ORACLE

Identity Propagation for REST using Oracle WebService Manager 12.1.2

August 2013

Step-by-Step Instruction Guide

Author: Prakash Yamuna

Oracle Corporation

Table of Contents

1 Getting Started

1.1 Pre-Requisites

This How-To guide assumes that you have already downloaded and installed the following products/components.

- Download and install FMW 12.1.2 – this includes Oracle WebService Manager 12.1.2.
- Download and install Database 11.2.0.3
- Download JDeveloper 12.1.2
- JDK7 is preinstalled

1.2 Install Locations

This How-To does not provide installation instructions for the pre-requisite components.
You can consult the following how-to for installing FMW 12.1.2:
http://www.oracle.com/technetwork/middleware/webservices-manager/owsm-installation-12c-1971739.pdf

You can also consult the appropriate Install guides. OWSM documentation can be found at:
http://docs.oracle.com/middleware/1212/owsm/index.html

The components in this How-To are installed at the following locations:

Component	Install location
Oracle Weblogic 12.1.2	D:\oracle_12.1.2\wlserver_10.3
Oracle Web Services Manager (OWSM) 12.1.2	D:\oracle_12.1.2\oracle_common
Oracle Enterprise Manager (EM) 12.1.2	D:\oracle_12.1.2\oracle_common
JDeveloper	D:\oracle_12.1.2\jdeveloper
JDK	D:\Java\jdk1.7.0_15

1.3 Topology

This How-To uses a single domain. The domain includes a single weblogic server. The steps provided in this How-To can vary based on Topology.

- Domain Name: base_domain
- Weblogic Server: AdminServer

1.4 Install & Topology Verification

Start the Admin Server
Navigate to: D:\oracle_12.1.2\user_projects\domains\base_domain\bin

1.4.1 Verify all Product Consoles are reachable

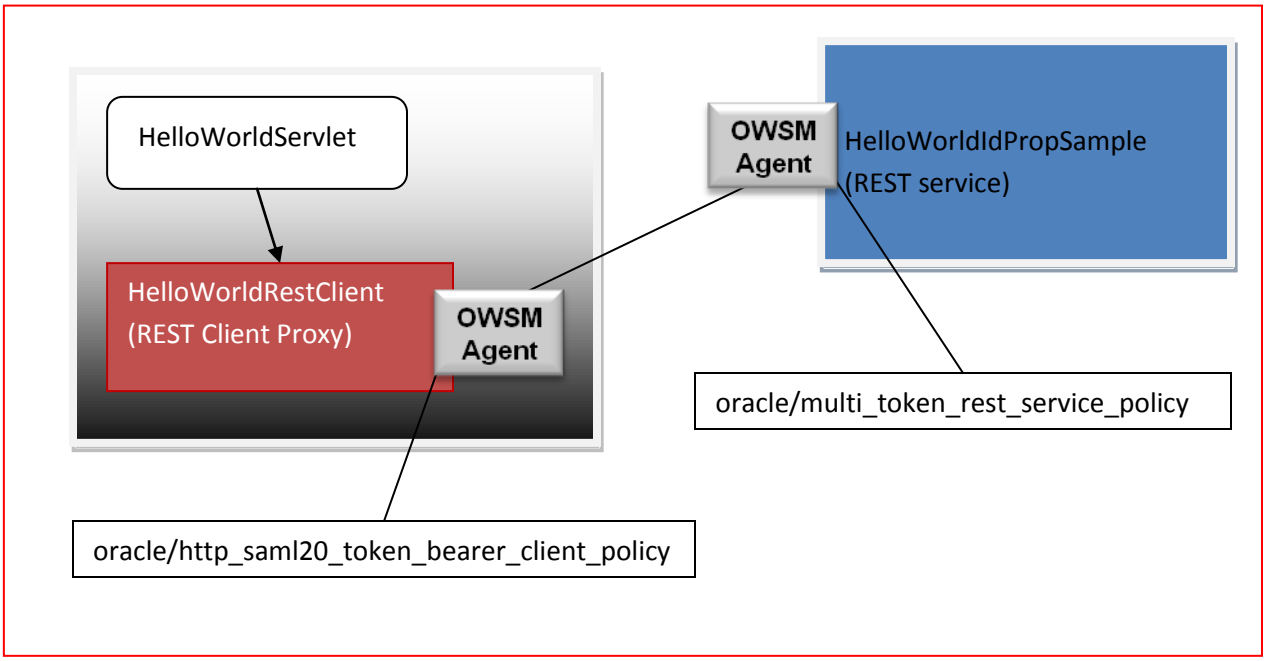
Go to the product console URL and provide username as weblogic and password as appropriate to your installation.

Product	URL	Note
Oracle WebLogic	http://localhost:7001/console	WebLogic Administration Console
Oracle Web Services Manager (OWSM)	http://localhost:7001/wsm-pm	Indicates status of OWSM Policy Manager. Presence of this page indicates that the Policy Manager has started
	http://localhost:7001/wsm-pm/validator	Show you all the out-of-the-box policy. If you see that page, OWSM policy store is properly deployed and running
Oracle Enterprise Manager (EM)	http://localhost:7001/em	Oracle Enterprise Manager

2 Usecase

Description

This How-To describes how to secure a JAX-RS REST service and client using OWSM 12.1.2.



Objective

The main objective of this How-To:

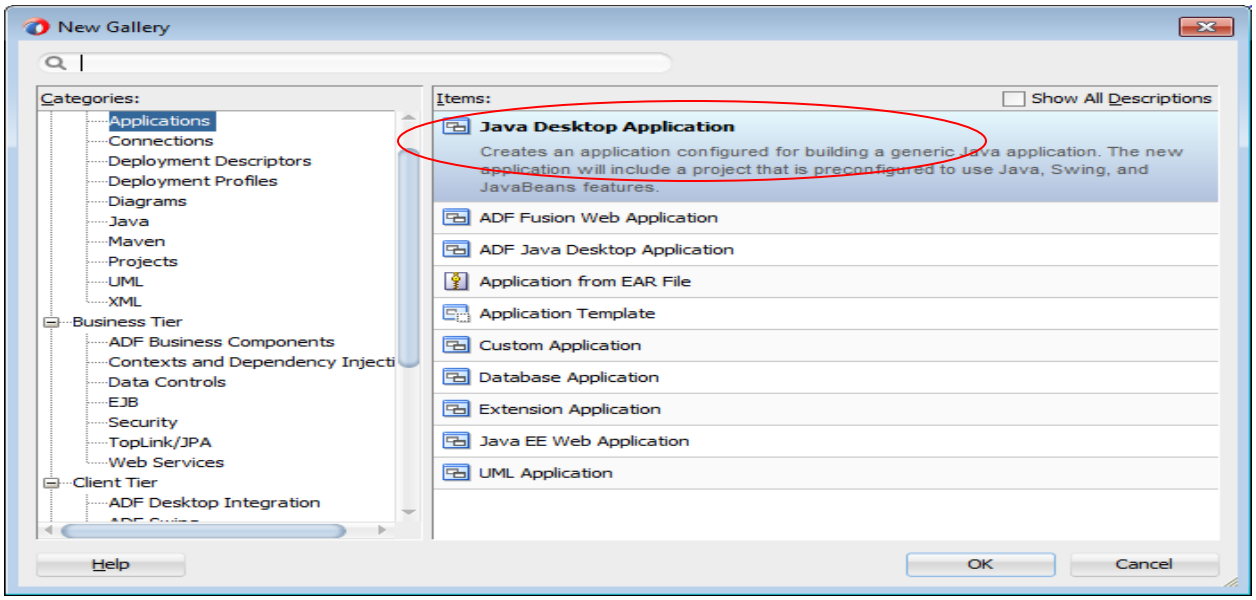
- How to build a simple REST services using JAX-RS technology in JDeveloper
- How to secure a simple HelloWorld JAX-RS application using OWSM Policy in JDeveloper
- Deploy and Run the HelloWorld JAX-RS application to a Weblogic domain
- How to build a simple HelloWorld REST Client Proxy for the HelloWorld REST service using JAX-RS technology in JDeveloper
- How to secure the HelloWorld REST Client Proxy using OWSM Policy in JDeveloper
- Testing end to end.

Policies Used

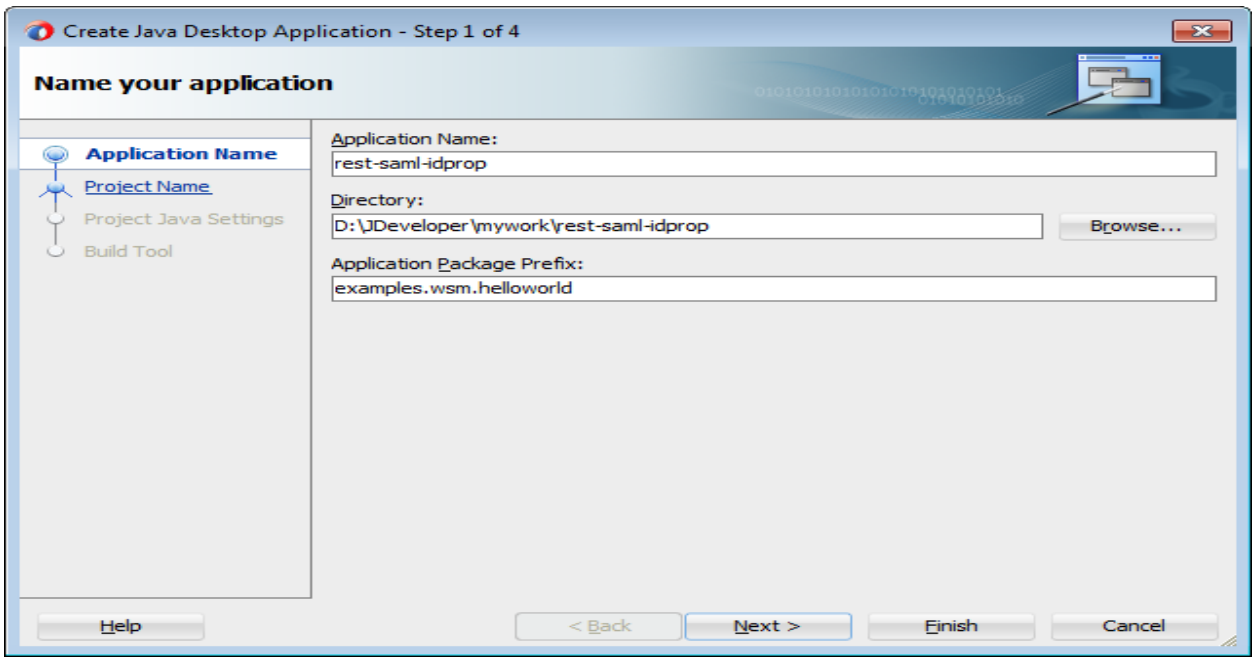
Service	Policy	Type
HelloWorldIdPropSample	oracle/multi_token_rest_service_policy	REST service
HelloWorldRestClient	oracle/http_saml20_token_bearer_client_policy	REST Client

3 Create "rest-saml-idprop" Application

This application will contain two projects. One for the REST service and another for the REST Client.



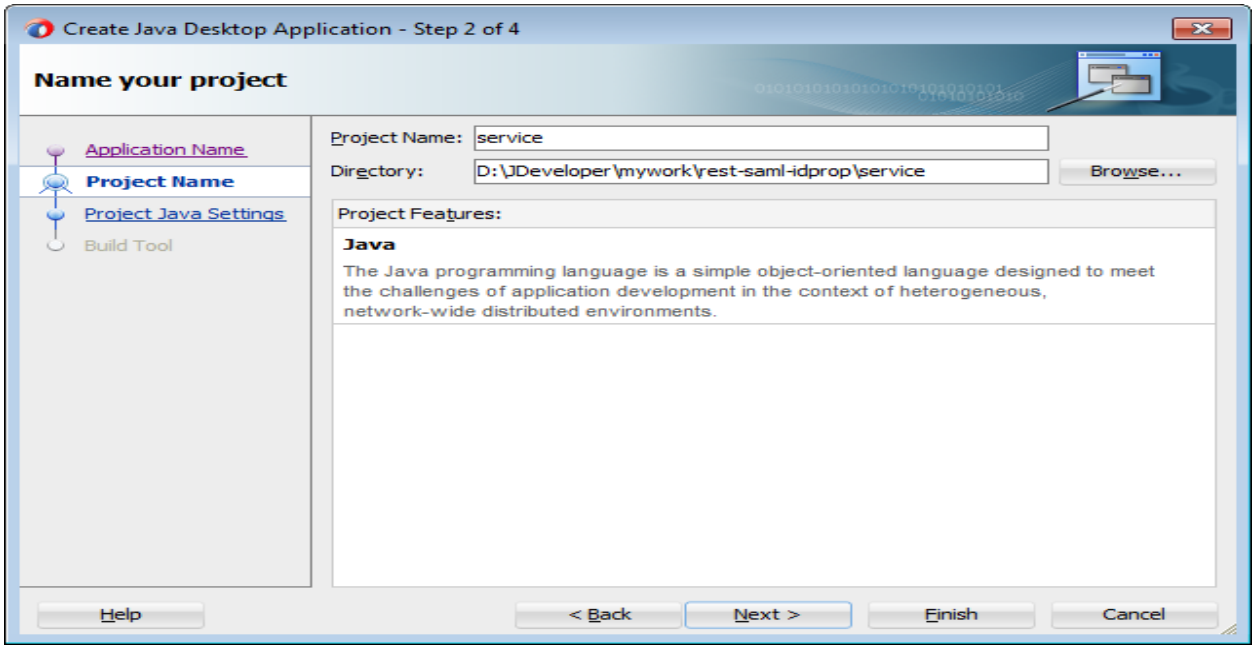
Provide the necessary information for creating the Application as shown in the screenshot below.



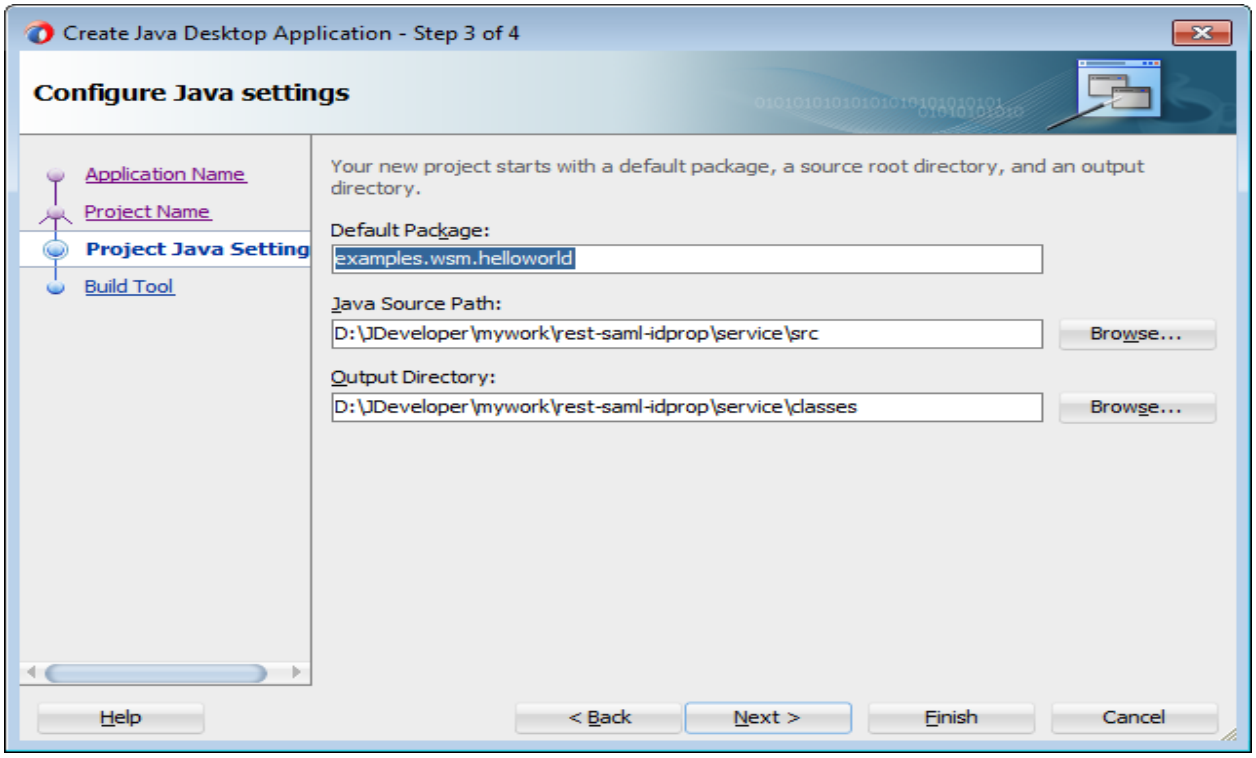
Click "Next"

3.1 Create "service" Project

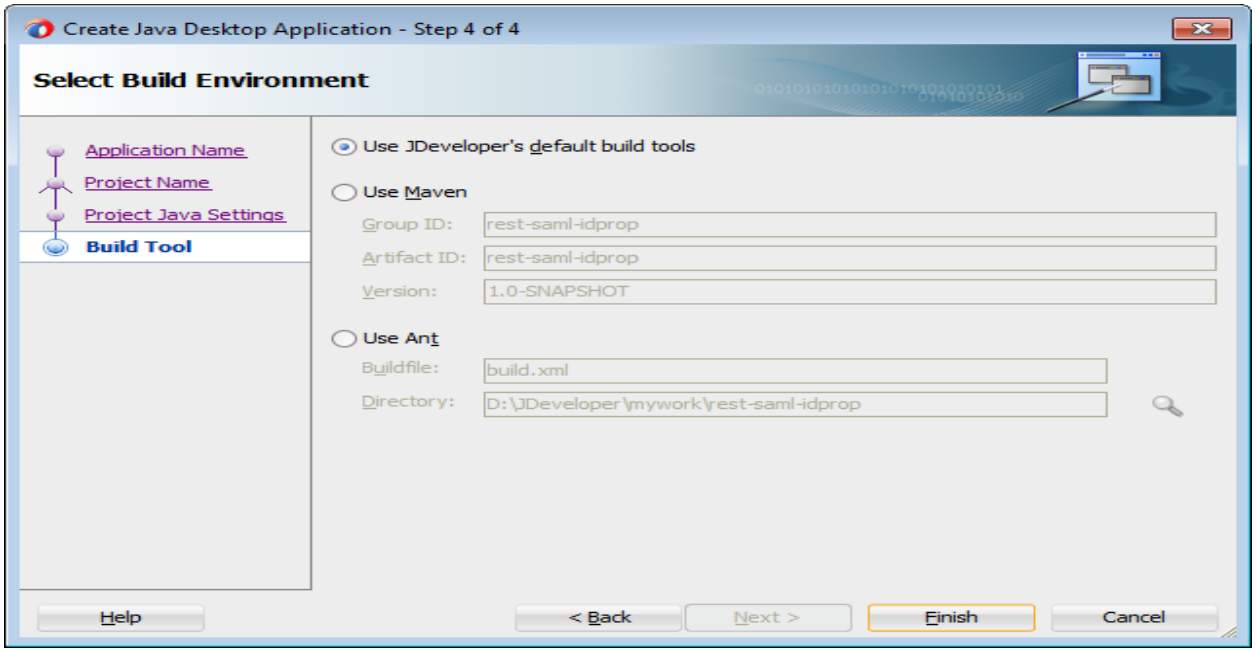
Create a project called "service" in JDeveloper as show in the screenshot below. This project will be used to build the REST service.



