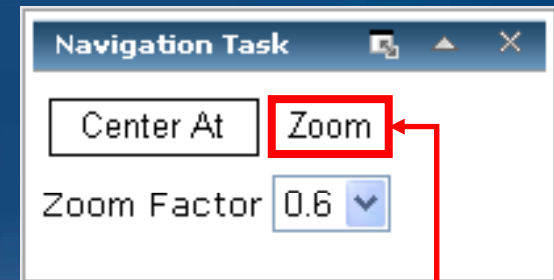


BufferTask.java

```
public class BufferTask {  
    double bufferDistance = 40;  
    public double getBufferDistance() { return bufferDistance; }  
    public void setBufferDistance(double bufferDistance) { . . . }  
    String selectLayer;  
    public String getSelectLayer() { return selectLayer; }  
    public void setSelectLayer(String selectLayer) { . . . }  
    public Map getSelectLayers() {return selectLayers;}  
}
```

Example: Add a command to a custom task

- Add a method with TaskEvent as argument
 - (com.esri.adf.web.faces.event)
 - Gives access to WebContext



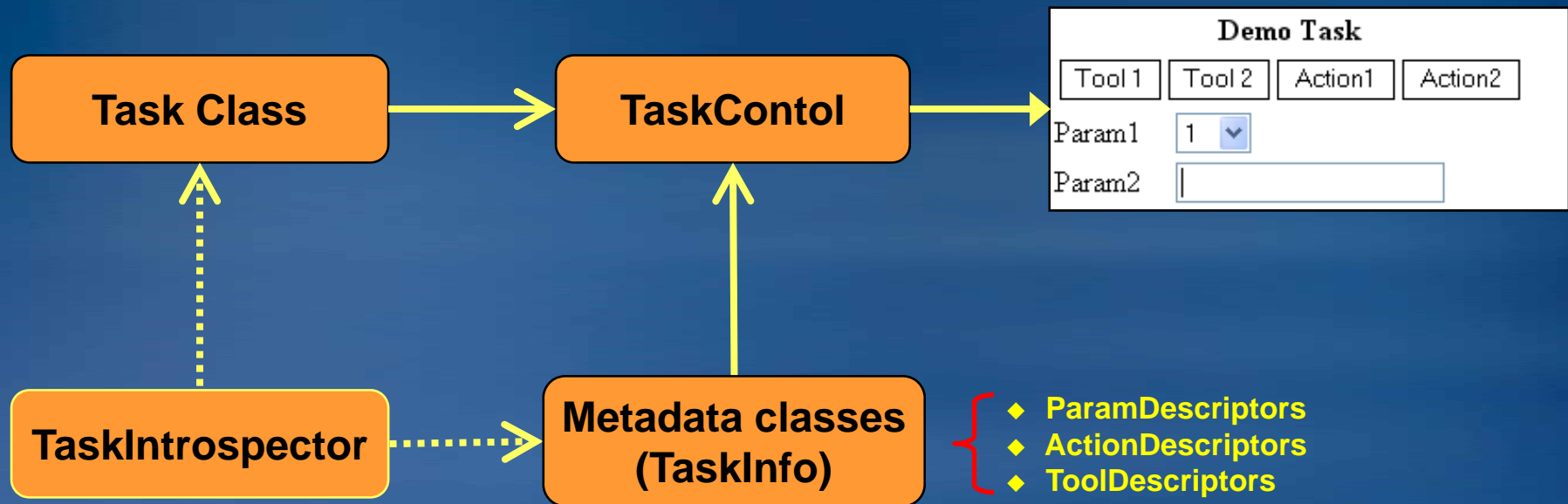
NavigationTask.java

```
public class NavigationTask {  
    public void zoom(TaskEvent event) {  
        . . .  
    }  
}
```

