Leveraging GOS Map and Data Services for Search and Rescue Operations using NASA WorldWind Open Source 3D Visualization Platform

Technical Documentation

August 1, 2010

Funded by the FGDC CAP Grant 2009 Program
http://www.fgdc.gov/grants/2009CAP/projects/G09AC00106

World Wind Geospatial One-Stop Portal Technical Documentation

The latest version of the World Wind Geospatial One-Stop Portal is embedded into the World Wind Search and Rescue application, available at http://worldwind.arc.nasa.gov/java/apps/SARApp/SARApp6.jnlp. The source code for the portal is available in the World Wind Java nightly code drop, available here: http://builds.worldwind.arc.nasa.gov/. The GOS portal sources are contained in the package `gov.nasa.worldwind.applications.gos`.

The package `gov.nasa.worldwind.applications.gos` is a collection of components that enable applications to add Catalog Service for Web (CSW) browsing capabilities to any World Wind Java based application. The components are designed to use replaceable interfaces for communicating with the catalog. This enables applications to replace these interfaces with their own implementation and communicate with any catalog. Currently, the sole implementation of these interfaces communicates with the U.S. Government Geospatial One-Stop catalog http://geodata.gov.

There are four major components in this package:

- `gov.nasa.worldwind.applications.gos.GeodataWindowJPanel` - Swing panel enabling CSW catalog search and result display.
- `gov.nasa.worldwind.applications.gos.GeodataRecentChangesPanel` - Swing panel displaying a feed of the most recent changes in a CSW catalog.
- `gov.nasa.worldwind.applications.gos.awt.GeodataLayerPanel` - Swing panel displaying the World Wind `gov.nasa.worldwind.layers.Layer` objects added from a CSW catalog.
- `gov.nasa.worldwind.applications.gos.GeodataPortalFrame` - Swing frame combining the above three panels in a single tabbed window.

Applications can choose between integrating an entire GeodataPortalFrame, or embedding CSW browsing components directly in their application layout. The section **Using the CSW Components** demonstrates each use case.

These components support the following capabilities for interacting with CSW catalogs:

- CSW Search
- CSW Results in Swing Component
- CSW Results on Globe
- CSW Update Feed
- CSW Layer List
- Connecting to WMS Servers published in a CSW Catalog

Using the CSW Components

Using GeodataPortalFrame

GeodataPortalFrame is a `javax.swing.JFrame` that combines the three CSW components provided in this package - GeodataWindowJPanel, GeodataRecentChangesPanel, and GeodataLayerPanel - into a single tabbed window. Using GeodataPortalFrame is the simplest way to integrate full CSW browsing capabilities into an application. Here is an example usage of GeodataPortalFrame:

```
GeodataPortalFrame cswFrame = new GeodataPortalFrame();

// If the application has a World Wind window, register it with the
// CSW frame as follows:
WorldWindow wwd = ...; // Reference to application's World Wind window. May be null.
if (wwd != null)
{
    cswFrame.setWorldWindow(wwd);
}

cswFrame.setVisible(true);
```

Using GeodataWindowJPanel

GeodataWindowJPanel is a `javax.swing.JPanel` that provides the most basic CSW capabilities: search and result display. Using GeodataWindowJPanel directly provides more flexibility on how applications can integrate CSW browsing capabilities into their visual layout. Here is an example usage of GeodataWindowJPanel:

```
GeodataWindowJPanel cswPanel = new GeodataWindowJPanel();

// If the application has a World Wind window, register it with the
// CSW panel as follows:
WorldWindow wwd = ...; // Reference to application's World Wind window. May be null.
if (wwd != null)
{
    cswPanel.setWorldWindow(wwd);
}

// Add the CSW panel to application's layout. For example, an
// application could add the CSW panel to the "east" portion of a
// parent container's border layout as follows:
Container parent = ...; // Reference to the parent container.
parent.add(cswPanel, BorderLayout.EAST);
```

Extending the CSW Components

Connecting to Another Catalog

By default, GeodataWindowJPanel connects to the Geospatial One-Stop CSW catalog hosted at http://geodata.gov. The catalog URL can be replaced by adding the following property to the application's World Wind configuration file:

```
<Property name="gov.nasa.worldwind.gos.CSWServiceURI" value="http://www.mycswcatalog.com"/>
```