Click “Next”



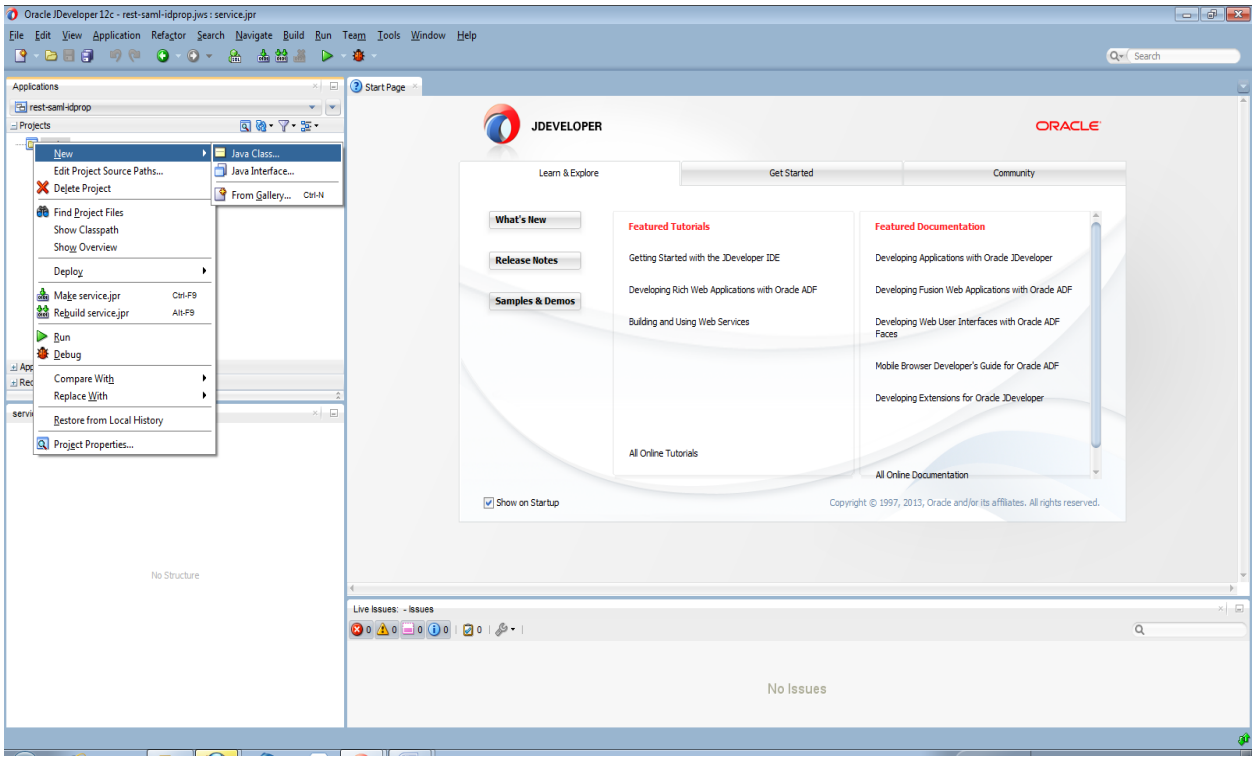
Click “Next”

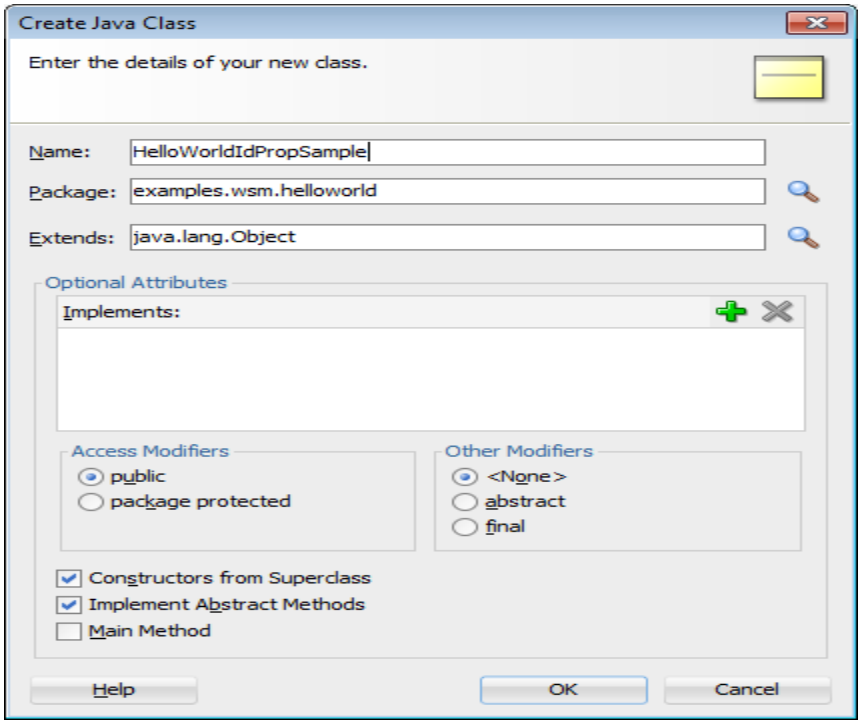


Click "Finish"

3.2 Create "HelloWorldIdPropSample" POJO class

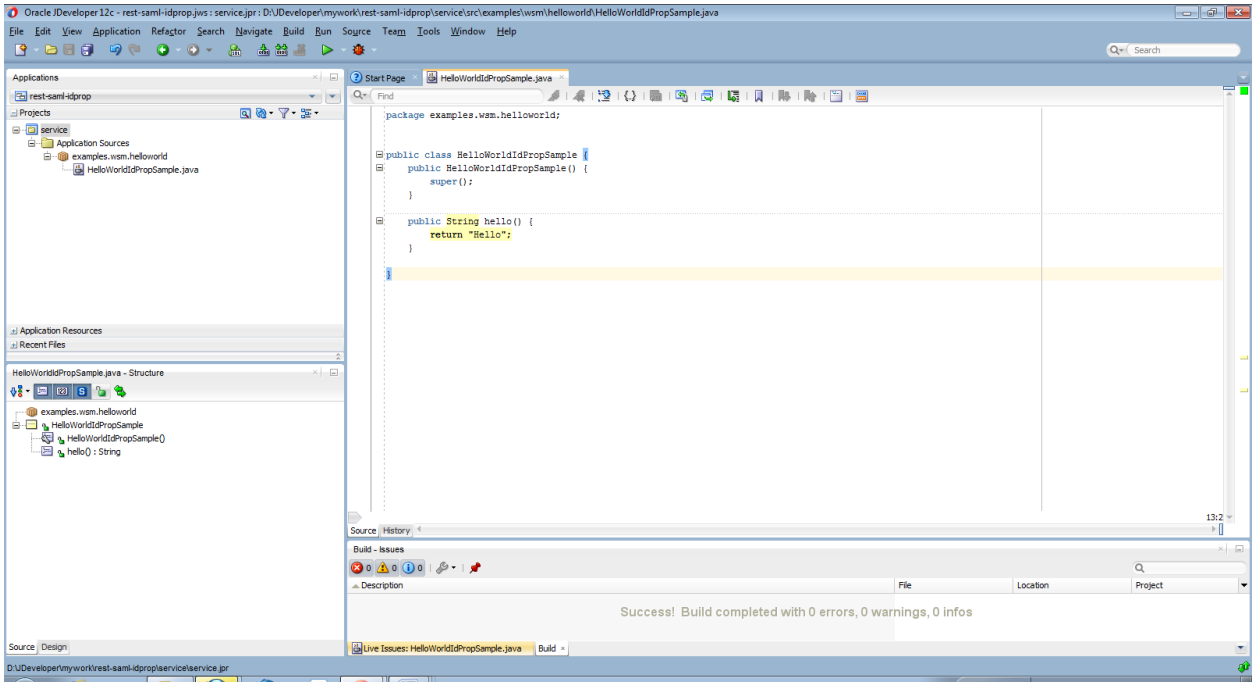
Create a POJO class under the "service" Project.





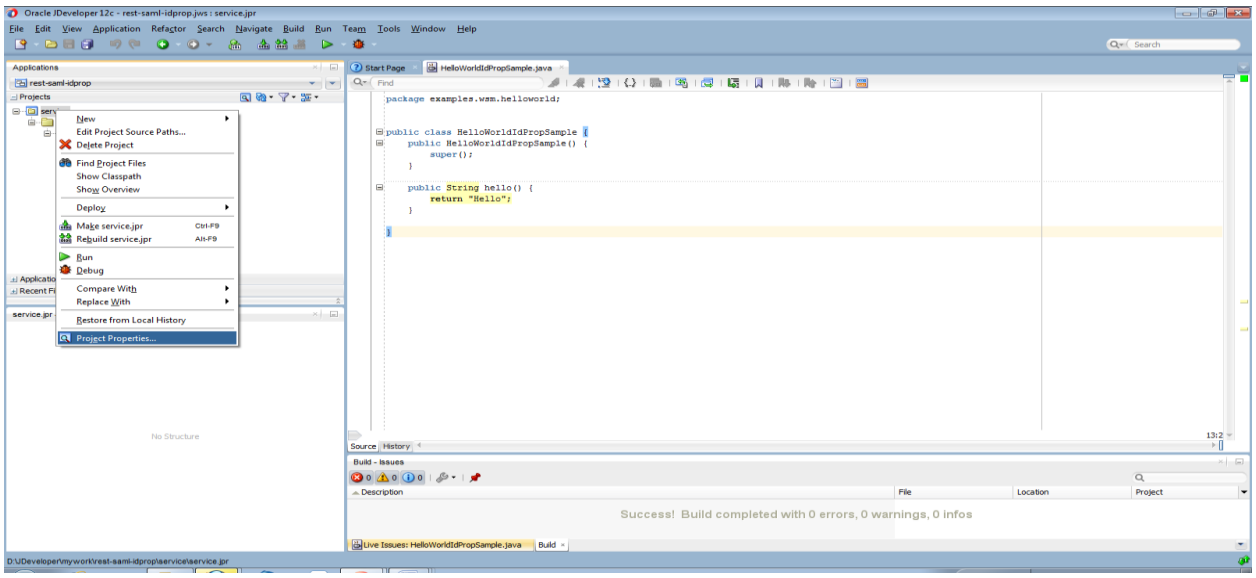
Click "OK"

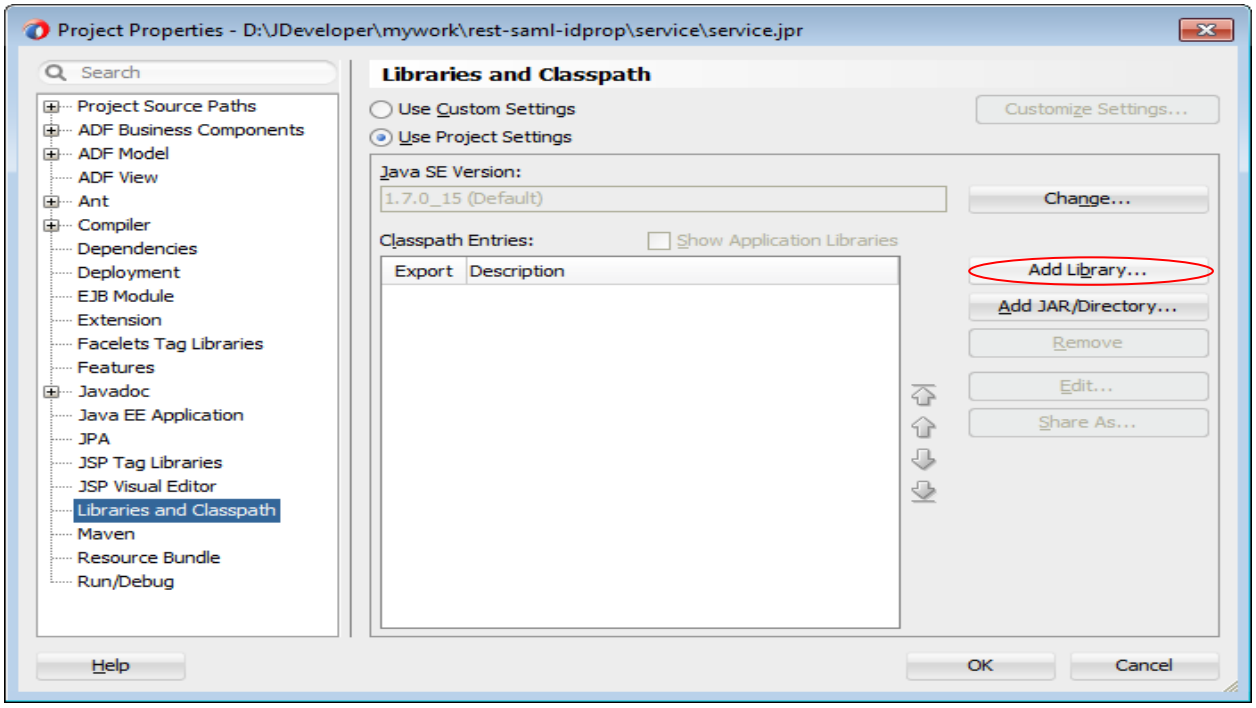
3.3 Add method "hello" to the POJO Class



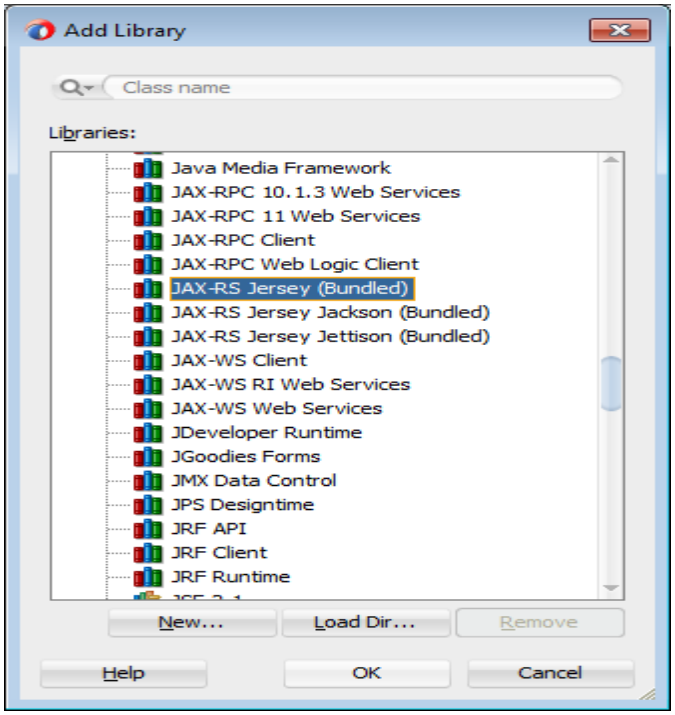
3.4 Add Jersey Libraries via Project Properties

We will use APIs that require Jersey Libraries in the Project. Add these Libraries via the Project Properties. This can be achieved by right-clicking on the "service" project and clicking on the "Project Properties" menu item as shown in the screenshot below.

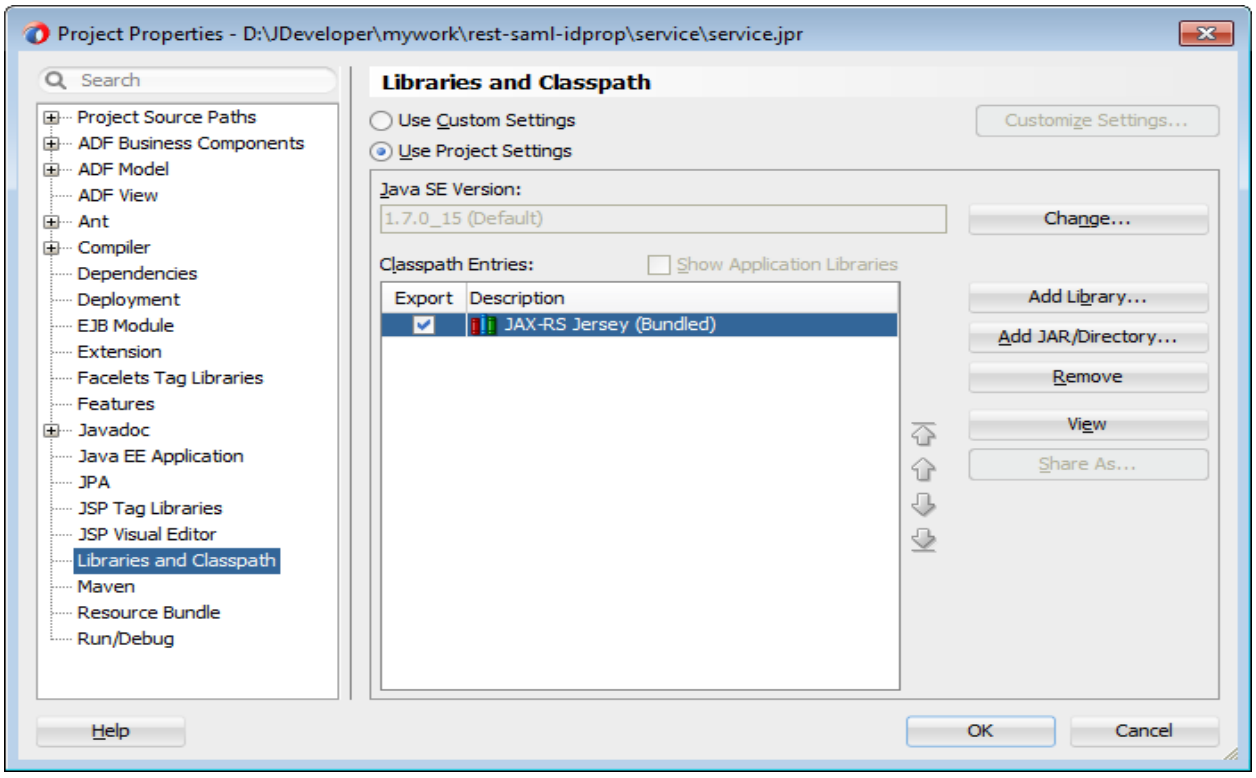




Click on "Add Library" button in the screenshot above. This will pop up the "Add Library" dialog below.



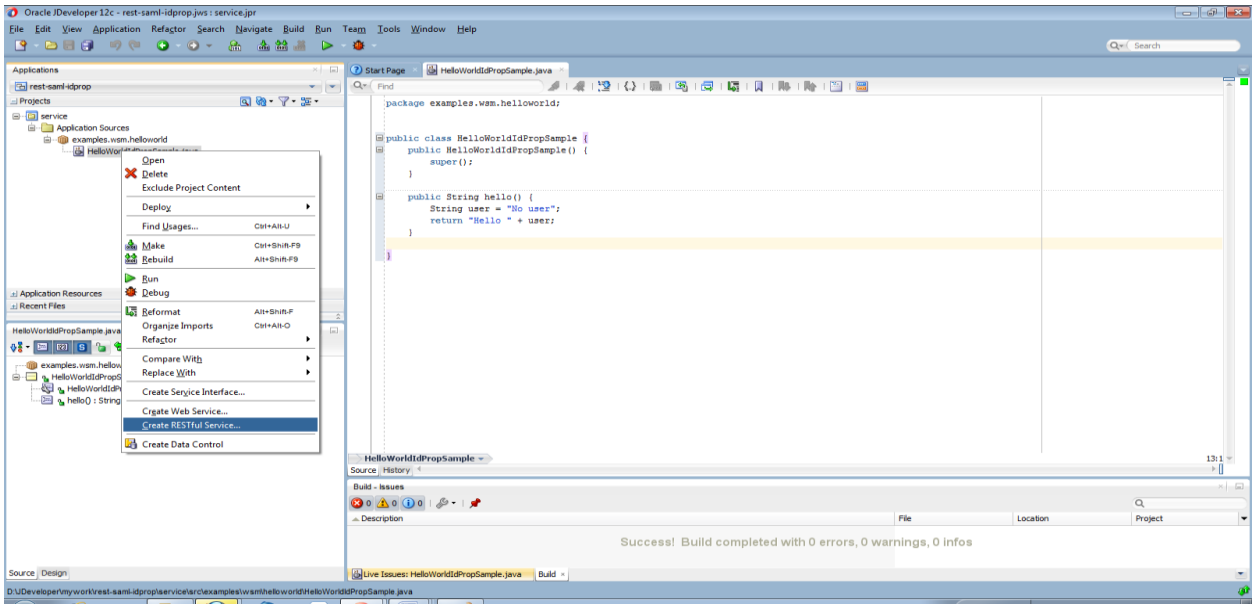
Click "OK"

