



Developing Custom Web Tasks using the Java Web ADF (Deep Dive)

David Cardella

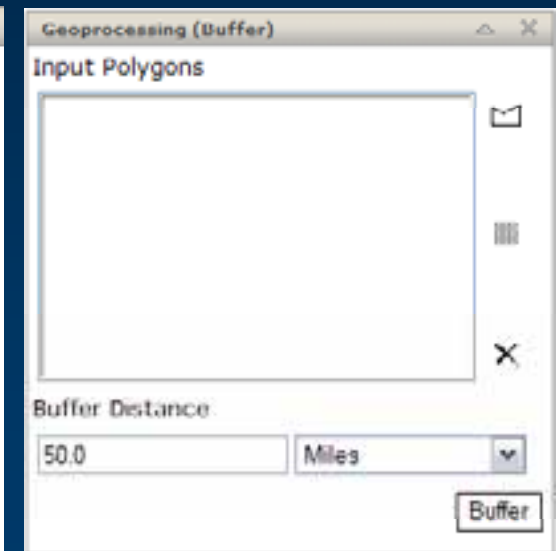
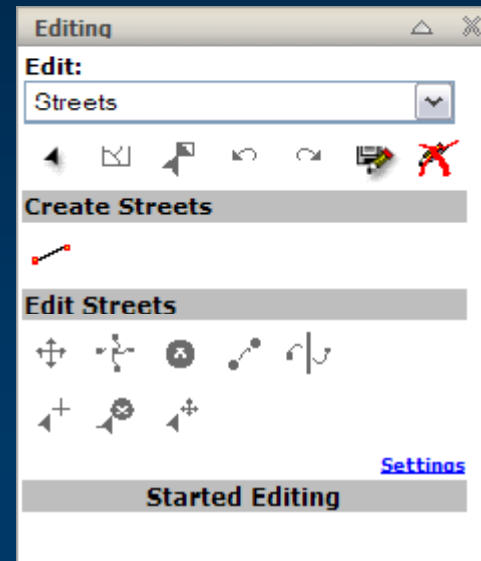
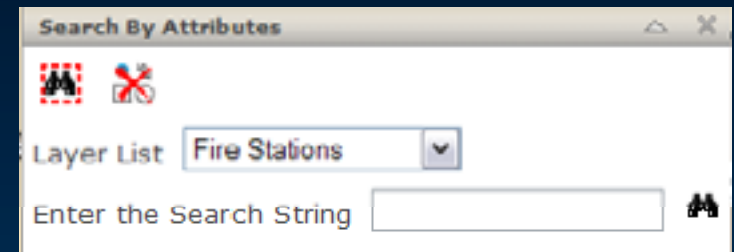
Keyur Shah

Presentation outline

- **Overview of tasks and the task framework**
- **Build a custom task**
 - Implement a custom task
 - Implement parameters, commands (actions) and tools in the task
 - 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
 - Navigation
 - Geoprocessing
 - Search by attributes
 - Editing
 - Find direction
 - Predefined query
 - Find place
 - Other ...
- Custom tasks



Demo

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

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 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>session</managed-bean-scope>  
</managed-bean>
```

Example: Implement a custom task (2)

③ Add the control to the jsp

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

xxx.jsp

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

faces-config.xml

```
<managed-bean>  
  <managed-bean-name>myTask</managed-bean-name>
```

Anatomy of a task

- **Parameter**

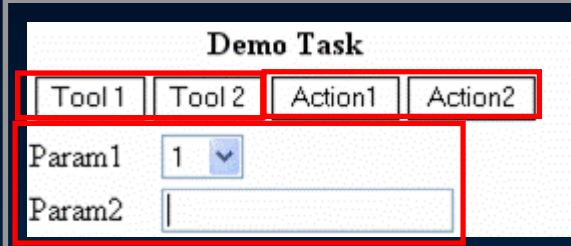
- Provides inputs to a task
- Examples: Layer name, zoom factor

- **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)

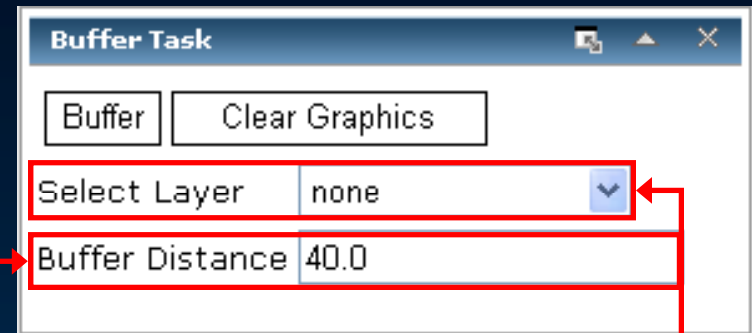


The screenshot shows a window titled "Demo Task". At the top, there are four buttons: "Tool 1", "Tool 2", "Action1", and "Action2". Below these buttons, there are two parameter fields: "Param1" with a dropdown menu showing the value "1", and "Param2" with an empty text input field. Red boxes highlight the buttons and the parameter fields.

Example: Add parameters to a custom task

- Parameters

- Provide inputs for the task

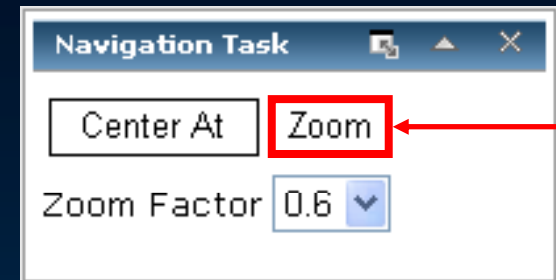


BufferTask.java

