ORACLE FUSION MIDDLEWARE

feature overview

Oracle AS MapViewer 10g

September 2005

Introduction

Oracle Application Server MapViewer (or simply, MapViewer) provides powerful geospatial data visualization and reporting services. Written purely in Java and run in a J2EE environment, MapViewer provides web application developers a versatile means to integrate and visualize business data with maps. MapViewer uses the basic capability included with Oracle10g (delivered via either Oracle Spatial or Locator) to manage geographic mapping data. MapViewer hides the complexity of spatial data queries and the cartographic rendering process from application developers.

Overview

This document describes the new features that have been introduced with MapViewer in Application Server 10g. These features are introduced below, those that introduce extensive new functionality are discussed in more detail in the following sections.

New and improved features available in MapViewer v10.1.2 include:

- Java Client API—An enhanced Java client API provides convenient access to most MapViewer functions from a Java application.
- JSP Tag Library—A set of custom JSP tags can be used to develop JSP applications using MapViewer features.
- Transparent PNG, and JPEG Format Support.
- Map Legend Support—The content and layout of a map legend can also be customized.
- Query Capabilities with Nonspatial Attributes—A set of methods in the Java Client API is provided for querying nonspatial attributes based on locations.
- Support for Georeferenced Images—Define image themes that can be rendered

along with the normal vector themes.
- Support for Spatial 10g Network and Topology data models.
- Workspace Manager support—Themes can be from a specified workspace.
- Multiple datasources in a map request—Themes in a map can be from different datasources.
- SVG map support—MapViewer can now output a SVG map that can be interactively viewed in a SVG client.
- OGC Web Map Server 1.1.1 interface—MapViewer now has a WMS 1.1.1 interface.
- Dynamic coordinate system transformations—Themes with different coordinate systems will be transformed into a common coordinate system prior to rendering.
- Performance Improvements—MapViewer utilizes an internal geometry and image cache for predefined themes.

Java Application Programming Interfaces

MapViewer exposes its services through an XML API, which application developers can use when formulating XML map requests for MapViewer and parsing XML responses from it. Through the XML API, an application can:

- Customize a map's data coverage, background color, size, image format and title, and other characteristics.
- Display a map with predefined base map, plus any other predefined themes not included in the base map.
- Display a map with dynamically defined themes, with each theme's data retrieved from a user-supplied SQL query.
- Display a map with one or more individual features that the application may have obtained from other sources.
- Through the XML response, obtain the URL to the generated map image, or the actual binary image data itself, plus the minimum bounding rectangle of the data covered in the generated map.

Since MapViewer release 9.0.4, the XML API is wrapped into a Java API so that Java application developers can easily use the new API instead of manipulating XML map requests directly.

JSP Tag Libraries

Through an XML-like syntax, the JSP tags provide a set of important MapViewer capabilities, such as setting up a map request, zooming, and panning, as well as identifying nonspatial attributes of features selected by a user with the mouse.

Creating JSP files is often easier and more convenient than using the XML or JavaBean-based API, although the latter two approaches give greater flexibility and

control over the program logic. All MapViewer JSP tags in the same session scope share access to a single MapViewer bean.

The tags enable you to perform several kinds of MapViewer operations:

- Create the MapViewer bean and place it in the current session: init tag, which must come before any other MapViewer JSP tags.
- Set parameters for the map display and optionally a base map: setParam tag.
- Add themes and a legend: addPredefinedTheme, addJDBCTheme, importBaseMap, and makeLegend tags.
- Get information: getParam, getMapURL, and identify tags.
- Submit the map request for processing: run tag.

The table below provides a summary of the individual tags and a synopsis of each tags function.

Tag Name	Explanation
init	Creates the MapViewer bean and places it in the current session. Must come before any other MapViewer JSP tags.
setParam	Specifies one or more parameters for the current map request.
addPredefinedTheme	Adds a predefined theme to the current map request.
addJDBCTheme	Adds a dynamically defined theme to the map request.
importBaseMap	Adds the predefined themes that are in the specified base map to the current map request.
makeLegend	Creates a legend (map inset illustration) drawn on top of the generated map.

Tag Name	Explanation
getParam	Gets the value associated with a specified parameter for the current map request.
getMapURL	Gets the HTTP URL for the currently available map image, as generated by the MapViewer service.
identify	Gets nonspatial attribute (column) values associated with spatial features that interact with a specified point or rectangle on the map display, and optionally uses a marker style to identify the point or rectangle.
run	Submits the current map request to the MapViewer service for processing. The processing can be to zoom in or out, to recenter the map, or perform a combination of these operations.