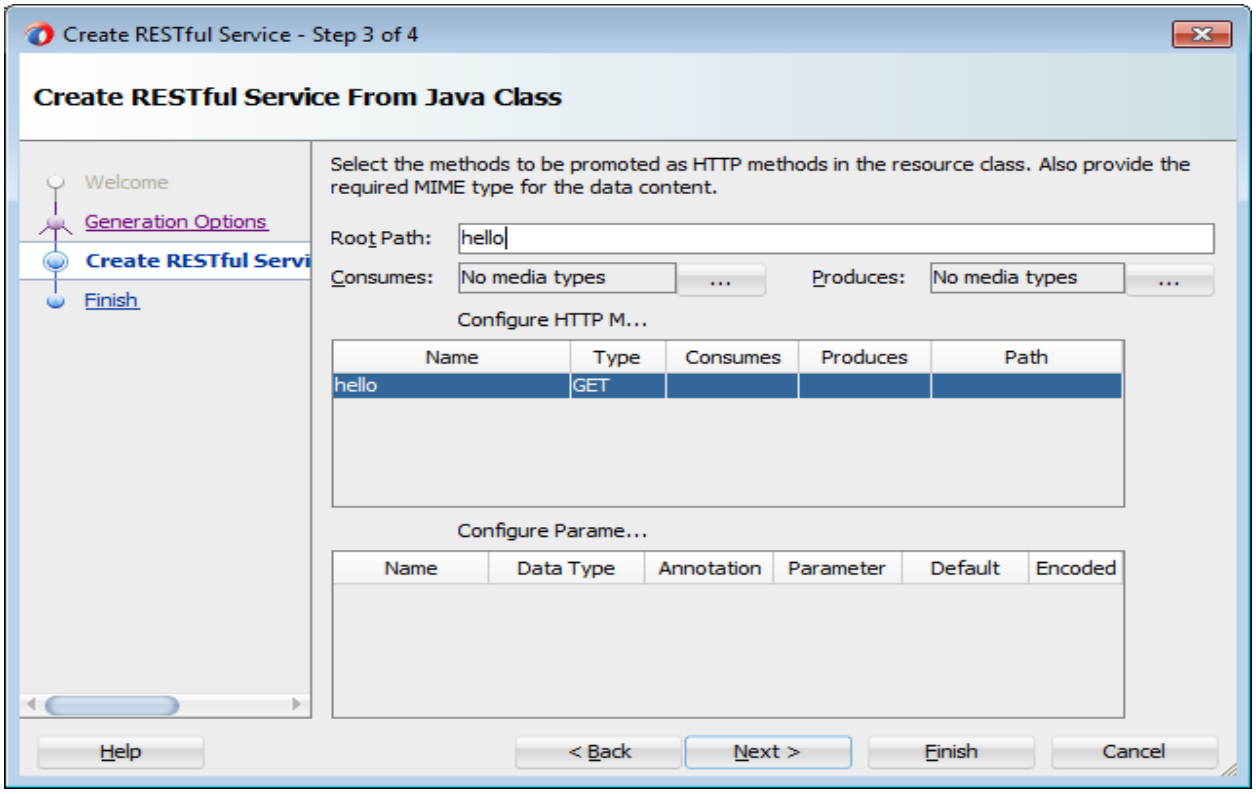


Click "OK"

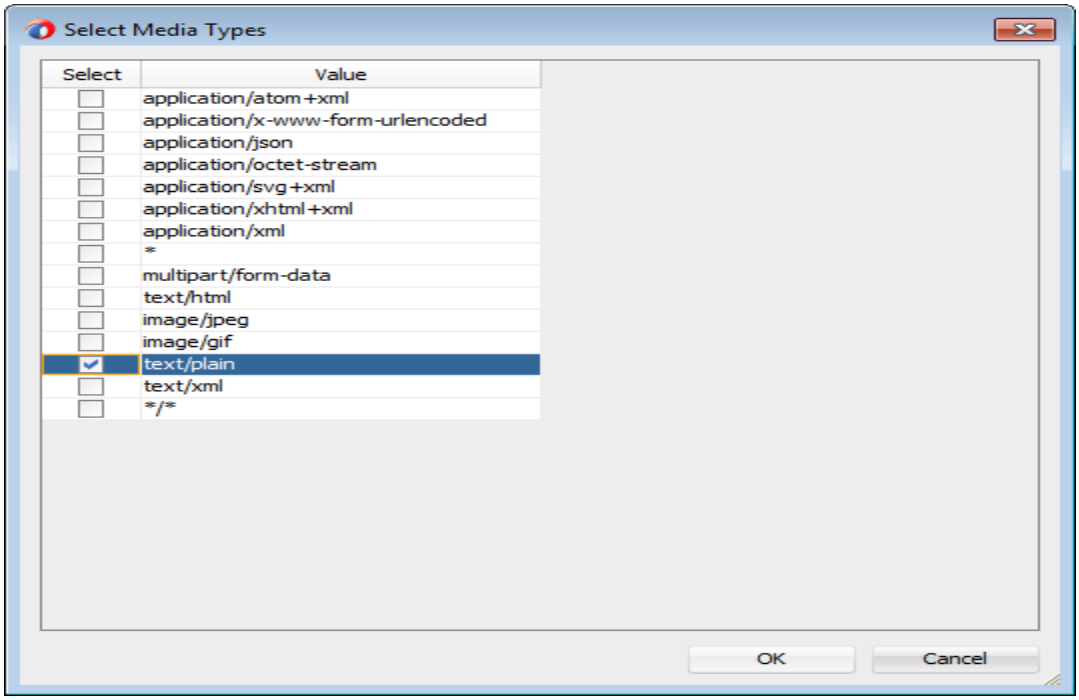
3.5 Create a RESTFul Service from the POJO Class

You can create a RESTFul Service by right clicking on the POJO Class and clicking on “Create RESTful Service” menu item as shown below.

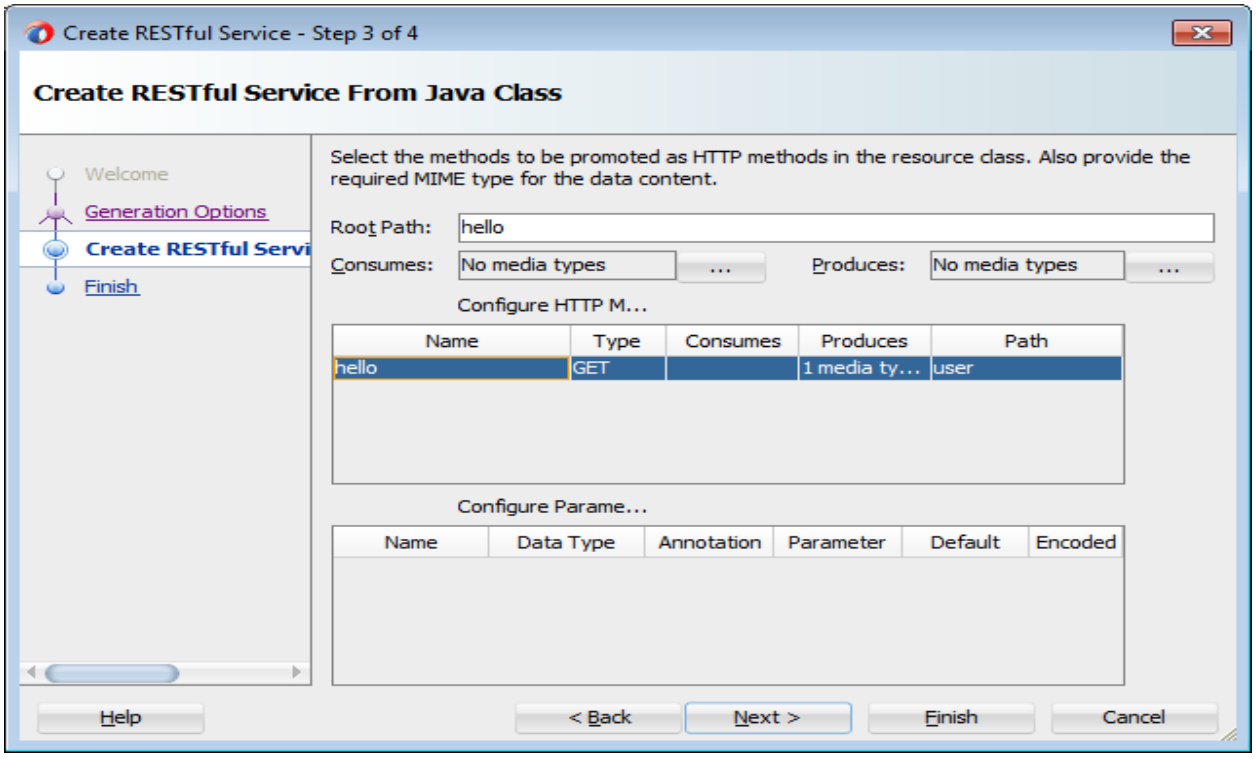




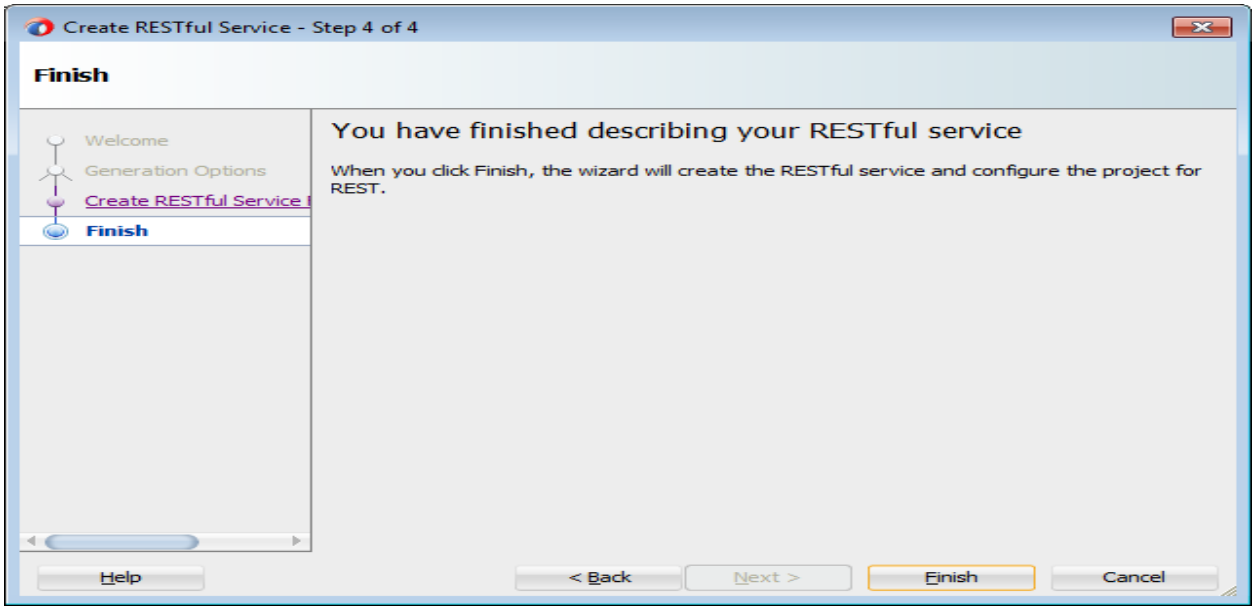
For the purposes of this How-To select the media type as text/plain by Clicking “Produces”.



Click "OK".



Click "Next"



Click “Finish”

3.6 Using @Context and SecurityContext

Use the @Context annotation and the SecurityContext object to get the authenticated user. The code to use the @Context annotation and the SecurityContext object is provided below.

```
package examples.wsm.helloworld;

import java.security.Principal;

//…skipping few imports

import javax.ws.rs.core.Context;  import javax.ws.rs.core.SecurityContext;

@Path("hello")

public class HelloWorldIdPropSample {

   public HelloWorldIdPropSample() {

      super();

   }

   @GET

   @Produces("text/plain")

   @Path("user")

   public String hello(@Context SecurityContext sc) {

      String user = "No user";

      if ( null != sc ) {

         Principal p = sc.getUserPrincipal();

         if ( null != p) {

            user = p.getName();

         }

      }

      return "Hello " + user;

   }

}
```

NOTE: Modify the code to use the @Context annotation to get the authenticated user after creating a RESTful Service.

Pay special attention to the import statements.

import java.security.Principal;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.SecurityContext;

Also note that that the method "hello" has a Path annotation.

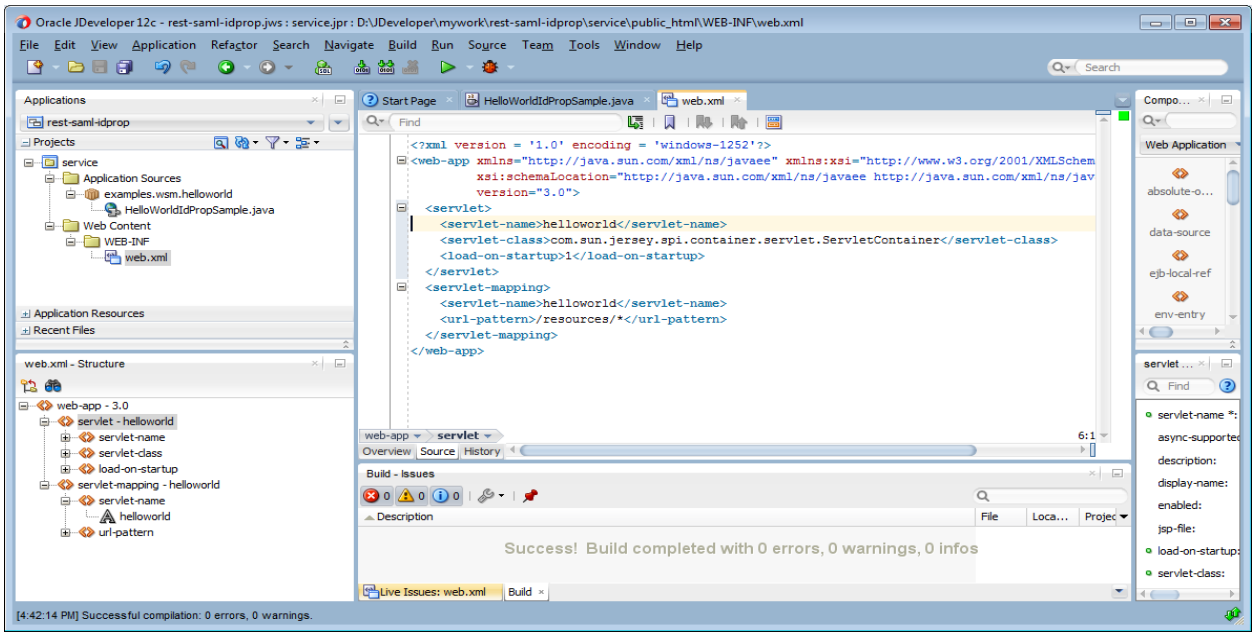
WARNING: If you add the @Context parameter before creating the RESTful service – then you will not be able to use the "Create RESTful Service" menu item as shown above to create a RESTful service. So the sequence described above is important.

While you can add the @Path annotations, etc manually the "Create RESTful Service" converts the project to a Web Project and adds the web.xml deployment descriptor – which allows you to secure the RESTful Service.

NOTE: There are other ways of programming this sample in JDeveloper. The sequence shown here is important only for this particular scenario.

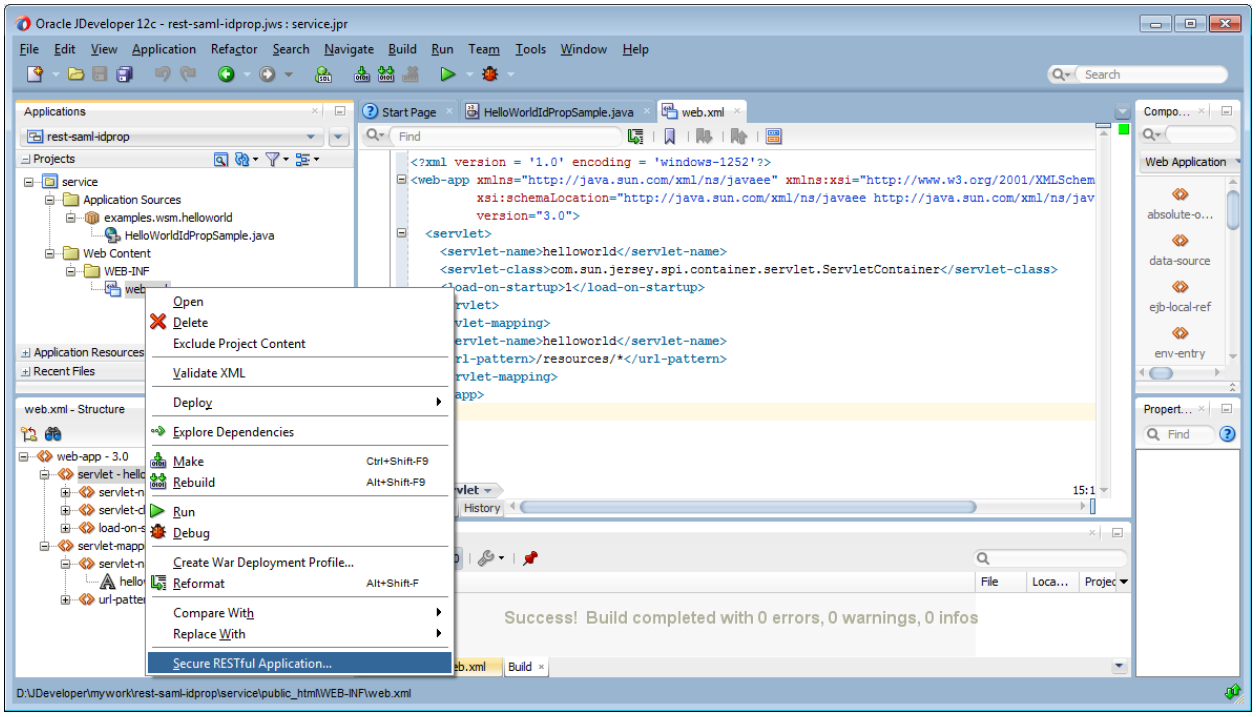
3.7 Modify the Servlet name

When you create a RESTful service – Jdeveloper automatically modified the project to be a Web Project and adds web.xml. However the default servlet name is called "jersey". Modify the servlet name and provide a more user friendly name as shown below.

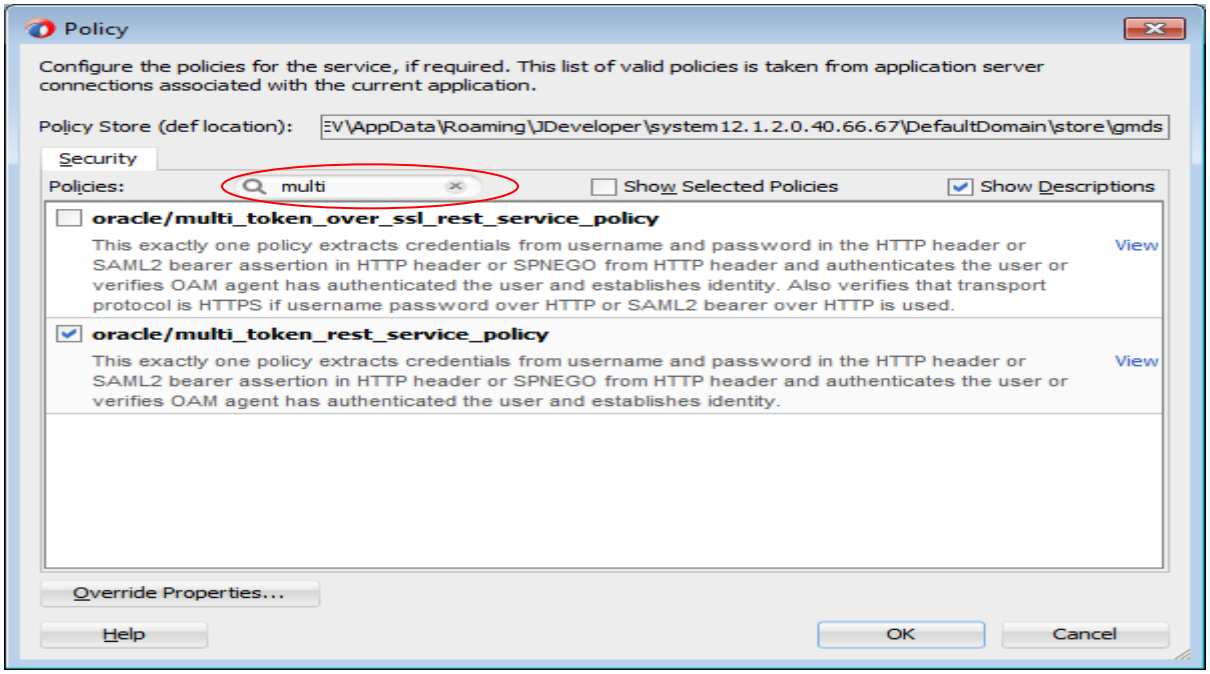


3.8 Secure the REST service in JDeveloper

You can secure a REST service by right-clicking on "web.xml" and selecting the "Secure RESTful Application" menu item as shown below.