```
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



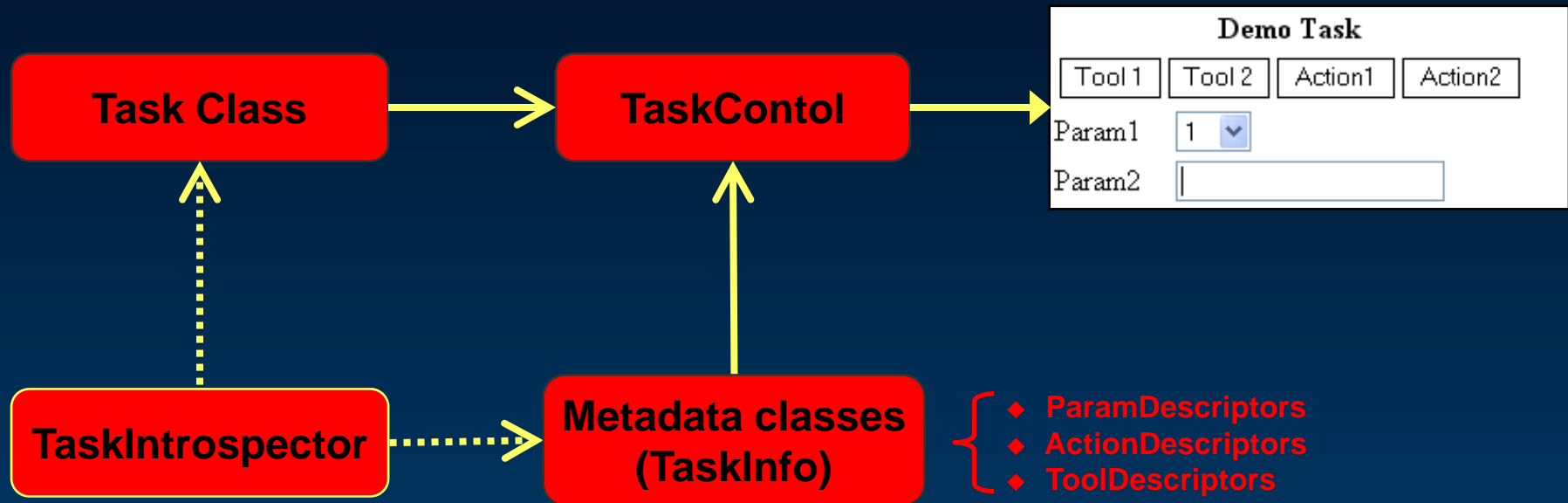
NavigationTask.java

```
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

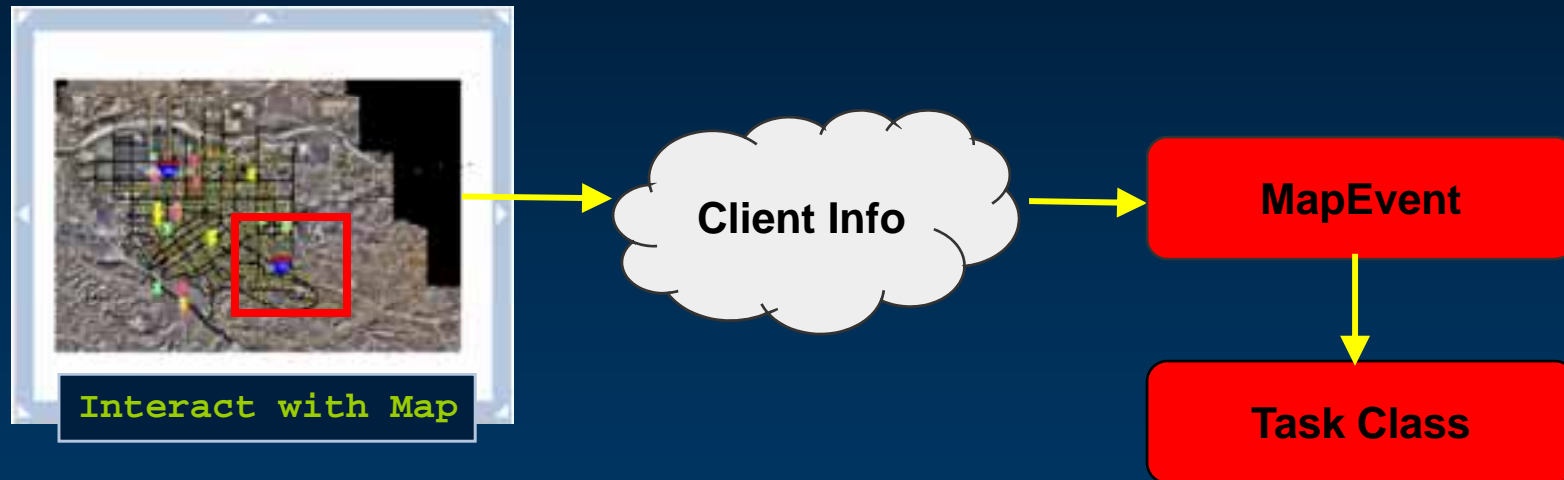
```
package java.beans;