Map Legends

A map legend is an inset illustration drawn on top of the map and describing what various colors, symbols, lines, patterns, and so on represent. The legend provides a key or an index that ties specific meaning to the map and its symbols. The appearance of a map can be customized in the following ways:

- Customize the background and border style
- Have one or more columns in the legend
- Add space to separate legend entries
- Indent legend entries
- Use any MapViewer style, including advanced styles

The example below is an excerpt from a MapViewer request that includes a legend.

```
<?xml version="1.0" standalone="yes"?>
<map_request
             basemap="density_map"
             datasource = "mvdemo">
  <center size="1.5">
     …
  </center>
  <legend bgstyle="fill:#ffffff;fill-opacity:128;stroke:#ff0000"
          position="NORTH_WEST">
    <column>
      <entry text="Map Legend" is_title="true" />
      <entry style="M.STAR" text="center point" />
      <entry style="M.CITY HALL 3" text="cities" />
      <entry is_separator="true" />
      <entry style="C.ROSY BROWN STROKE" text="state boundary" />
      <entry style="L.PH" text="interstate highway" />
      <entry text="County population:" />
```

```
      <entry style="V.COUNTY_POP_DENSITY" tab="1" />
    </column>
  </legend>
  <themes>
    …
  </themes>

</map_request>
```

Querying Nonspatial Attributes

It is often necessary to query nonspatial attributes that are associated with the spatial features being displayed in the current map image. For example, assume that a map request is issued to draw a map of all customer locations within a certain county or postal code. The next logical step is to find more information about each customer being displayed in the resulting map image. In a typical situation the user clicks on or identifies a feature displayed on the map to find out more (nonspatial attributes) about the feature This action can be essentially implemented using a query with the desired nonspatial attributes in its SELECT list, and a spatial filter as its WHERE clause. The spatial filter is an Oracle Spatial SQL operator that checks if the geometries in a table column (the column of type SDO_GEOMETRY in the customer table) spatially interact with a given target geometry (in this case, the user's mouse-click point). The spatial filter in the WHERE clause of the query selects and returns only the nonspatial attributes associated with the geometries that are being clicked on by the user.

You will need to call an Oracle Spatial operator to perform the filtering; however, you can use the MapViewer bean-based API to obtain information, and to preassemble the spatial filter string to be appended to the WHERE clause of your query. The identify method simplifies the task even further.

Support for Georeferenced Images

An image theme is a special kind of MapViewer theme useful for visualizing geographically referenced imagery (raster) data, such as from remote sensing and aerial photography. Image data can be used to communicate specific information that is critical in applications such asset management and in many transportation related applications.

You can define an image theme dynamically or permanently (as a predefined theme) in the database. You can use image themes with vector (non-image) themes in a map. The figure below shows a map in which an image theme (showing an aerial photograph of part of the city of San Francisco) is overlaid with themes showing several kinds of roadways in the city.

Improving Performance

To tune MapViewers performance the memory cache and disk cache that MapViewer uses for spatial geometry objects can be customized. This is accomplished by using the <spatial_data_cache> element. For example:

```
<spatial_data_cache    max_cache_size="64"
                       max_disk_cache_size="512"
                       disk_cache_path="/var/tmp"
/>
```

You can specify the following information as attributes of the <spatial_data_cache> element:

- The max_cache_size attribute specifies the maximum number of megabytes (MB) of in-memory cache.

  Default value: 64

- The max_disk_cache_size attribute specifies the maximum number of megabytes (MB) of disk cache.

  Default value: 512
- The disk_cache_path attribute specifies the temporary disk path where the spooled cache will be located.

  Default value: location specified for the Java environment variable java.io.tmpdir

The spatial data cache is always enabled by default, even if the element is commented out in the configuration file. To completely disable the caching of spatial data, you need to specify the max_cache_size attribute value as 0 (zero).

Conclusion

MapViewer is a tool available as a component of the Application Server or as an extension to the JDeveloper environment. MapViewer is used to improve the delivery of complex geographic data and mainstream business information to applications and business intelligence tools. The current 10.1.2 version of MapViewer includes a range of new features that support functional improvements (Legends, JSP tags, support for geographic images etc.) and performance and manageability features designed to improve the ease-of-use of this important map making extension to the location-enabled infrastructure in the Oracle platform.

Top of Page

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065

Worldwide Inquiries:
+1.650.506.7000
Fax +1.650.506.7200
http://www.oracle.com/