

How to run the MapViewer 11g Secure Map Rendering demo

In this demo, you will learn how to enforce access control on the CUSTOMERS table that comes with the MVDEMO sample data set. Each customer record stored in this table is assigned one of the 4 account managers, *alex*, *goerge*, *max* or *stacy* (in the *account_mgr* column). When one of these account managers logs into the Application Server or standalone OC4J and requests a map that displays the data from this table, only those customers that are assigned to him or her will be displayed. If you log into AS or standalone OC4J as any other user, you will not be able to see any customer on your map.

The following are the basic steps for getting this demo up and running.

Note that throughout this document, unless otherwise noted, the term **user** (and sometimes **authenticated user**) refers to the application or web user that logs into Application Server or standalone OC4J. More specifically, it must be the name of one of our imaginary account managers, *alex*, *george*, *max* or *stacy*.

1 Adding access control to the CUSTOMERS table

Assuming you have imported the latest MVDEMO sample data set into the database user *mvdemo*. All you need to do now is log into the database as user *mvdemo*, and run the **mvdemo-sec.sql** script (found in the latest MVDEMO sample data zip file), like this:

```
SQL> @mvdemo-sec.sql
```

You can check out the script for details. It basically performs the following tasks in the MVDEMO schema:

1. Modifies the existing CUSTOMERS table to add a new column *account_mgr*.
2. Updates each record in CUSTOMERS to assign an account manager, by setting *account_mgr* to one of the 4 names: *alex*, *george*, *max* or *stacy*.
3. Creates a PL/SQL package named **web_user_info**. This package is central to this Secure Map Rendering demo.
4. Creates a view *CUSTOMERS_VIEW* on the CUSTOMERS table, with the following predicate: **where account_mgr = web_user_info.get_user**. This restricts the access to the CUSTOMERS table by the web user currently logged in.
5. Inserts an entry into the USER_SDO_GEOM_METADATA view for *CUSTOMERS_VIEW*. This is required in order for MapViewer to recognize this view as being 'spatially enabled'.
6. Creates a MapViewer theme named MYCUSTOMERS based on the view *CUSTOMERS_VIEW*.

From this point on, whenever MapViewer is to display the MYCUSTOMERS theme, it will only get those records that are accessible by the currently logged in web user. It will receive no records if there is no authenticated user exists in the current web session, or the user is not one of the aforementioned account managers.

Note that you can actually move the original table, CUSTOMERS, to a different database schema, and grant read-only access to the MVDEMO database user. Then follow the same steps from step 4 and on to create the view and theme in MVDEMO. This approach allows you to store all of your sensitive spatial data tables in a separate schema, while MapViewer connects to a database schema (MVDEMO in this case) that contains only public spatial data (such as those for generating base maps) and views that implement access control based on user identity.

2 Adding web users

This step adds the four account managers as authenticated web users to the OC4J container. Please consult your Oracle Application Server or standalone OC4J's documentation for instructions. The following shows how to add the four new users to the *users* group/role in a standalone OC4J version 10.1.3.

1. Go to the EM web stie of your standalone OC4J
2. Select Administration tab once logged in.
3. Click on the "Go To Task" icon for the task "Security > Security Providers".
4. Click on the "Instance Level Security" button
5. Click on the "Realms" tab
6. Click on the numeric value of the "Users" cell of the "jazn.com" table row. This leads to a page where all existing users for the OC4J home instance are listed.
7. Click on the "Create" button to create a new user *alex*. Make sure the *users* role is assigned to this new user. If the *users* role does not exist, you can go back to step 6 and click on the "Roles" and "jazn.com" cell value to create this role.
8. Repeat step 7 to create users *george*, *max* and *stacy*, all with the "users" role.

3 Defining a secure data source

Deploy MapViewer to Oracle Application Server or standalone OC4J if not already done so. Once MapViewer is up and running, you will need to create a data source named "MVDEMO" for the MVDEMO sample data set. This data source needs to be added to the mapViewerConfig.xml file, NOT as a dynamically created data source (which does not support secured data sources).

In your mapViewerConfig.xml file, add the following data source definition, note the line in bolded text:

```
<map_data_source name="mvdemo"  
    jdbc_host="your db host name"  
    jdbc_sid="your db sid"  
    jdbc_port="your db port"
```

```
jdbc_user="mvdemo"  
jdbc_password="!mvdemo"  
jdbc_mode="thin"  
number_of_mappers="3"  
allow_jdbc_theme_based_foi="false"  
plsql_package="web_user_info"
```

/>

This data source looks and behaves just like any other data source of MapViewer's, except the additional attribute named "**plsql_package**". This is where you specify the name of the PL/SQL package created by the mvdemo-sec.sql script in step 1. The presence of this attribute automatically turns this data source into a Secure Data Source, and MapViewer will always ensure the authenticated user's identity is set for every database session on this data source.

So what if you do not specify the plsql_package attribute for the above data source? Does this mean the data in the CUSTOMERS table is no longer protected? Of course not. If a map request is issued against the table via a "non-secure" data source's connection, MapViewer will not be setting the web user identity in the database, which in turn means all access to the view CUSTOMERS_VIEW will return no data as the user identity information in the PL/SQL package will be null in such cases.

4 Accessing the demo page

Once a secure data source is defined in the MapViewer config file, you will need to restart the OC4J container to pick up the newly added web users and also for MapViewer to register the new data source. After that you can visit the following web page to test the secure rendering demo (change the host and port to those of your particular site):

<http://localhost:8888/mapviewer/demo/secure-mapping.html>.

When first opening this page, you will be prompted to log in. Make sure you log in as one of the four account managers. If the log in is successful, you will be presented with a simple map request that displays the customer data on a base map. Simply click Submit and you should see only those records that are visible by the current user/account manager. On this page there is also a simple Oracle Maps demo that displays the same set of customer data as interactive theme-based FOIs.

To change the web user, simply exit your browser (which terminates current authenticated user session), restart the browser, then open the above demo page again and log in as a different account manager.

5 Behind the scene

This section is for those wishing to know more about how visiting `secure-mapping.html` page triggers the login prompt. In order to force a log in when accessing this page, your MapViewer's `web.xml` file actually contains the following security constraints:

```
<!-- For Secure Map Rendering demos only -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Secure mapping demo page</web-resource-name>
    <description>MapView secure map rendering demo</description>
    <url-pattern>/demo/secure-mapping.html</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>secure_maps_role</role-name>
  </auth-constraint>
</security-constraint>

...

<!-- For Secure Map Rendering demos only -->
<security-role>
  <role-name>secure_maps_role</role-name>
</security-role>
```

What the above does, is telling the J2EE container to always prompt for login whenever user first attempts to access the page `/mapviewer/demo/secure-mapping.html`. It further requires that the authenticated user must be in the logic role `secure_maps_role`. So even if you login in as a legitimate OC4J user, such as `oc4jadmin`, but if this user is not in the role `secure_maps_role`, the container will consider it an invalid login and force you to retry.

So how do you map the logic role `secure_maps_role`, which is MapViewer's own, to the `users` role of the container (which is what `alex`, `george`, `max` and `stacy` has)? This is specified in an OC4J-proprietary deployment descriptor file `orion-web.xml` that ships with MapViewer. Specifically, this file contains the following mapping:

```
<!-- for Secure Map Rendering demos only -->

<security-role-mapping name="secure_maps_role">
  <group name="users" />
</security-role-mapping>
```