Demo

- **Demo**
 - **Build a custom task that contains a parameter and a command**

The task architecture



TaskInfo explained

- TaskInfo
 - Returns description objects of the task

TaskInfo.java

```
package com.esri.adf.web.data.tasks;

public interface TaskInfo {
    TaskDescriptor getTaskDescriptor();
    TaskParamDescriptorModel[ ] getParamDescriptors();
    TaskActionDescriptorModel[ ] getActionDescriptors();
    TaskToolDescriptorModel[ ] getToolDescriptors();
    TaskLayout[ ] getTaskLayout();
}
```


TaskInfo explained

(cont')

- **BeanInfo**
 - Returns description objects of the bean

BeanInfo.java

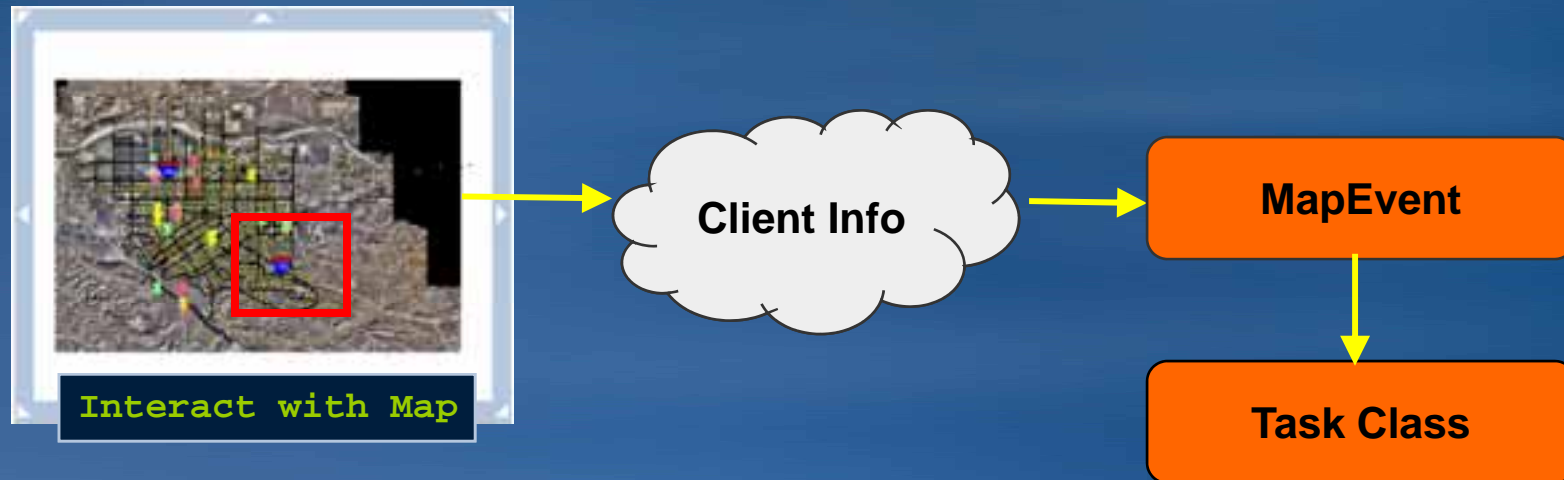
```
package java.beans;

public interface BeanInfo {
    BeanDescriptor getBeanDescriptor();
    PropertyDescriptor[ ] getPropertyDescriptors();
    MethodDescriptor[ ] getMethodDescriptors();

    . . .
}
```

Adding tools to a custom task

- Tools execute business logic based on user interaction with a map
 - Both client-side and server-side actions



- ① Add method with MapEvent as argument
- ② Create a TaskInfo class
- ③ Provide a TaskToolDescriptor

Example: Add a tool to a custom task (1)

- 1 Add method with MapEvent as argument



MyTask.java

```
public class MyTask {  
    public void selectCountries(MapEvent event) {  
        . . .  
    }  
}
```

MapEvent explained

- **MapEvent** (`com.esri.adf.web.faces.event`)
 - Gives access to important information
 - WebContext and WebGeometry
 - Type of geometry depends on user action on client



Example: Add a tool to a custom task (2)

② Create a TaskInfo class

- Extend SimpleTaskInfo

MyTaskInfo.java

```
public class MyTaskInfo extends SimpleTaskInfo {  
    . . . .  
}
```

Example: Add a tool to a custom task (3)

③ Provide a TaskToolDescriptor

- Client-side action: Controlled by JavaScript functions
 - EsriMapRectangle, EsriMapPan, EsriMapPoint, etc.