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) {  
        . . .  
    }  
}
```

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.

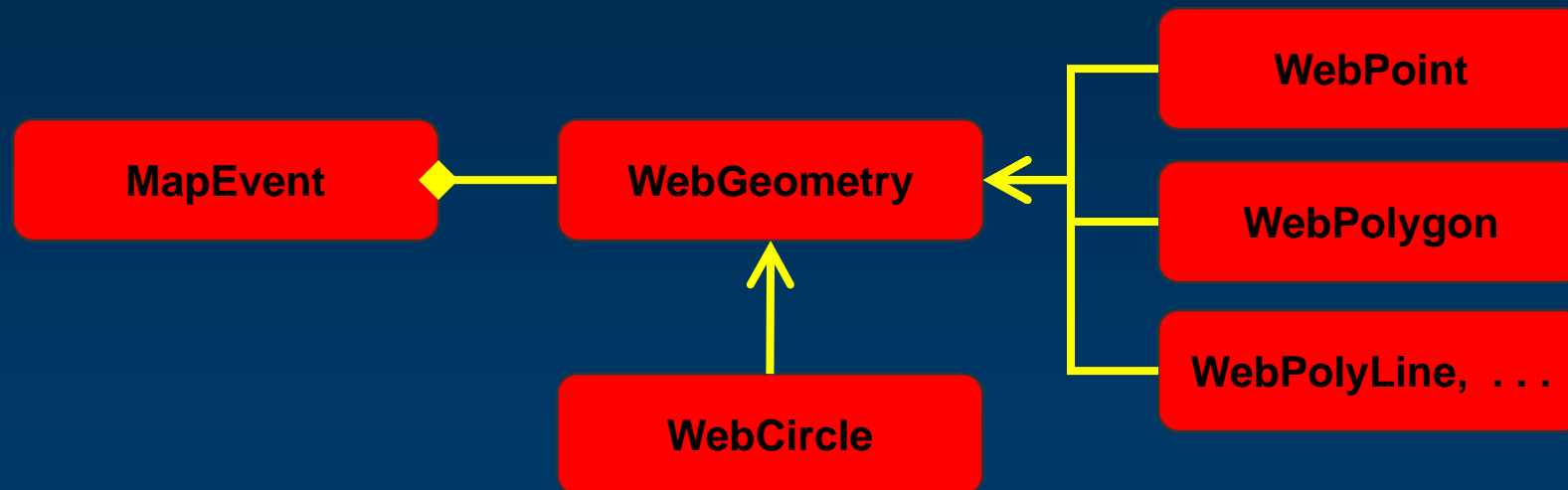
MyTaskInfo.java

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

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

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



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. The 'Identify Result(s)' folder is expanded, showing a list of results for 'Utah'. A context menu is open over the 'Utah' result, listing actions: 'Zoom', 'HighLight', and 'Clear Graphic'. Below the tree view is a 'Result Details' table.

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