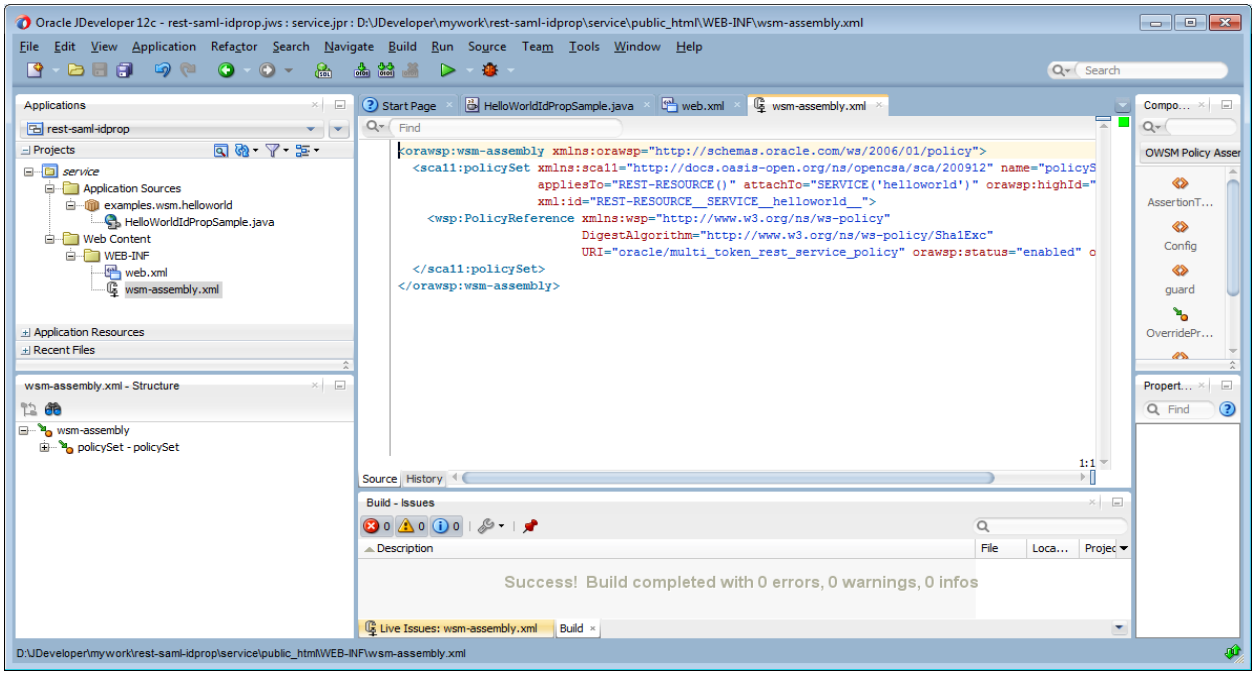


You can search for a policy of a particular name in JDeveloper as shown below.

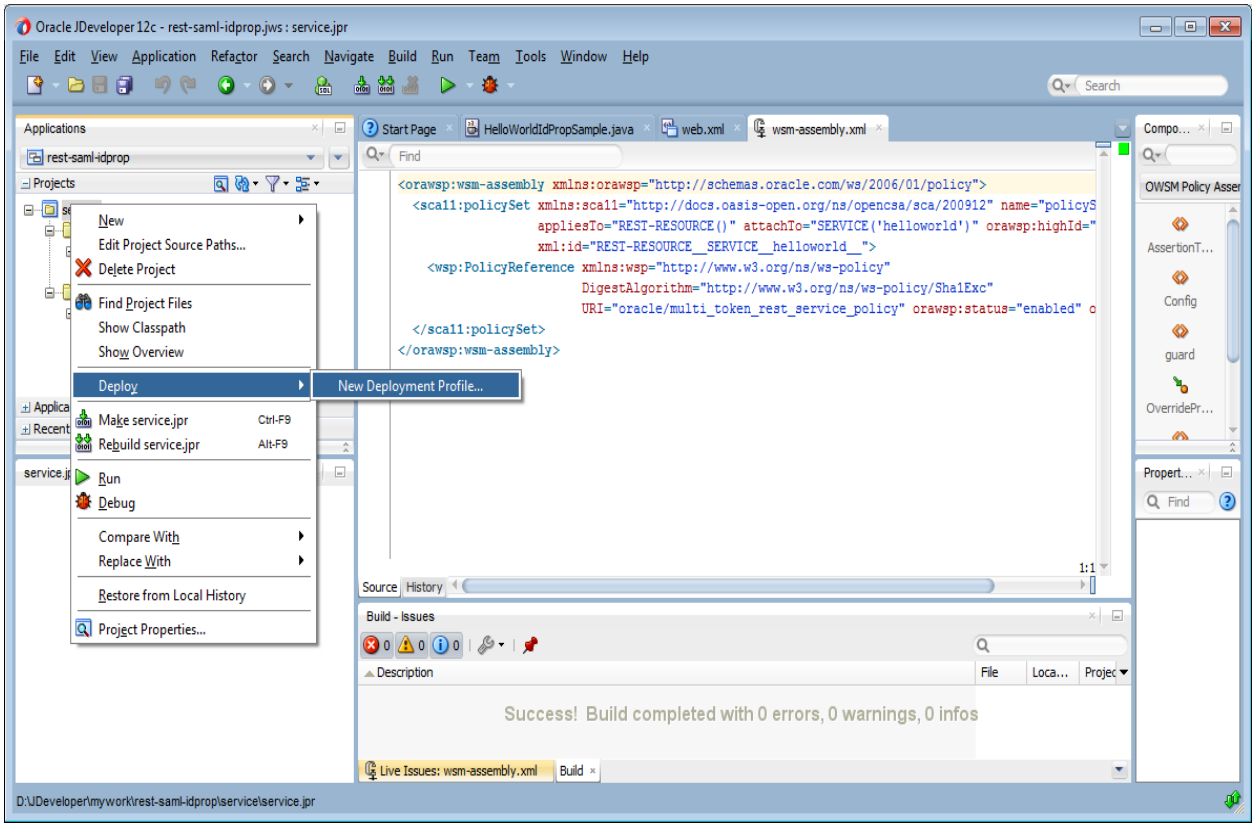


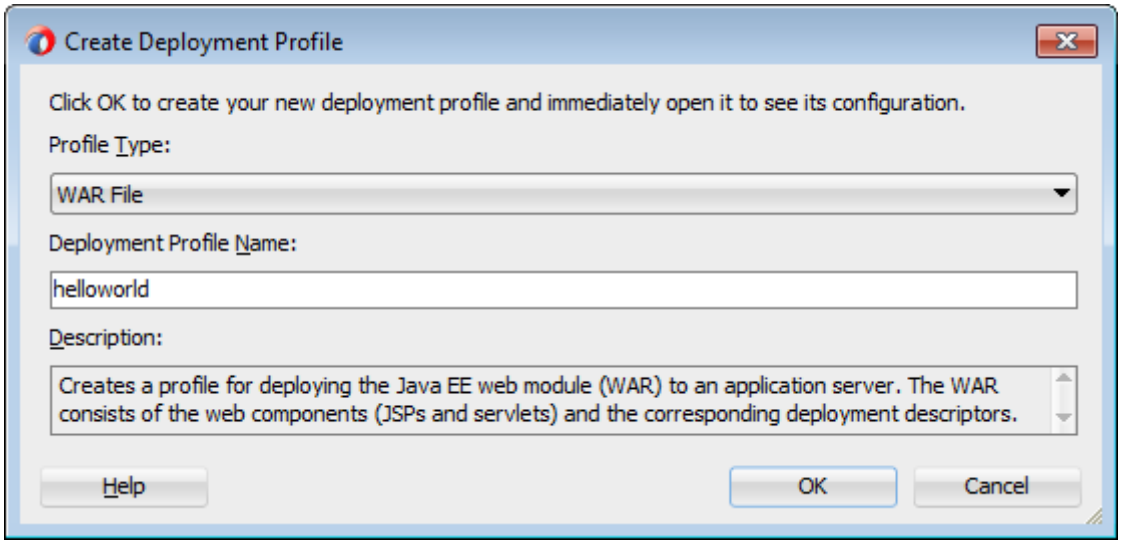
Click “OK”

Securing a RESTful Service will create a wsm-assembly.xml document as shown below.

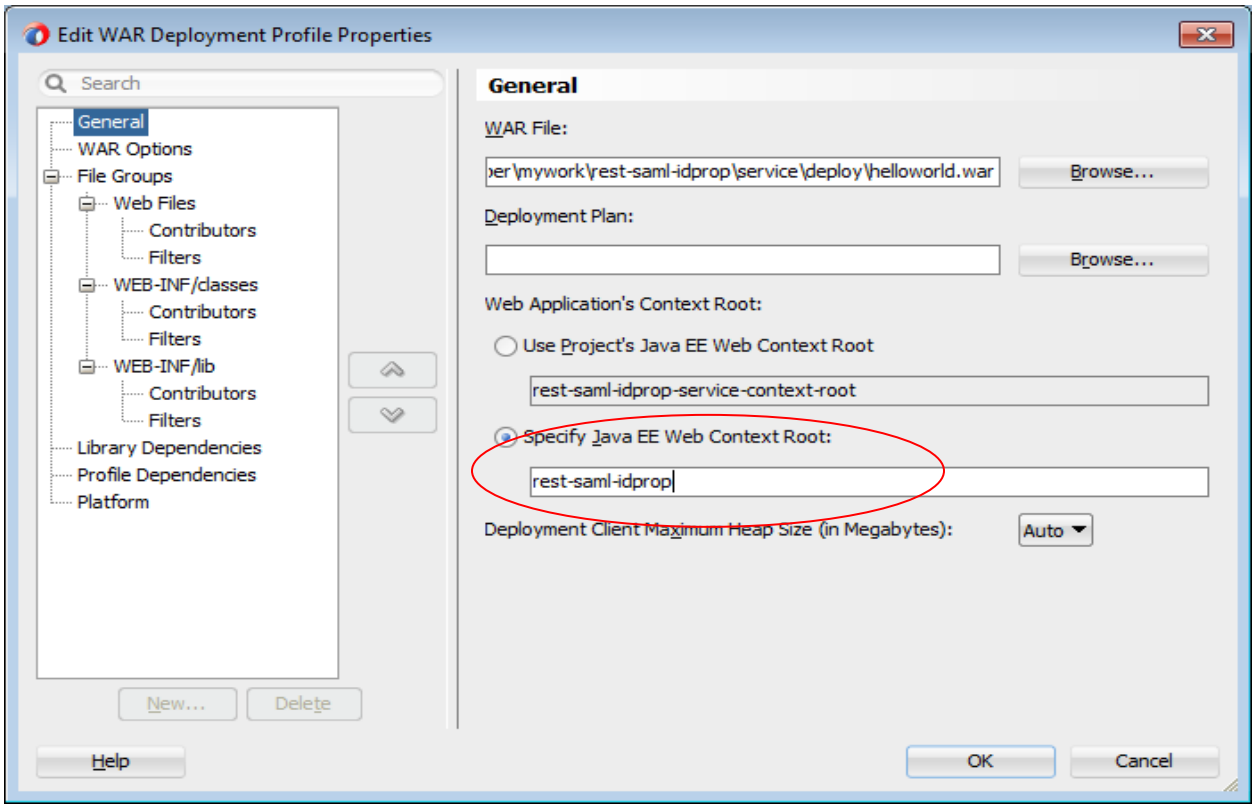


3.9 Create a Deployment Profile and WAR

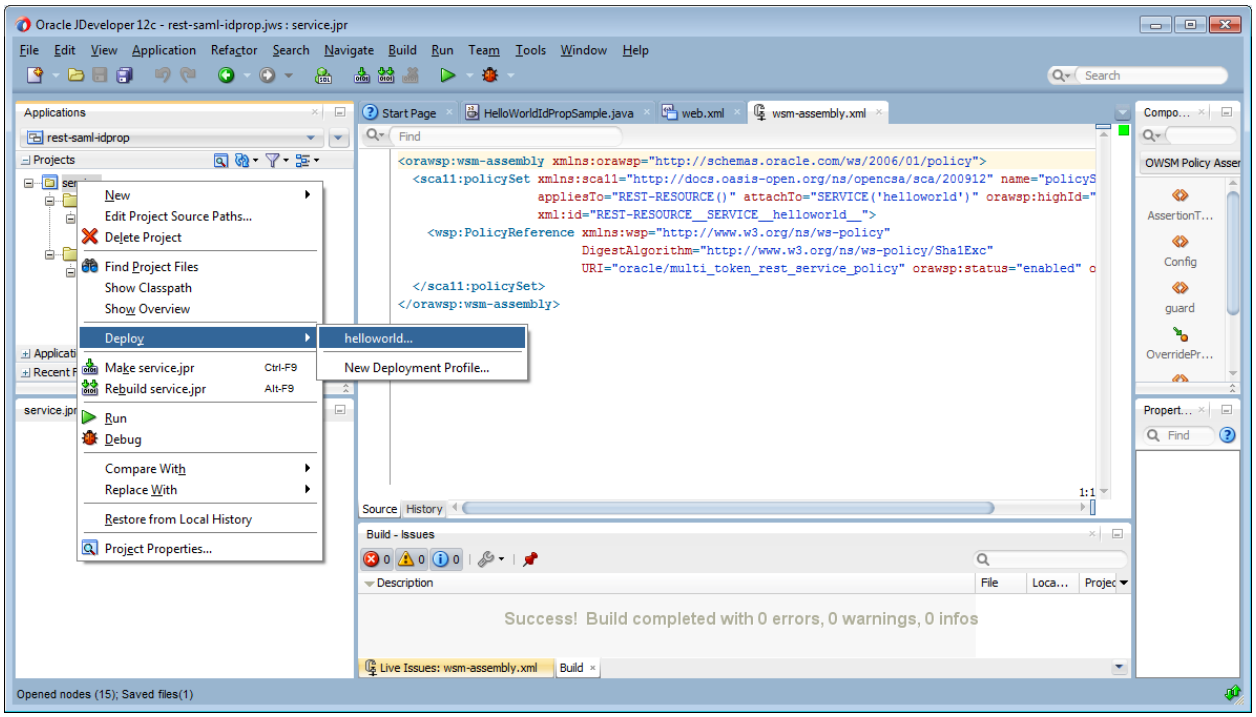


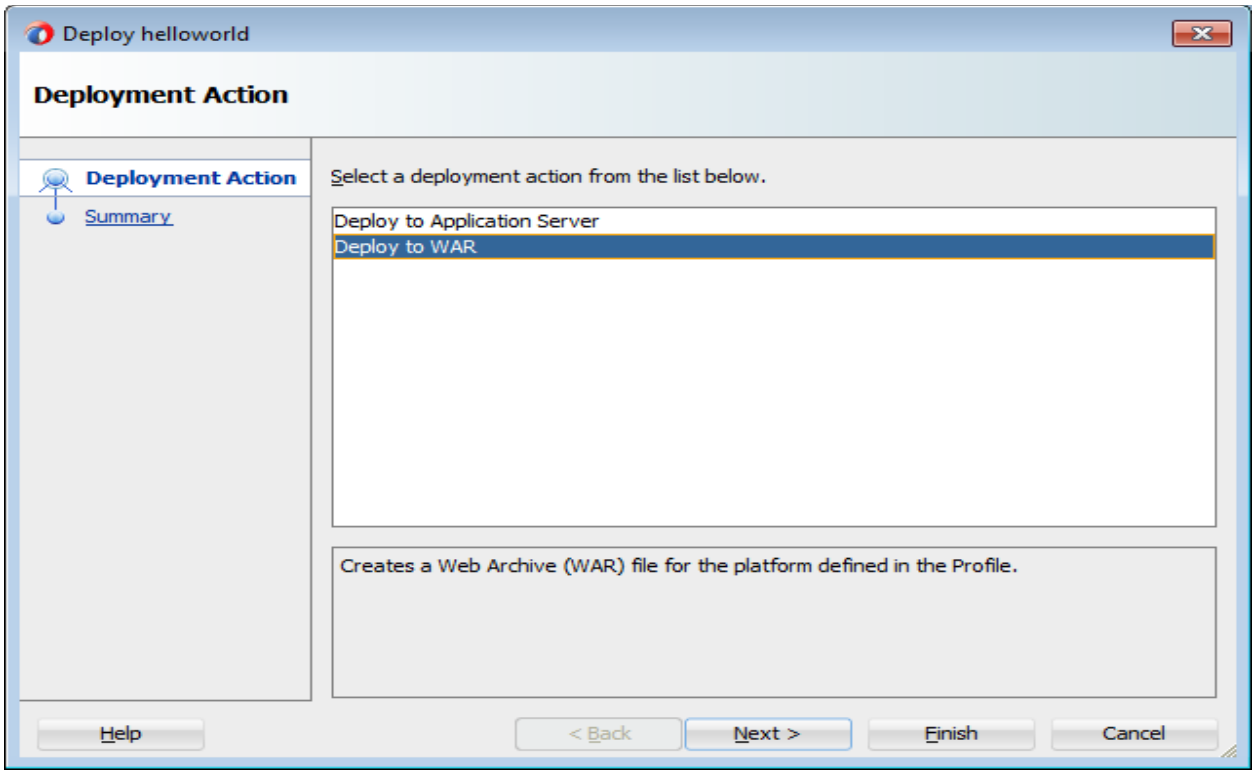


Click "OK"

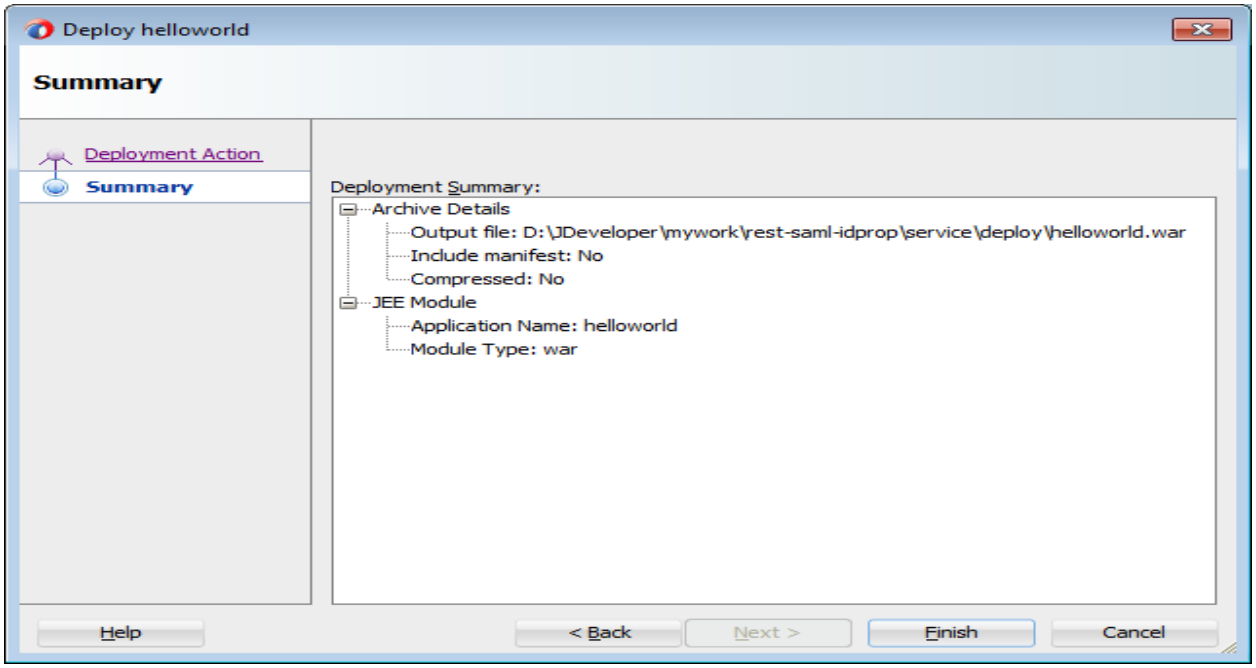


Click "OK"


Oracle JDeveloper 12c - rest-saml-idprop.jws : service.jpr
File Edit View Application Refactor Search Navigate Build Run Team Tools Window Help
Applications
rest-saml-idprop
Projects
New
Edit Project Source Paths...
Delete Project
Find Project Files
Show Classpath
Show Overview
Deploy
Make service.jpr Ctrl-F9
Rebuild service.jpr Alt-F9
Run
Debug
Compare With
Replace With
Restore from Local History
Project Properties...
helloworld...
New Deployment Profile...
Start Page
HelloWorldIdPropSample.java
web.xml
wsm-assembly.xml
<orawsp:wsm-assembly xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy">
<scal1:policySet xmlns:scal1="http://docs.oasis-open.org/ns/opencsa/sca/200912" name="policyS
appliesTo="REST-RESOURCE()" attachTo="SERVICE('helloworld')" orawsp:highId="
xml:id="REST-RESOURCE__SERVICE__helloworld__">
<wsp:PolicyReference xmlns:wsp="http://www.w3.org/ns/ws-policy"
DigestAlgorithm="http://www.w3.org/ns/ws-policy/Sha1Exc"
URI="oracle/multi_token_rest_service_policy" orawsp:status="enabled" o
</scal1:policySet>
</orawsp:wsm-assembly>
Source History
Build - Issues
Description File Loca... Projec
Success! Build completed with 0 errors, 0 warnings, 0 infos
Live Issues: wsm-assembly.xml Build
Opened nodes (15); Saved files(1)