MyTaskInfo.java

```
public class MyTaskInfo extends SimpleTaskInfo {
    public TaskToolDescriptorModel[ ] getToolDescriptors() {

        return new TaskToolDescriptor[ ] {
            new TaskToolDescriptor(MyTask.class, "select",
                "Select", "EsriMapRectangle")};
        }
    }
}
```

Demo

- **Demo**
 - Implement a custom select tool
 - Display on graphics layer

Task results

- Task operations may generate results
 - Tools or commands
- Results can be arbitrary Java objects
 - Query results
 - Address candidates
 - Geoprocessing results
- Types of information
 - Display text
 - Result details
 - Actions that can be performed on results

The screenshot shows a 'Results' window with a tree view on the left and a 'Result Details' table on the right. The tree view includes 'Identify Result(s)', 'Buffer', 'Inputs', 'Outputs', and 'Messages'. The 'Identify Result(s)' folder is expanded, showing a list of actions: 'Utah', 'Zoom', 'HighLight', and 'Clear Graphic'. The 'Utah' action is selected, and its details are shown in the table below.

Name	Value
FID	36
Shape	Polygon
AREA	219813609472
PERIMETER	1974284.125
STATES#	36
STATES-ID	62
STATE_FIPS	49
STATE_NAME	Utah
STATE_ABBR	UT

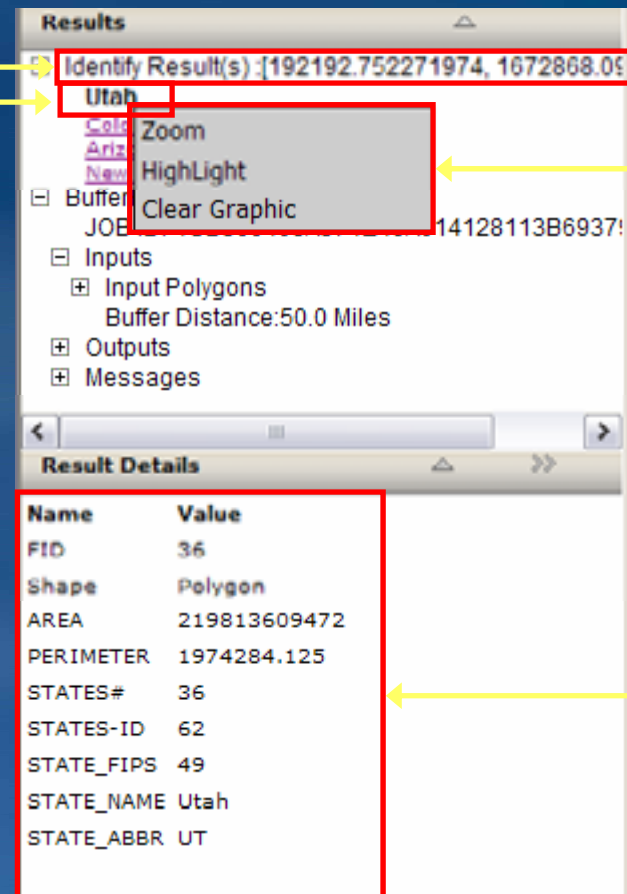
Task results

(cont')

- **WebResults**

- Container object for results
- Attribute to the Context
- Results displayed as tree by default
- Toc control is re-used