However, most CSW catalogs do not have a unified search or result model. CSW catalogs typically use different search terms, and a different result model. Therefore most catalogs require custom logic to execute a CSW search and parse the CSW response. To accomplish this, applications can replace the GeodataWindowJPanel's `gov.nasa.worldwind.applications.gos.GeodataController`. The GeodataController defines how a GeodataWindowJPanel behaves when a GeodataWindowJPanel's search button is pressed. Applications can extent `gov.nasa.worldwind.applications.gos.AbstractGeodataController` to avoid re-implementing the plumbing of performing a catalog search in a separate thread. Here is an example of replacing GeodataController to connect to a CSW catalog with a search or result model different from the geodata.gov catalog:

```
public class MyCSWController extends AbstractGeodataController
{
    protected void doExecuteSearch(final AVList params) throws Exception
    {
        // Create a string containing an xml-encoded CSW query string
        // appropriate for the CSW catalog we're connecting to.
        String cswQuery = this.buildCSWQueryString(params);

        // Construct a URI for a CSW get records request, and execute
        // an HTTP retrieval of that URL. We send the CSW query string
        // to the catalog as the HTTP POST content.
        URI cswURI = new CSWGetRecordsRequest(new URI("http://www.mycswcatalog.com")).getURI();
        URLRetriever retriever = new HTTPPostRetriever(cswURI.toURL(), cswQuery, null);
        retriever.call();

        // Our thread has been interrupted; abort this search operation.
        if (retriever.getState().equals(URLRetriever.RETRIEVER_STATE_INTERRUPTED))
            return;

        if (!retriever.getState().equals(URLRetriever.RETRIEVER_STATE_SUCCESSFUL))
        {
            String message = Logging.getMessage("generic.RetrievalFailed", uri.toString());
            Logging.logger().severe(message);
```

```
            throw new WWRuntimeException(message);
        }

        if (retriever.getBuffer() == null || retriever.getBuffer().limit() == 0)
        {
            String message = Logging.getMessage("generic.RetrievalReturnedNoContent", uri.toString());
            Logging.logger().severe(message);
            throw new WWRuntimeException(message);
        }

        // Parse the CSW response as an XML DOM document, and create a
        // gov.nasa.worldwind.applications.gos.RecordList from the response.
        Document responseDoc = WWXML.openDocument(WWIO.getInputStreamFromByteBuffer(retriever.getBuffer()));
        final RecordList recordList = this.createRecordList(responseDoc);

        SwingUtilities.invokeLater(new Runnable()
        {
            public void run()
            {
                getGeodataWindow().setRecordList(recordList, params);
            }
        });
    }
}

...

GeodataWindowJPanel cswPanel = new GeodataWindowJPanel();
cswPanel.setGeodataController(new MyCSWController());
```

Changing How Results are Displayed

GeodataWindowJPanel displays the following properties of each `gov.nasa.worldwind.applications.gos.Record` in the `gov.nasa.worldwind.applications.gos.RecordList` created by the GeodataWindowJPanel's GeodataController:

- Title
- Type
- Sector
- Abstract
- Website Online Resource
- Metadata Online Resource
- Service Status Online Resource
- WMS Capabilities Online Resource

The display of CSW results on the GeodataWindowJPanel is handled by creating a Swing `javax.swing.JComponent` for each CSW record result. To extent or replace the default result component, applications can replace the GeodataWindowJPanel's `gov.nasa.worldwind.applications.gos.awt.RecordListPanel`. Here is an example of replacing the default Swing components created for each CSW record:

```
public class MyRecordPanel extends JPanel
{
    public MyRecordPanel(Record record, GlobeModel globeModel)
    {
        // Create custom panel layout for the specified CSW record.
    }
}

...

public class MyRecordListPanel extends RecordListPanel
{
    protected JComponent createRecordComponent(Record record, GlobeModel globeModel)
    {
        return new MyRecordPanel(record, globeModel);
    }
}

...
```

```
GeodataWindowJPanel cswPanel = new GeodataWindowJPanel();
cswPanel.setRecordListPanel(new MyRecordListPanel());
```

The display of CSW results on the World Wind Globe is handled by `gov.nasa.worldwind.applications.gos.GeodataWindowJPanel.assembleGlobeRenderables()`. By default, this creates a `gov.nasa.worldwind.render.SurfacePolygon` and an `gov.nasa.worldwind.render.Annotation` for each CSW result. To extend or replace the default behavior, applications can extend GeodataWindowJPanel and override the method `assembleGlobeRenderables`. Here is an example of extending GeodataWindowJPanel to customize how CSW results are displayed on the World Wind Globe:

```
public class GeodataWindowJPanel extends GeodataWindowJPanel
{
    protected void assembleGlobeRenderables()
    {
       if (this.recordList == null || this.recordList.getRecords() == null)
           return;

       for (Record record : this.recordList.getRecords())
       {
           // Create a representation for each record, then add that representation to the World Wind Globe.
       }
    }
}
```