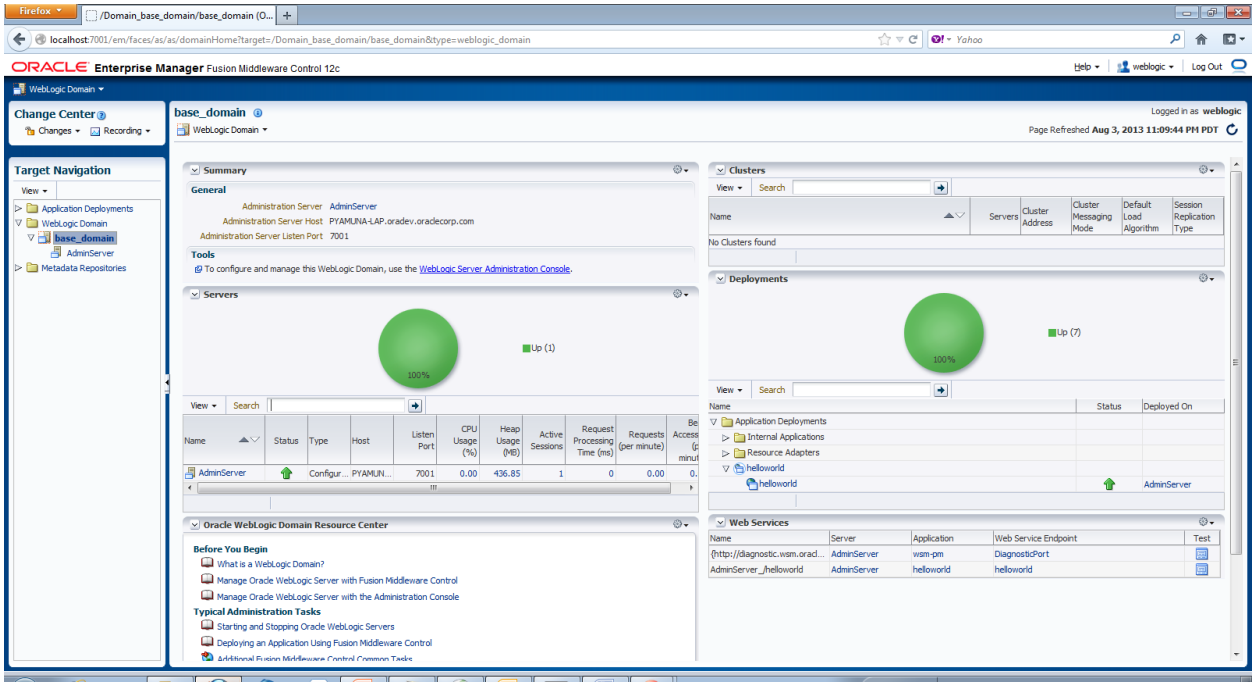

Deploy helloworld
Deployment Action
Deployment Action
Summary
Select a deployment action from the list below.
Deploy to Application Server
Deploy to WAR
Creates a Web Archive (WAR) file for the platform defined in the Profile.
Help
< Back
Next >
Finish
Cancel

Click “Next”

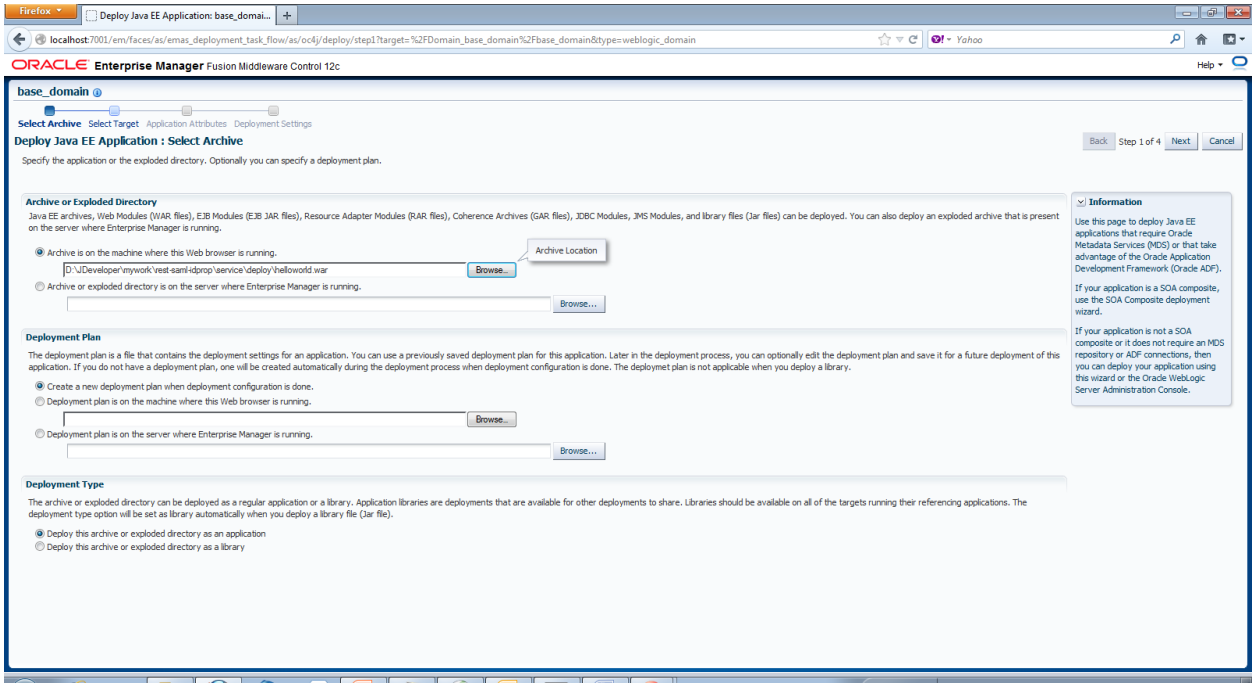


Click “Finish”

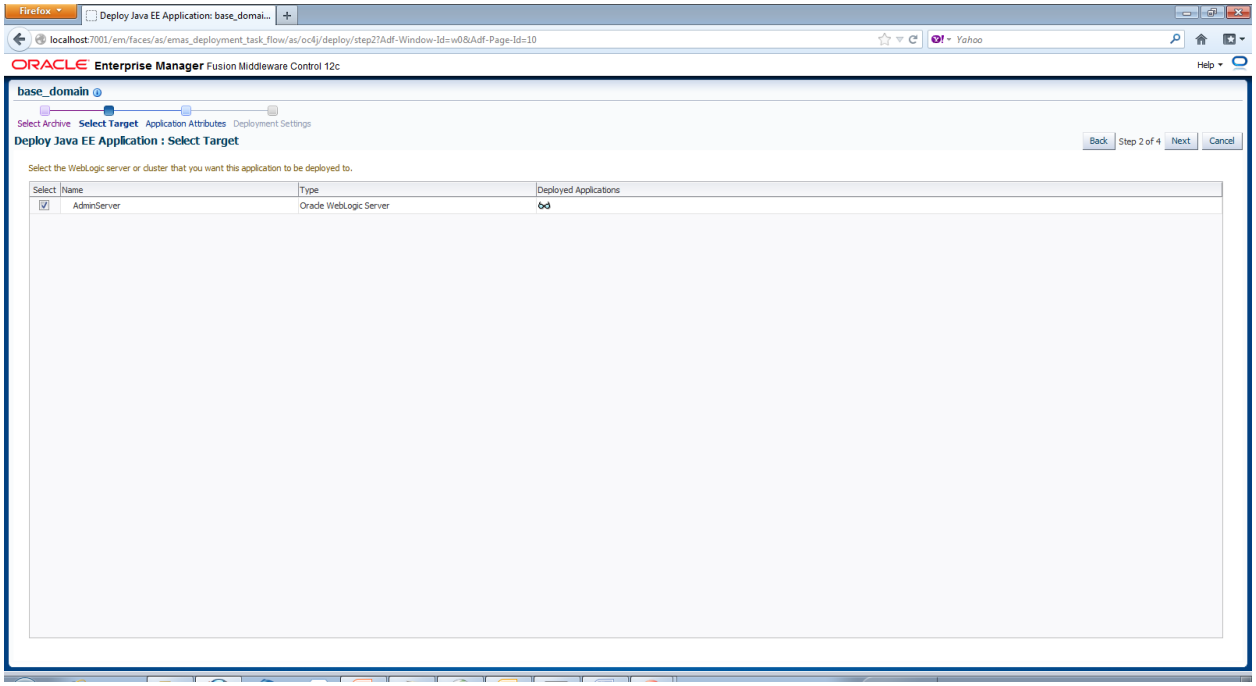
3.10 Deploy the REST service “helloworld.war” to WLS using EM



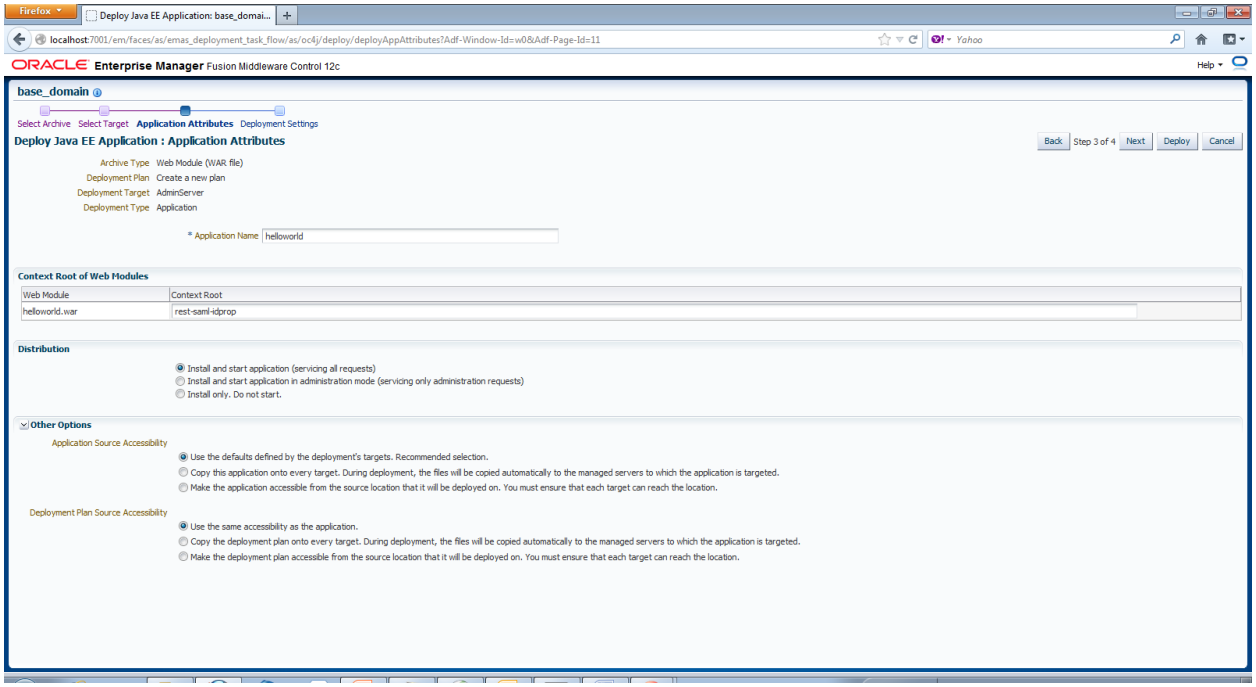
Click “Webogic Domain->Deployments” from the RHS menu in the screenshot above.



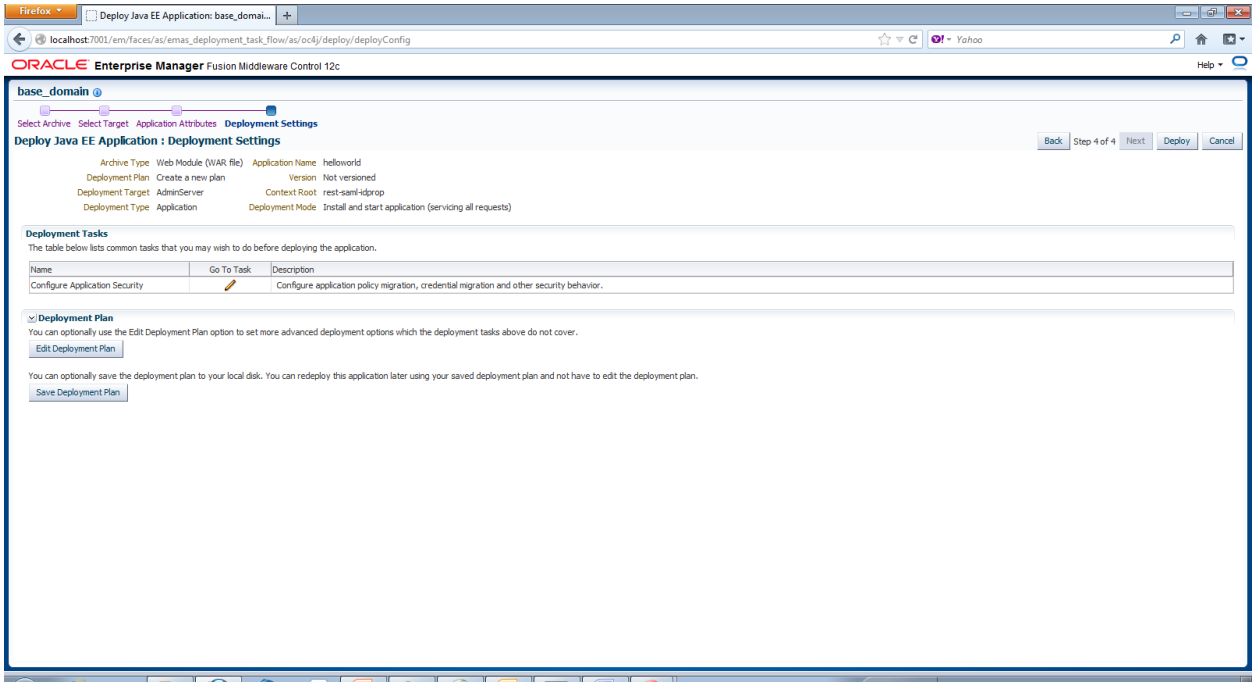
Click “Next”



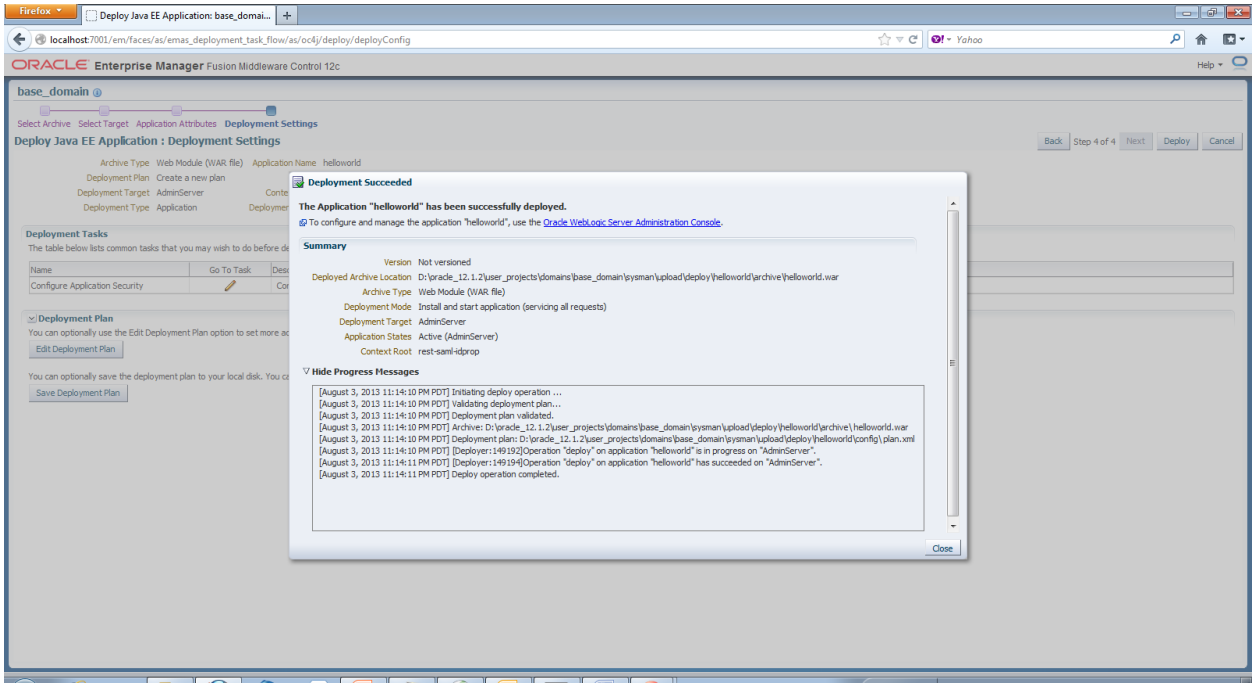
Click “Next”



Click “Next”



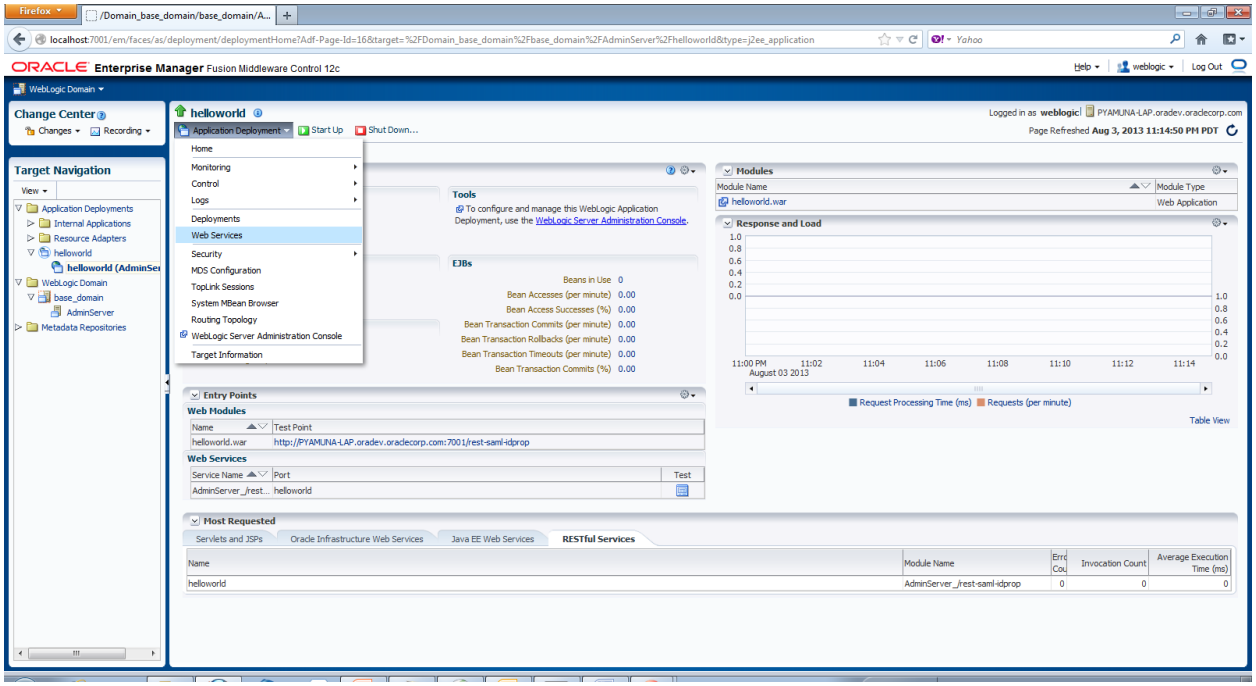
Click “Deploy”