- `com.esri.adf.web.data.results`



`addResultsWithActionArray`

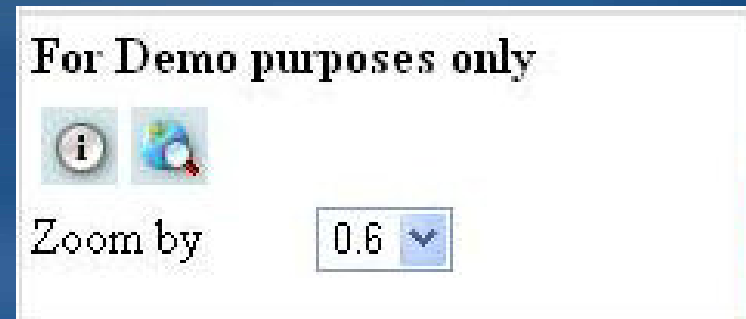
```
public ResultNode addResultsWithActionArray(java.lang.String groupHeader,  
java.util.List results,  
java.lang.String displayNameMethodName,  
java.lang.String detailsMethodName,  
java.lang.String[] actionMethodsNames)
```

Demo

- Demo
 - Display attributes in the task framework

Inside the TaskInfo Interface

- **Why implement a custom a TaskInfo class?**
 - **Layout**
 - Position of parameters, commands and tools
 - **Titles/text**
 - Name of task on title bar
 - Messages in the task UI
 - **Icons**
 - Commands and tools

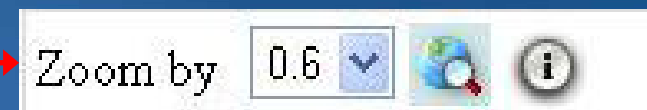
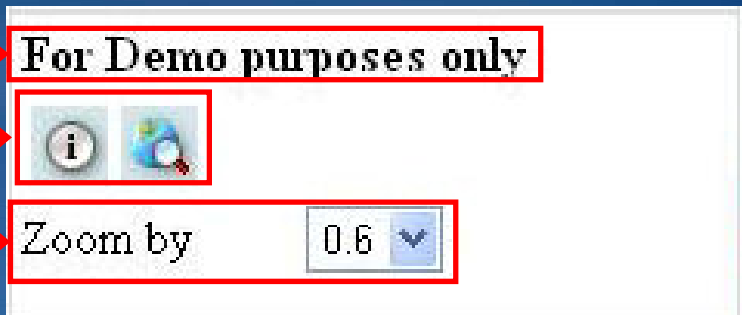


Inside the TaskInfo Interface

- Task class contains business logic
- TaskInfo class contains metadata
- TaskInfo class changes look and feel without affecting any business logic

TaskInfo.java

```
package com.esri.adf.web.data.tasks;  
  
public interface TaskInfo {  
    getTaskDescriptor();  
    getParamDescriptors();  
    getActionDescriptors();  
    getToolDescriptors();  
    getTaskLayout();  
}
```



Demo

- **Implement a custom TaskInfo class**

Summary

- Overview of tasks and the task framework
- Customizing the tasks included with the ADF
- Build a custom task
- Display results

Still have questions?

Additional Resources

Questions, answers and information...

- **Tech Talk**

- *Outside this room right now!*

- **Meet the Team**

- *Java Development team*

- Wednesday 6 – 7pm Oasis 2*

- **Other sessions**

- *Customizing Graphics and MapTips with the Java Web ADF*

- **ESRI Resource Centers**

- PPTs, code and video



resources.esri.com

- **Social Networking**



[www.twitter.com/
ESRIDevSummit](http://www.twitter.com/ESRIDevSummit)

facebook

[tinyurl.com/
ESRIDevSummitFB](http://tinyurl.com/ESRIDevSummitFB)

Want to Learn More?

ESRI Training and Education Resources

- **Instructor-Led Training**
 - [Developing Applications with ArcGIS Server Using the Java Platform](#)
- **Free Web Training Seminar**
 - [Building Applications with ArcGIS Server Using the Java Platform](#)