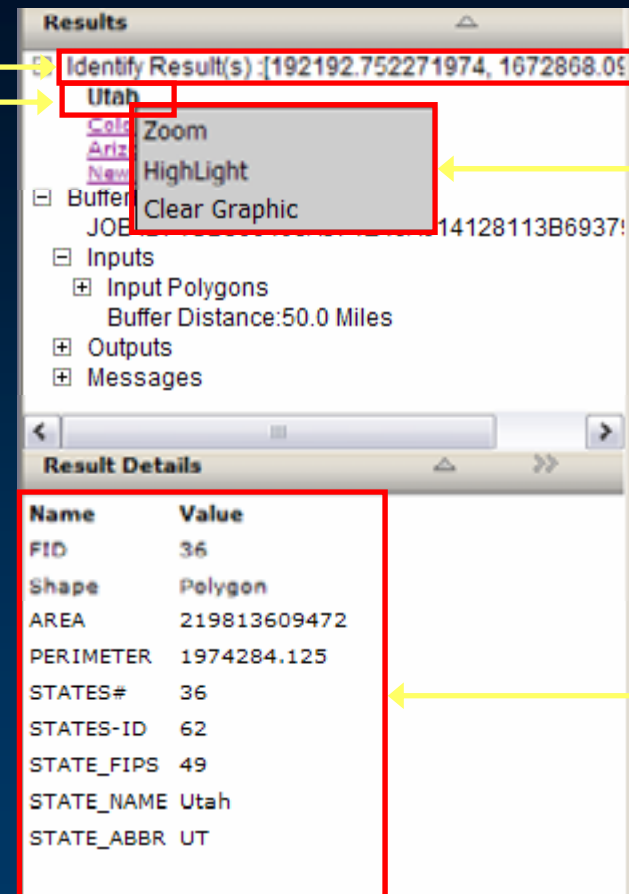
Task results

(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`

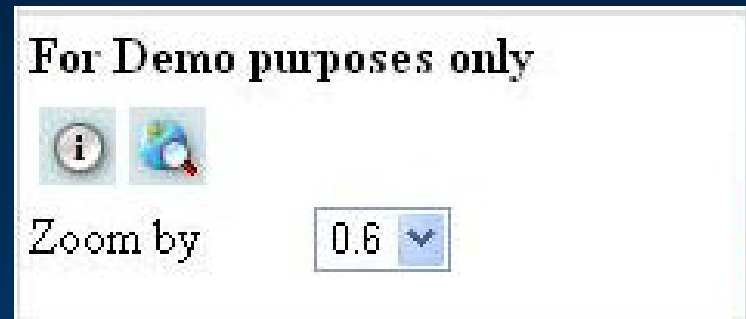
```
public ResultNode addResultsWithActionArray(java.lang.String groupHeader,  
java.util.List results,  
java.lang.String displayNameMethodName,  
java.lang.String detailsMethodName,  
java.lang.String[] actionMethodsNames)
```

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**

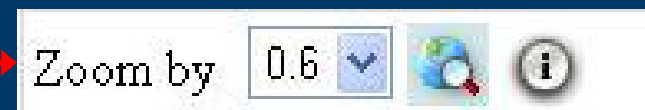
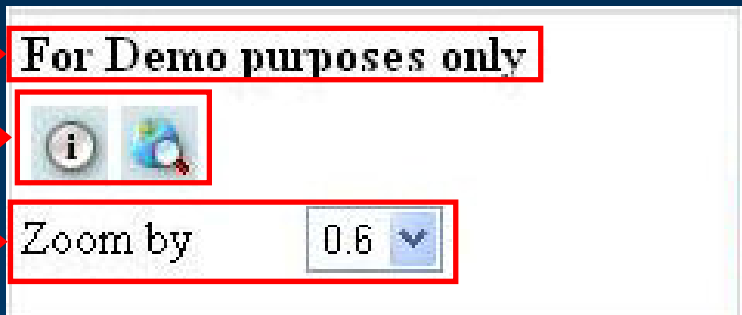


Inside the TaskInfo Interface

- 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**

Where do you go from here?

- **Developing Web Applications with ArcGIS Server using the Java platform**
- **EDN**
 - <http://edn.esri.com/java>