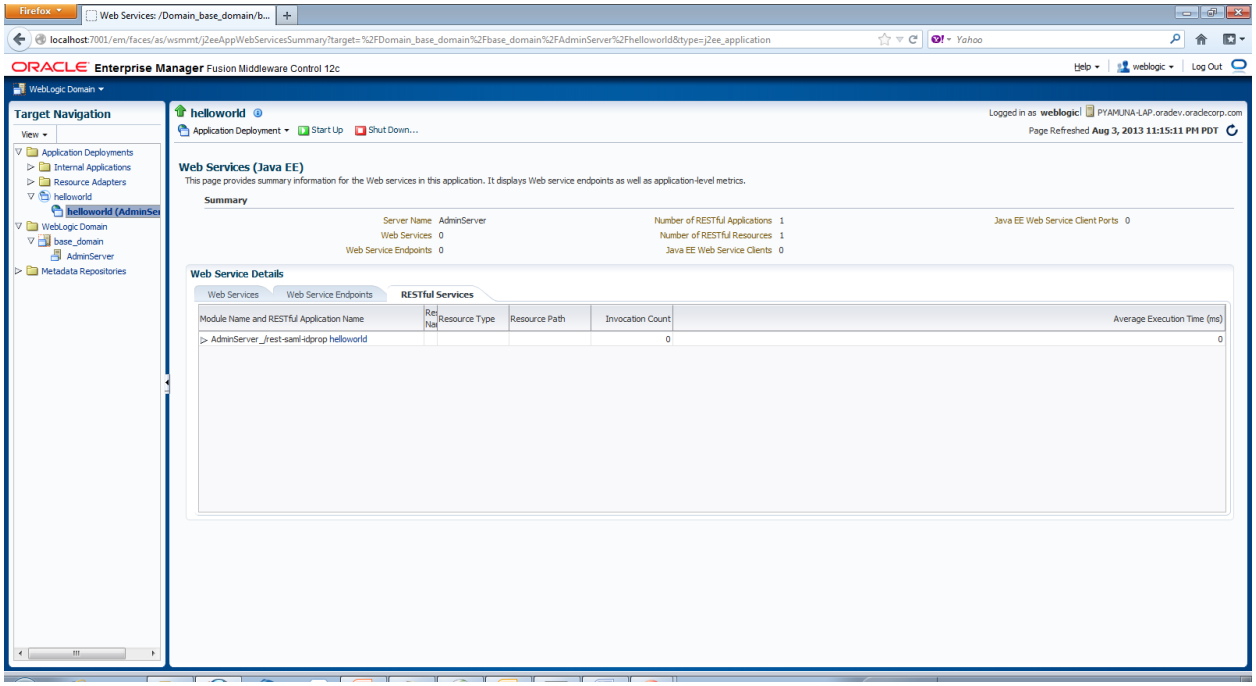


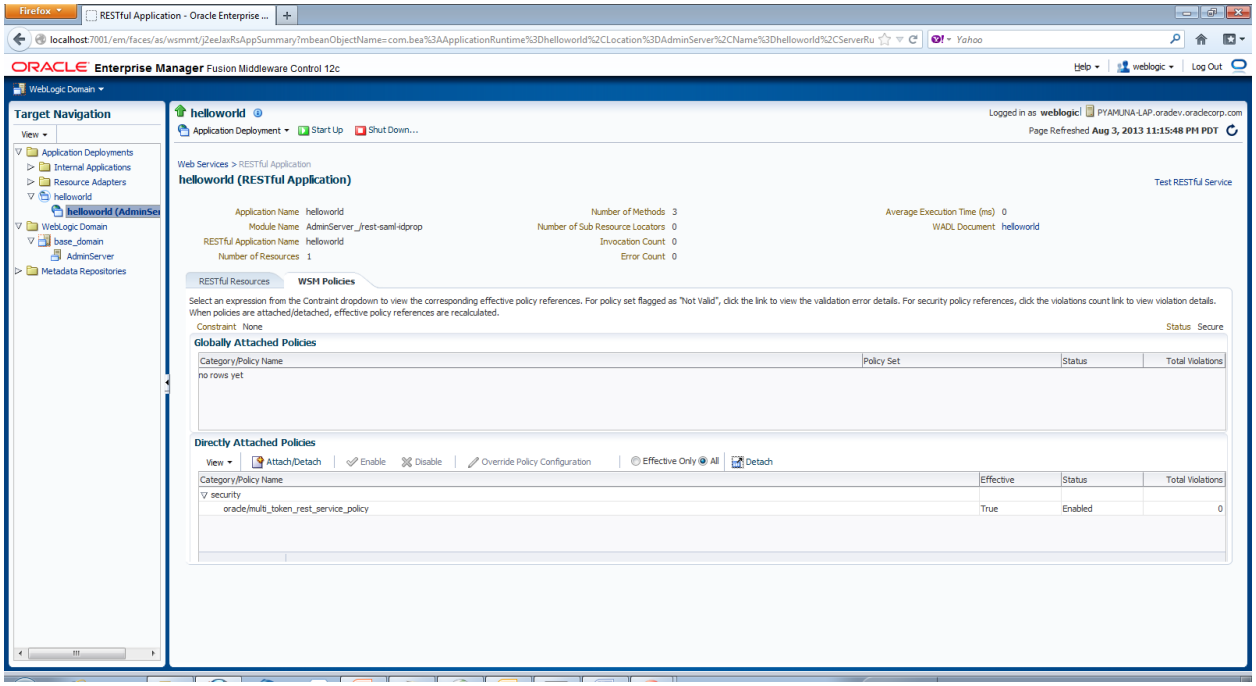
Click “Close”

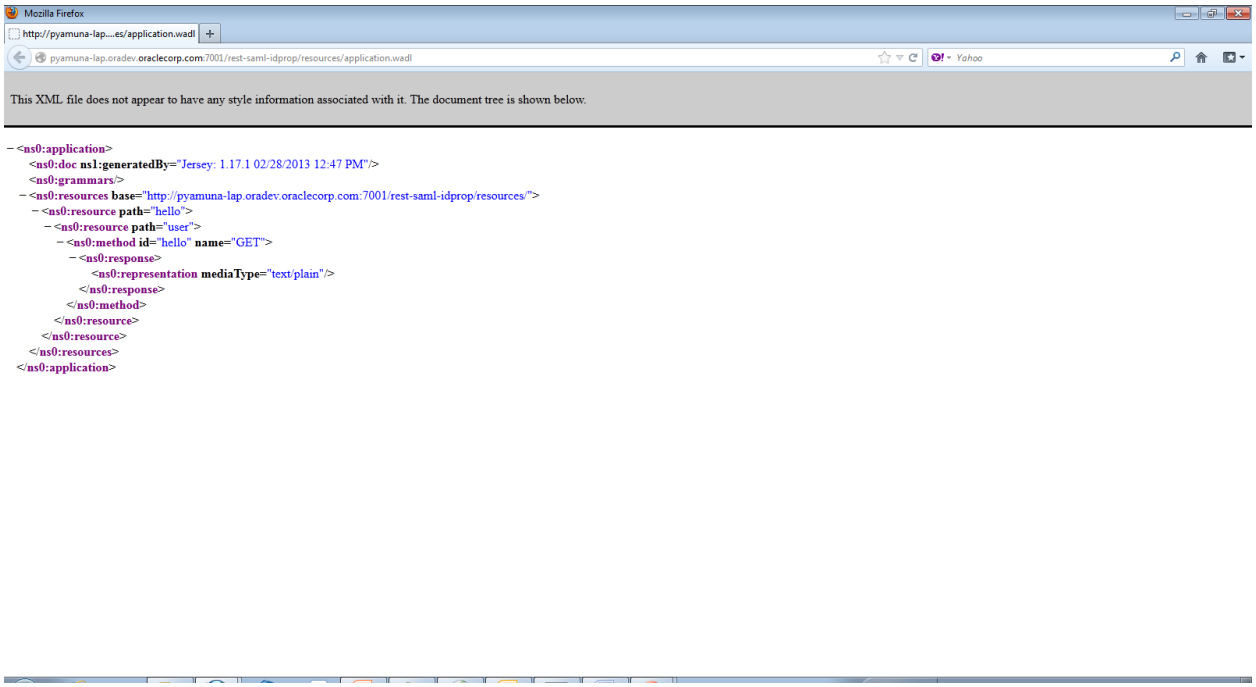
3.11 Validate the REST service

Expand the “helloworld” node in the LHS and click on the helloworld (AdminServer) instance as shown in the screenshot below. This will open up the Application Home page in EM. Select “Application Deployment->Web Services” menu item on the RHS as shown in the below screenshot.




Web Services: /Domain_base_domain/b...
localhost:7001/em/faces/as/wsmmt/j2eeAppWebServicesSummary?target=%2FDomain_base_domain%2Fbase_domain%2FAdminServer%2Fhelloworld&type=j2ee_application
ORACLE Enterprise Manager Fusion Middleware Control 12c
Target Navigation
helloworld
Application Deployment
Start Up
Shut Down...
Web Services (Java EE)
This page provides summary information for the Web services in this application. It displays Web service endpoints as well as application-level metrics.
Summary
Server Name AdminServer
Web Services 0
Web Service Endpoints 0
Number of RESTful Applications 1
Number of RESTful Resources 1
Java EE Web Service Clients 0
Java EE Web Service Client Ports 0
Web Service Details
Web Services
Web Service Endpoints
RESTful Services
Module Name and RESTful Application Name
Resource Type
Resource Path
Invocation Count
Average Execution Time (ms)
AdminServer_/rest-saml-idprop helloworld
0
0

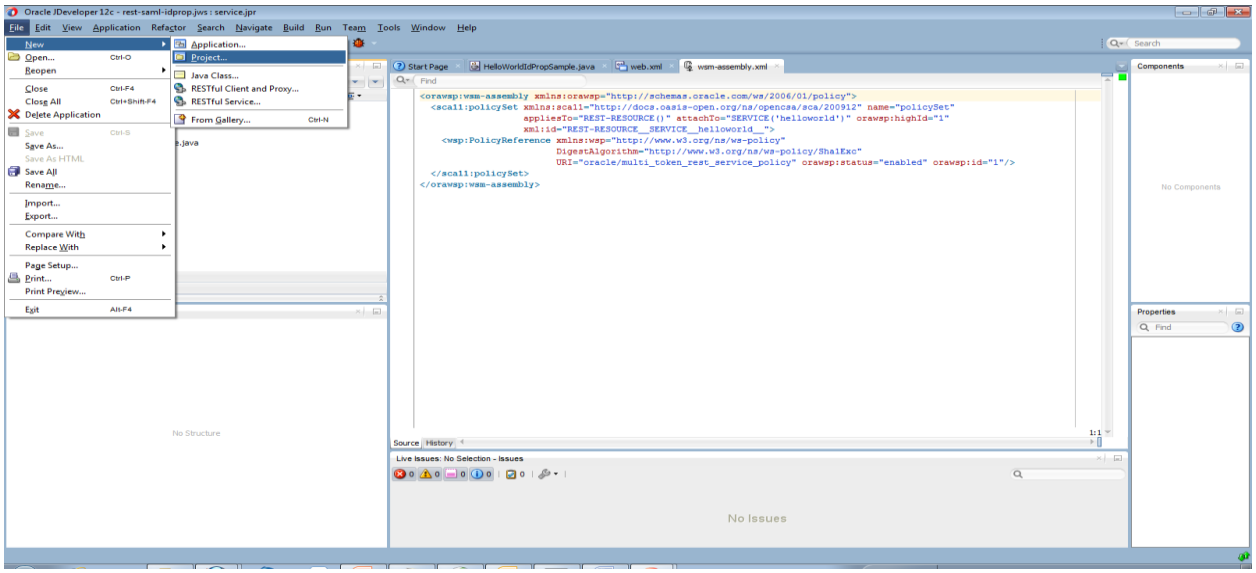

RESTful Application - Oracle Enterprise ...
localhost:7001/em/faces/as/wsmmt/j2eeJaxRsAppSummary?mbeanObjectName=com.bea%3AApplicationRuntime%3Dhelloworld%2CLocation%3DAdminServer%2CName%3Dhelloworld%2CServerRu
ORACLE Enterprise Manager Fusion Middleware Control 12c
Target Navigation
helloworld
Application Deployment
Start Up
Shut Down...
Web Services > RESTful Application
helloworld (RESTful Application)
Test RESTful Service
Application Name helloworld
Module Name AdminServer_/rest-saml-idprop
RESTful Application Name helloworld
Number of Resources 1
Number of Methods 3
Number of Sub Resource Locators 0
Invocation Count 0
Error Count 0
Average Execution Time (ms) 0
WADL Document helloworld
RESTful Resources
WSM Policies
Select an expression from the Contraint dropdown to view the corresponding effective policy references. For policy set flagged as "Not Valid", click the link to view the validation error details. For security policy references, click the violations count link to view violation details. When policies are attached/detached, effective policy references are recalculated.
Constraint None
Status Secure
Globally Attached Policies
Category/Policy Name
Policy Set
Status
Total Violations
no rows yet
Directly Attached Policies
View
Attach/Detach
Enable
Disable
Override Policy Configuration
Effective Only
All
Detach
Category/Policy Name
Effective
Status
Total Violations
security
oracle/multi_token_rest_service_policy
True
Enabled
0


Mozilla Firefox
http://pyamuna-lap....es/application.wadl
pyamuna-lap.oradev.oraclecorp.com:7001/rest-saml-idprop/resources/application.wadl
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<ns0:application>
<ns0:doc ns1:generatedBy="Jersey: 1.17.1 02/28/2013 12:47 PM"/>
<ns0:grammars/>
<ns0:resources base="http://pyamuna-lap.oradev.oraclecorp.com:7001/rest-saml-idprop/resources/">
<ns0:resource path="hello">
<ns0:resource path="user">
<ns0:method id="hello" name="GET">
<ns0:response>
<ns0:representation mediaType="text/plain"/>
</ns0:response>
</ns0:method>
</ns0:resource>
</ns0:resource>
</ns0:resources>
</ns0:application>

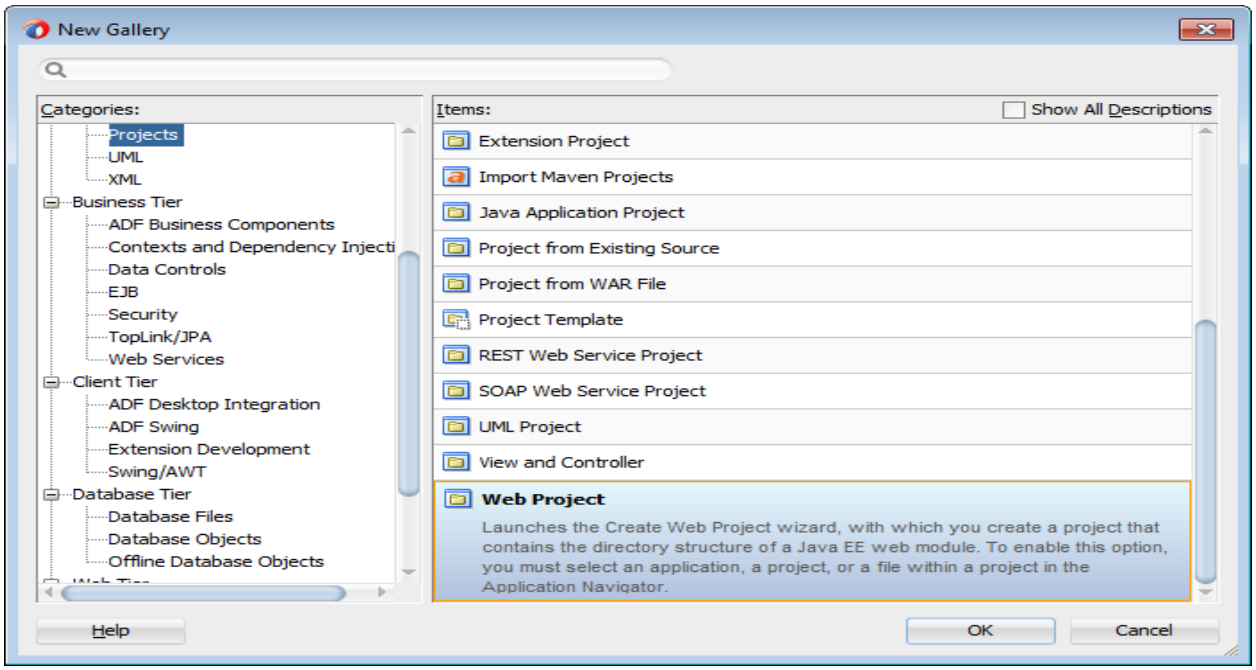
4 Create REST client

4.1 Create a "rest-client" Web Project

Create a new project by clicking on "File->New->Project" as shown in the screenshot below.

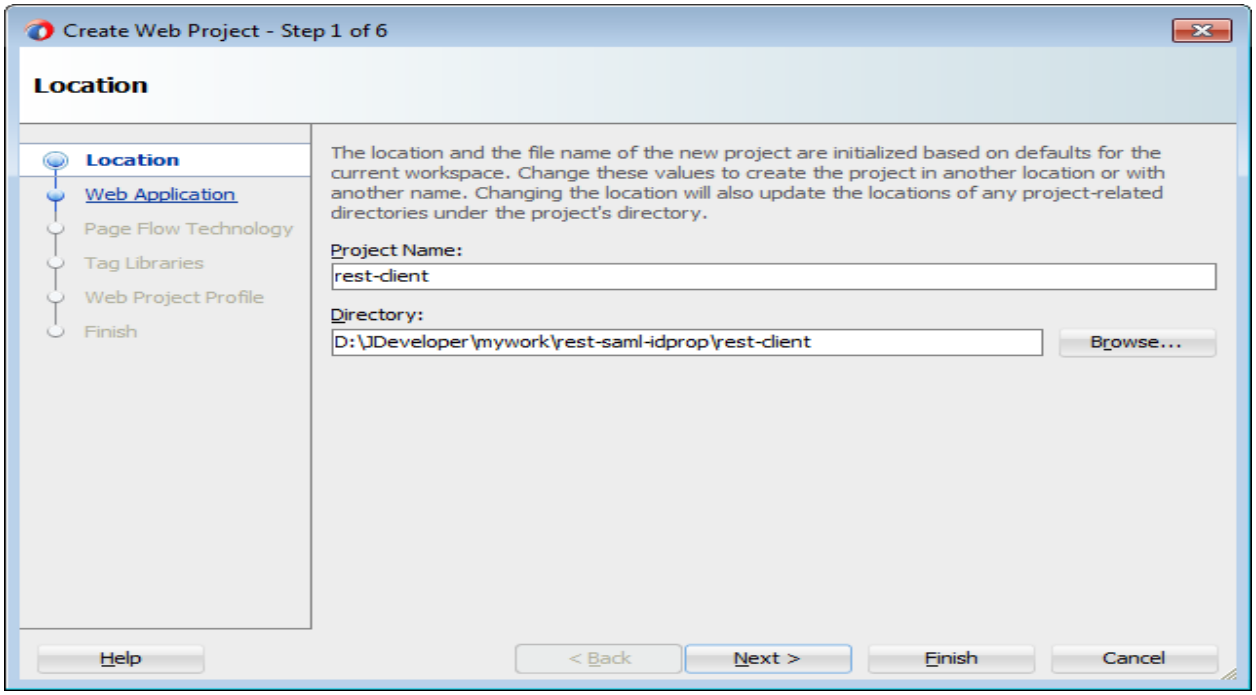


Select "Web Project" from the "New Gallery" as shown in the screenshot below.

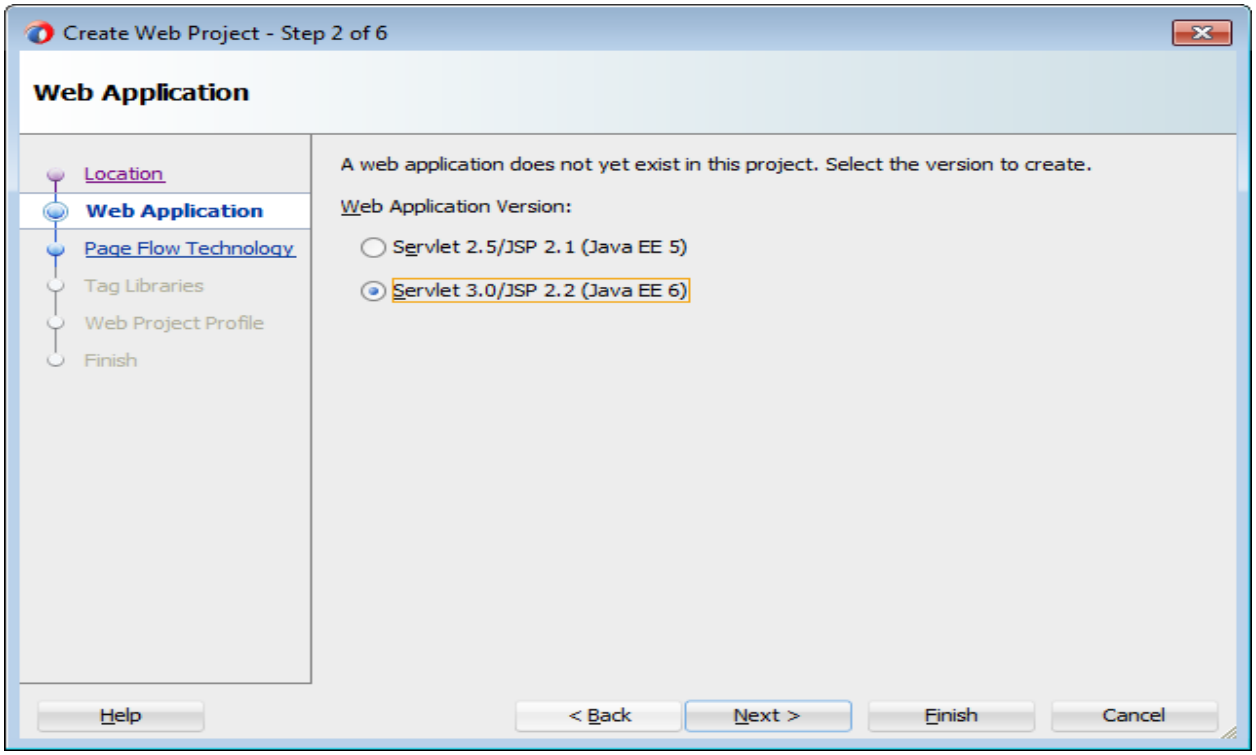


Click "OK".

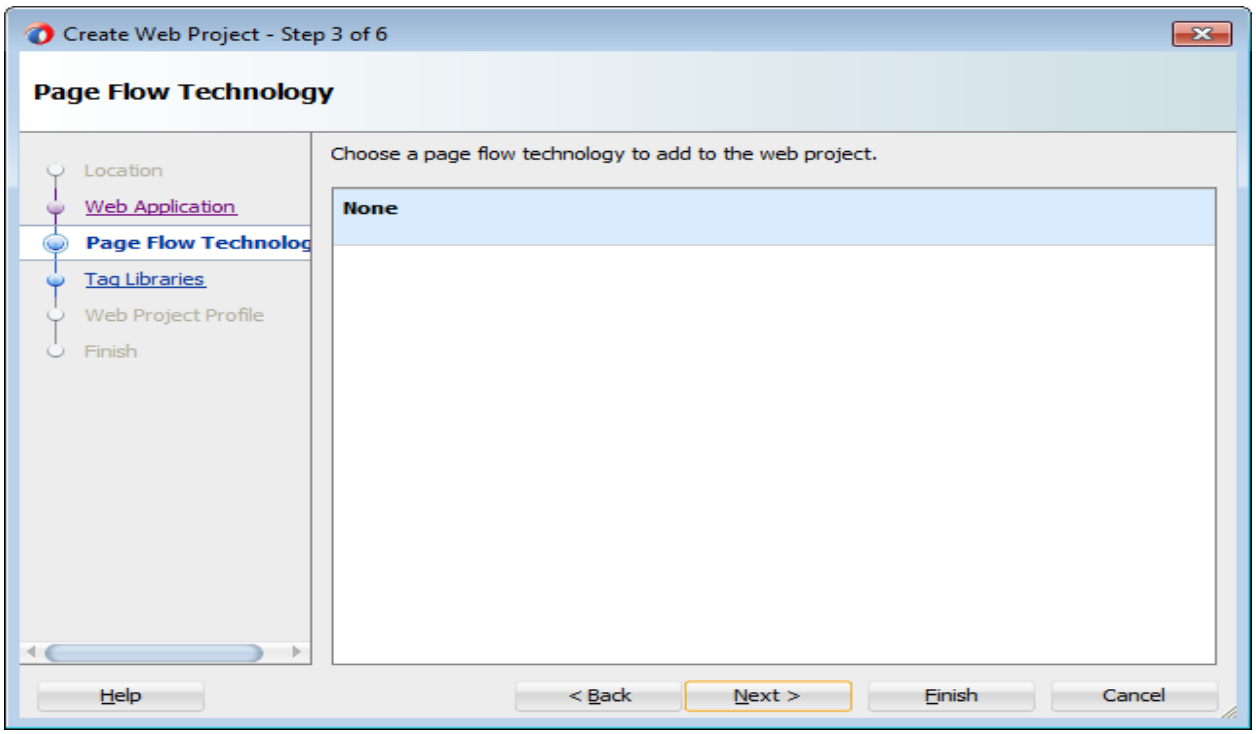
We will create “Web Project” since the client will act as a JEE Client.



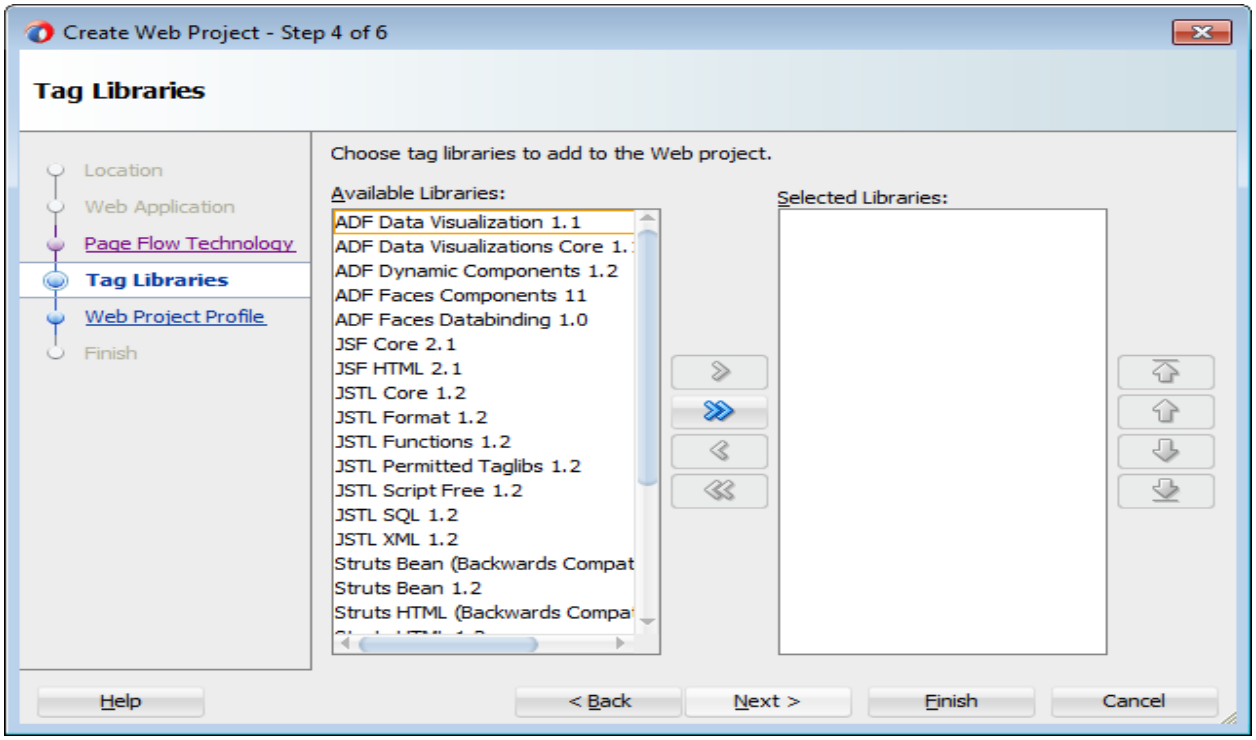
Click “Next”



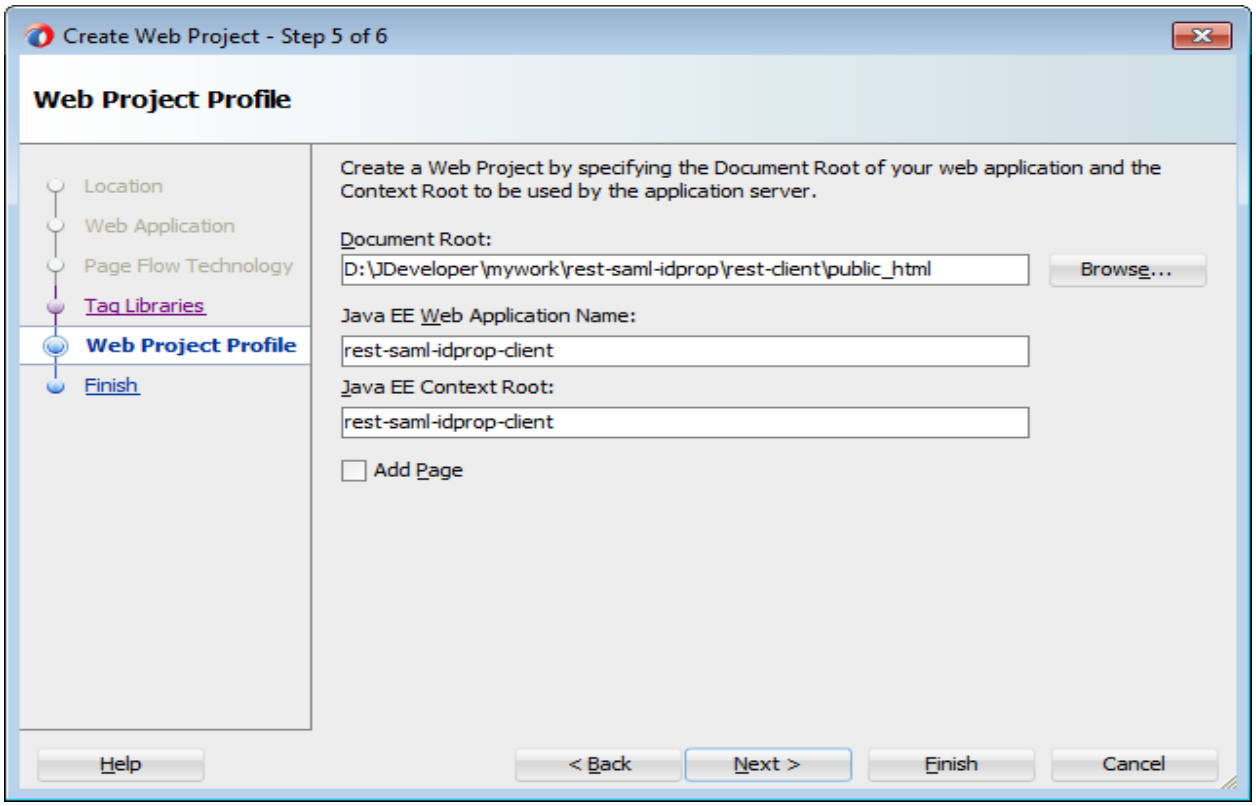
Click “Next”



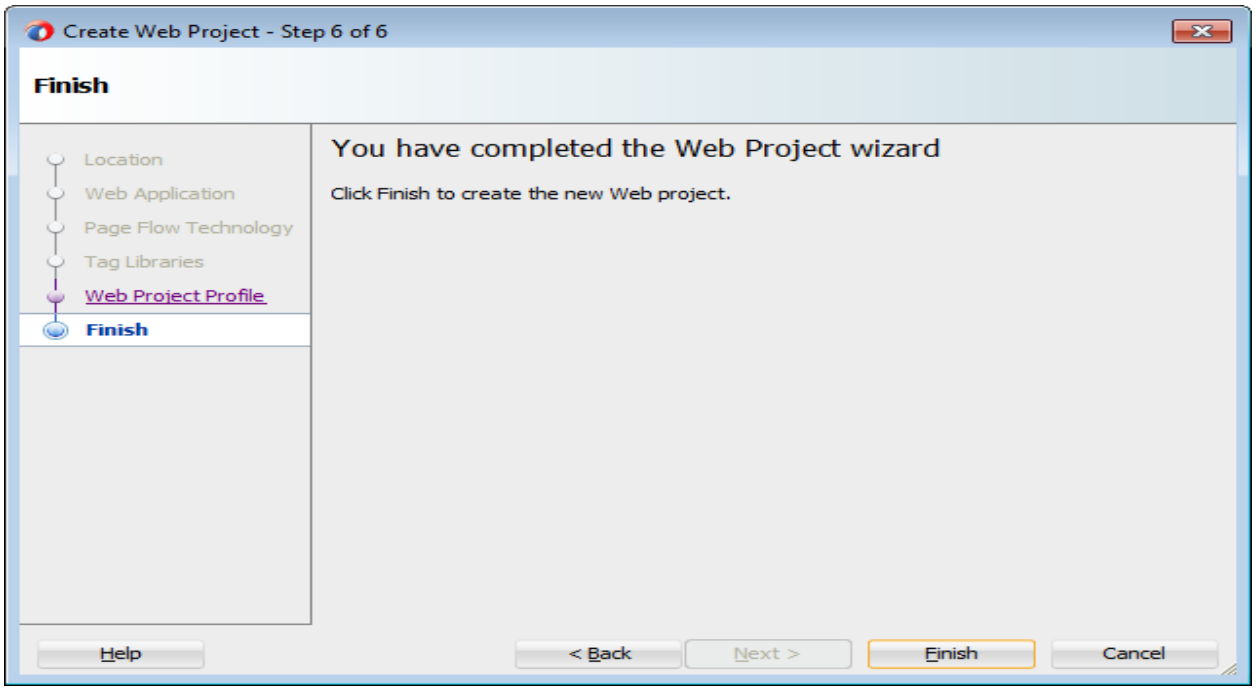
Click “Next”



Click “Next”



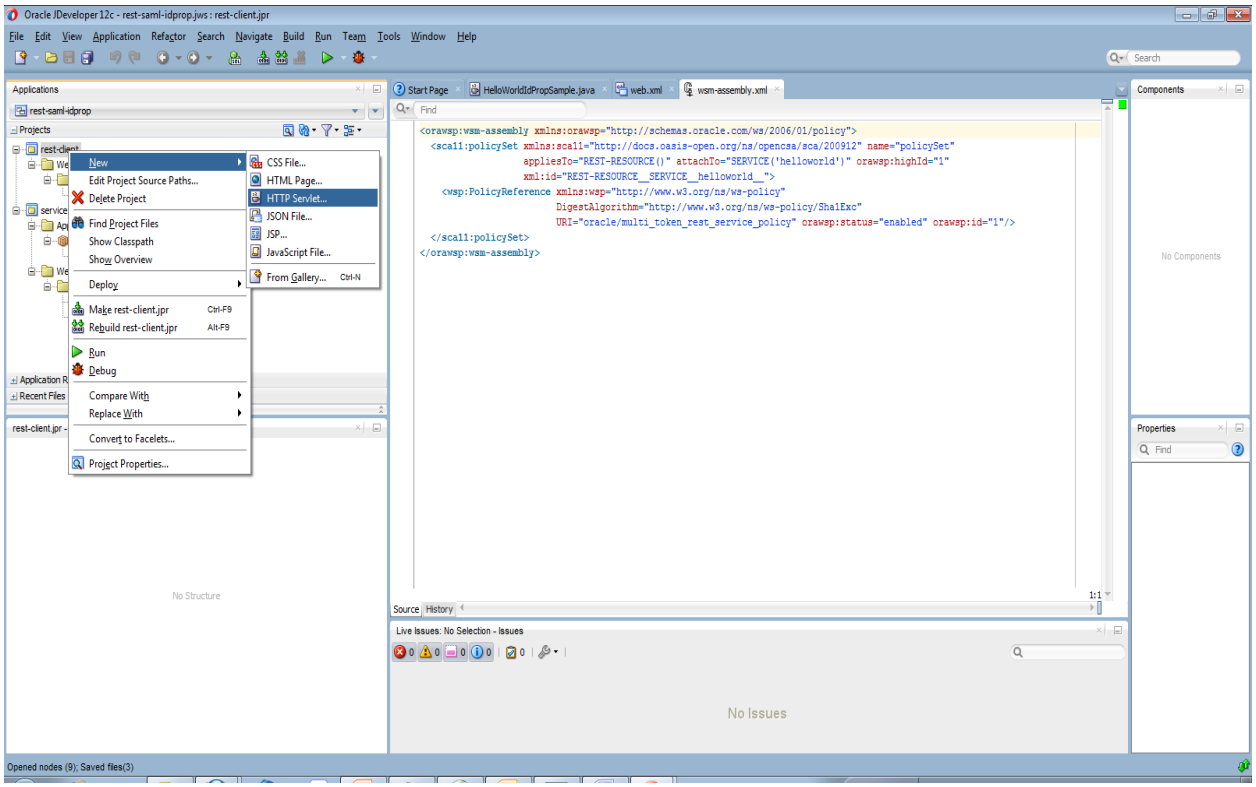
Click “Next”

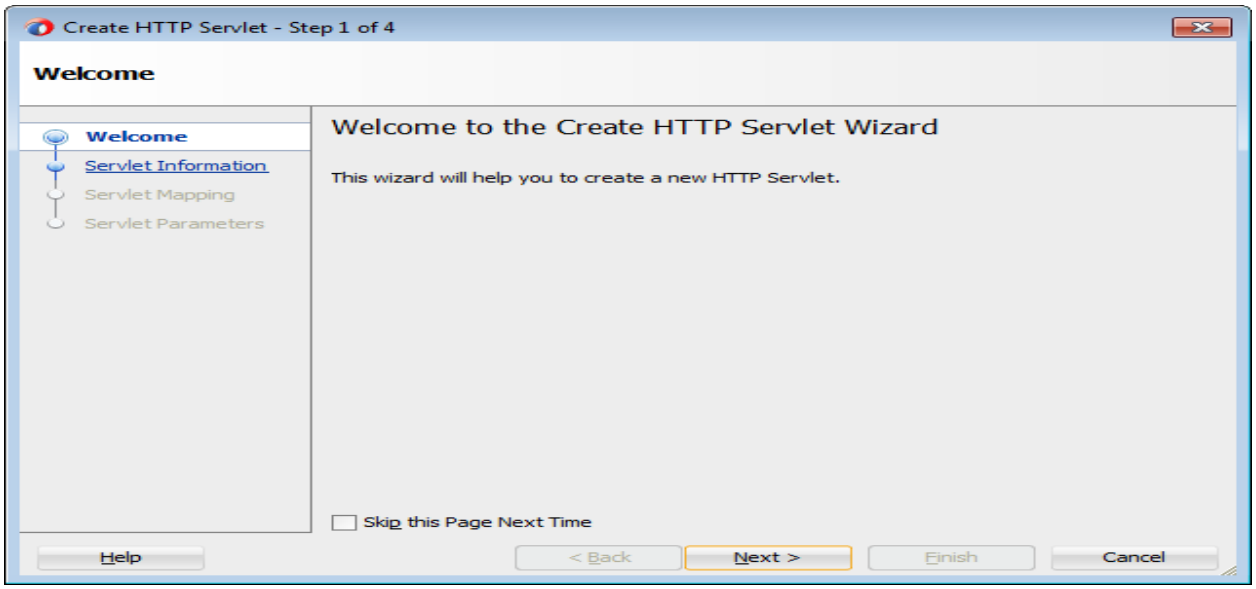


Click “Finish”

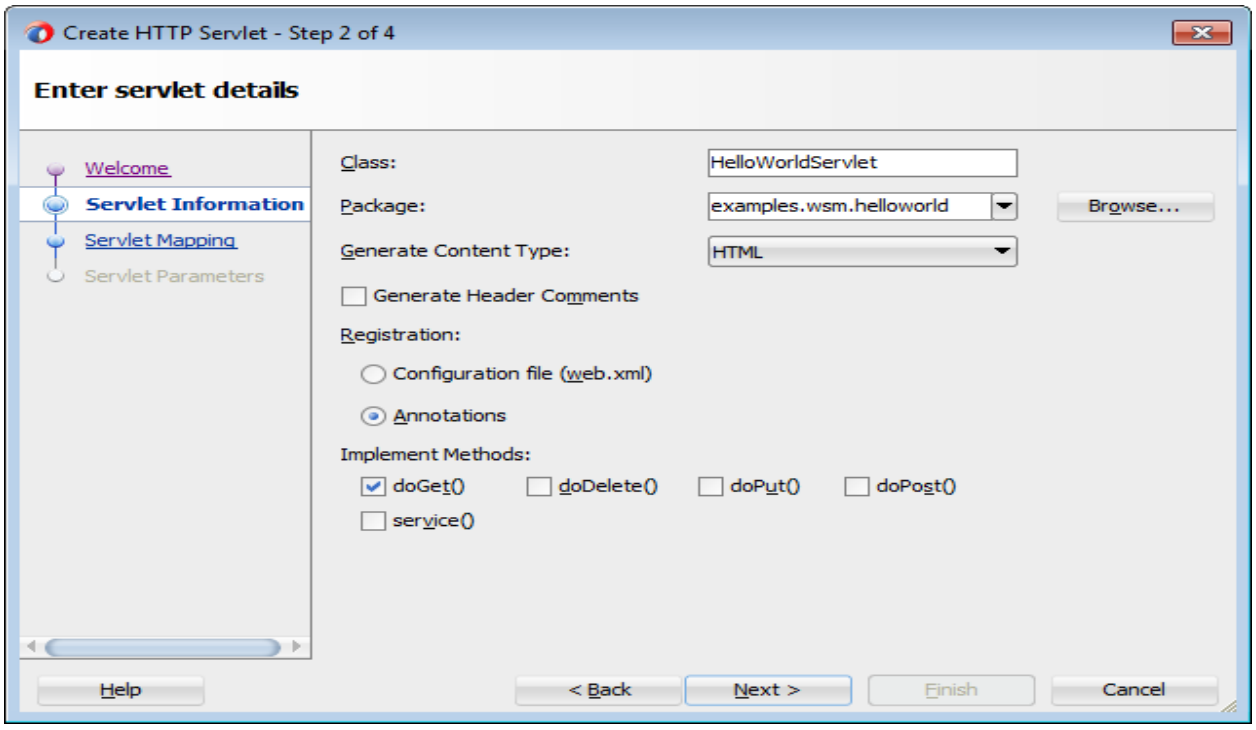
4.2 Create "HelloWorld Servlet"

Create a HTTPServlet which will act as the REST Client. You can create a Servlet by right-clicking on the project and selecting "New->HTTPServlet" from the context menu.

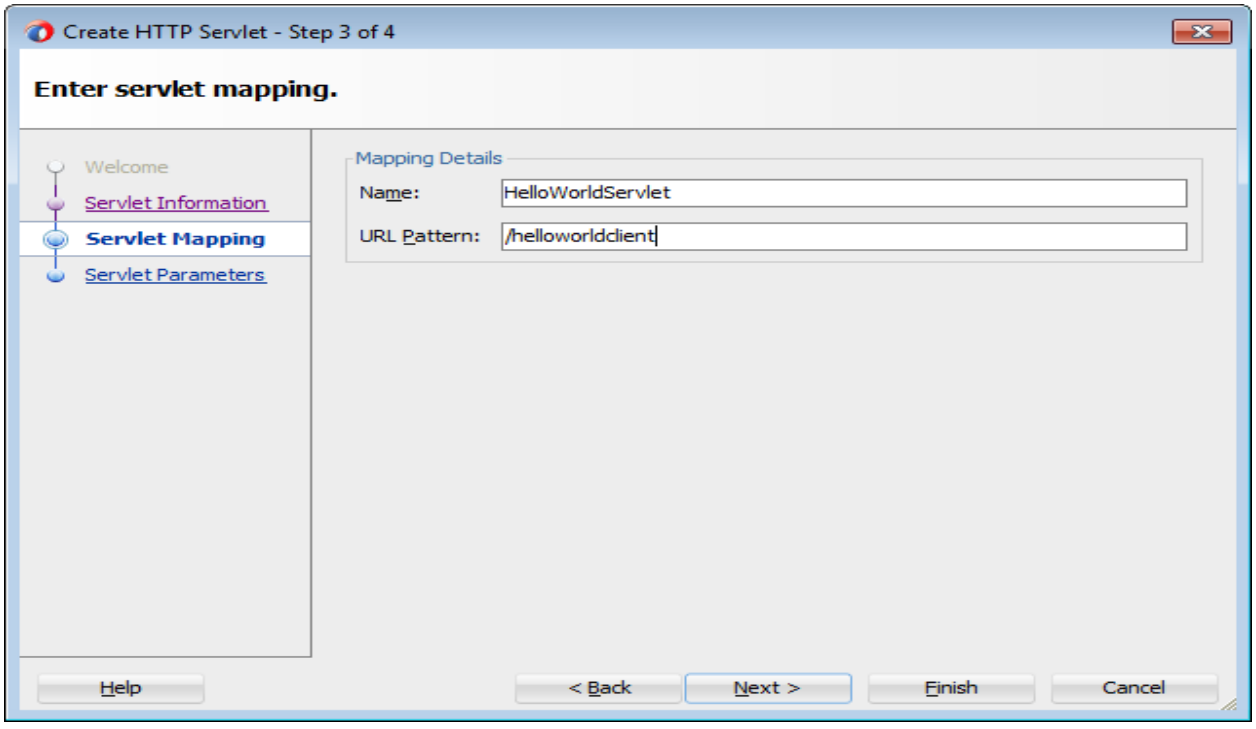




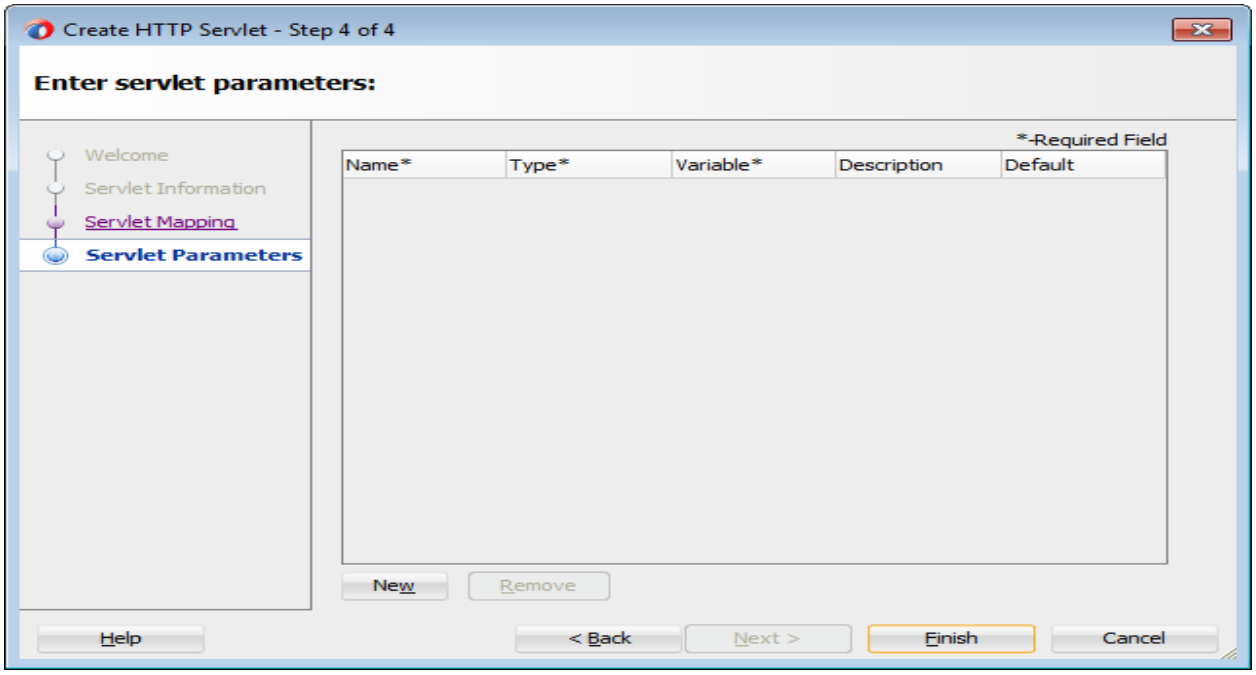
Click "Next"



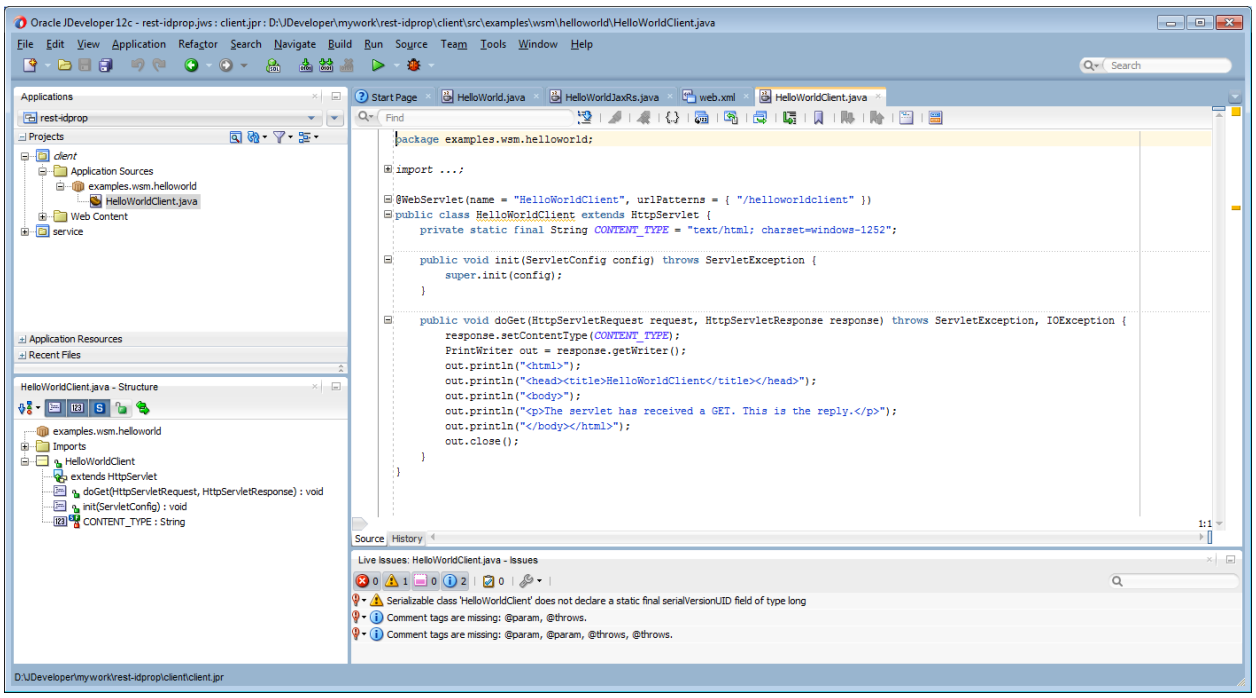
Click “Next”



Click “Next”

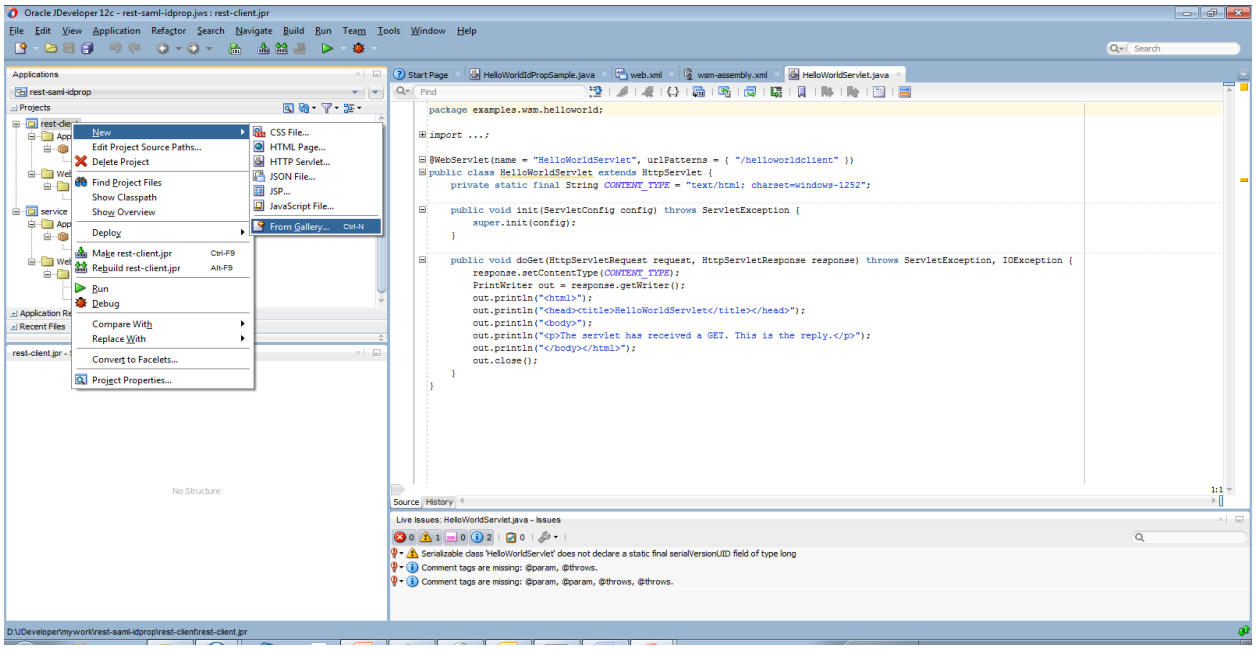


Click “Finish”

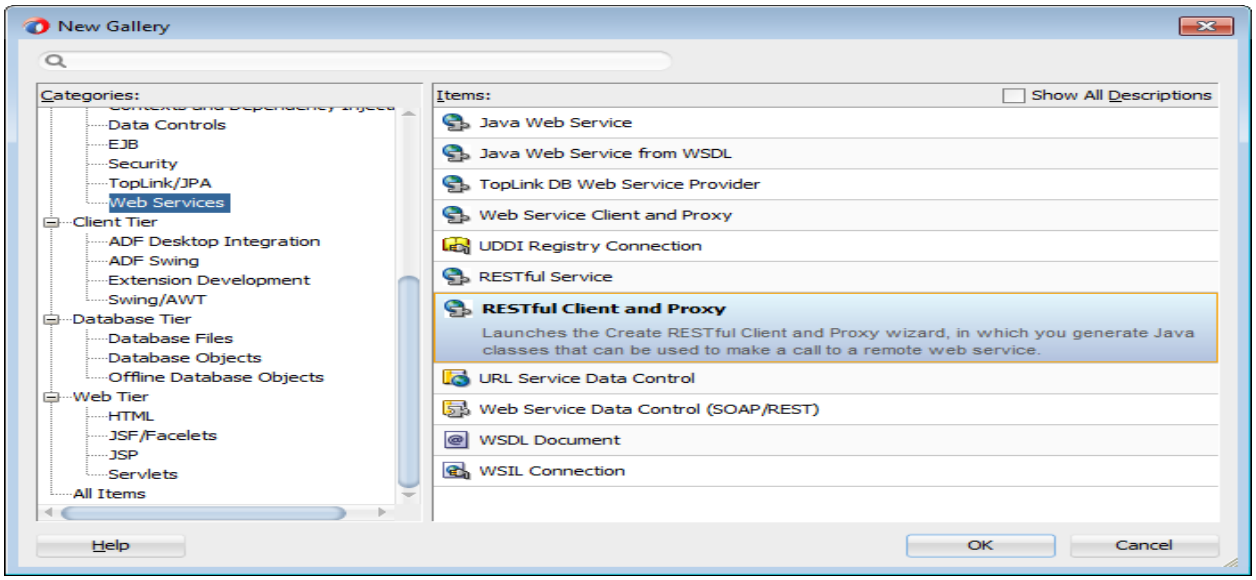


4.3 Create REST Client Proxy

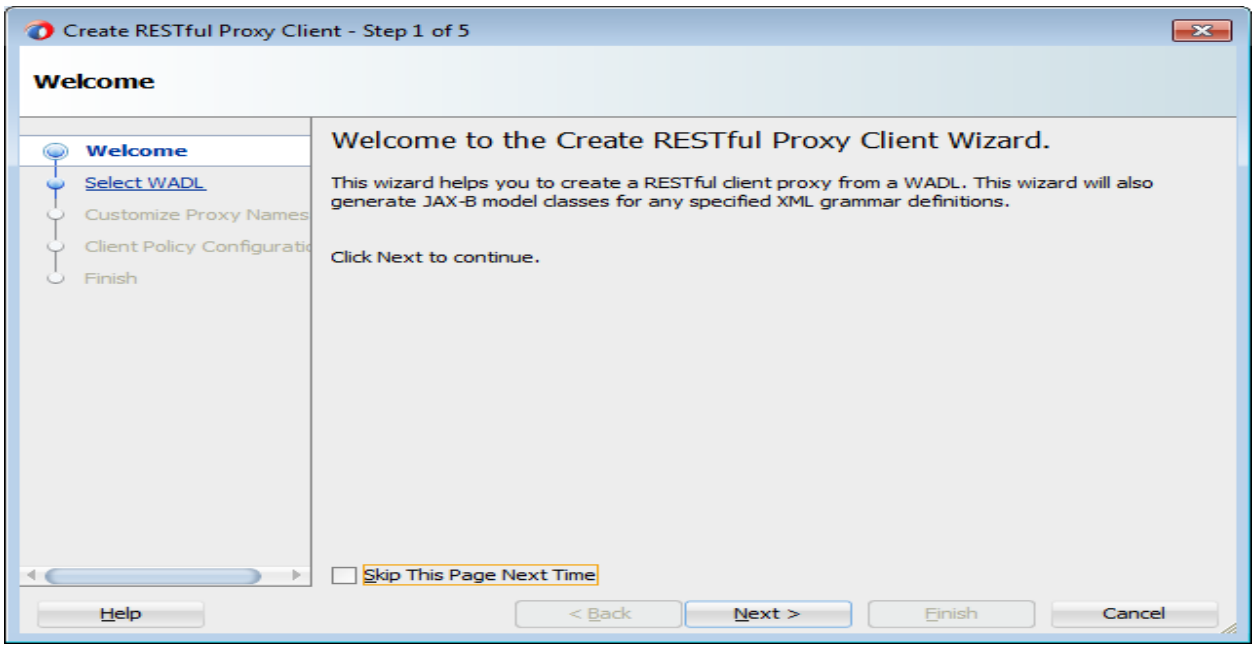
You can create a REST Client Proxy by right-clicking on the project and selecting “New->From Gallery” as shown below.



Select the “Web Services” category from the “New Gallery” dialog on the Left Hand side. This will show all the options related to Web Services on the Right hand side. Select “Restful Client and Proxy” from RHS to create a REST client as shown in the screenshot below.

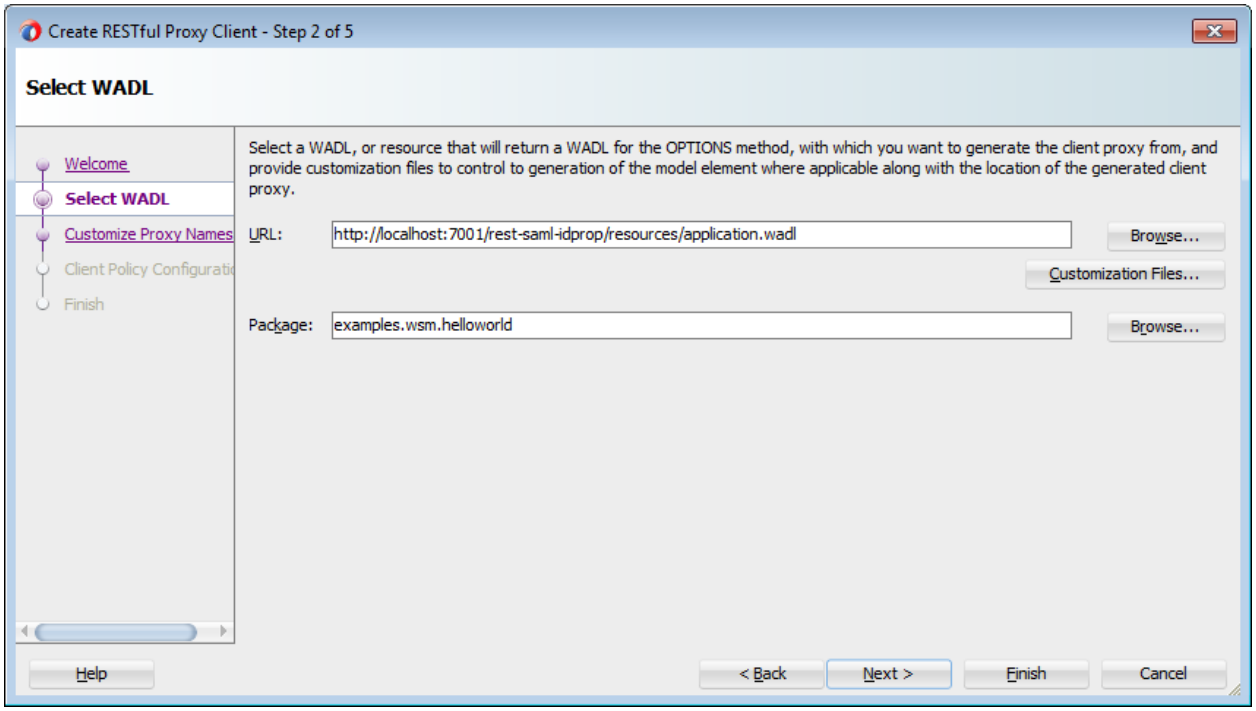


Click “OK”



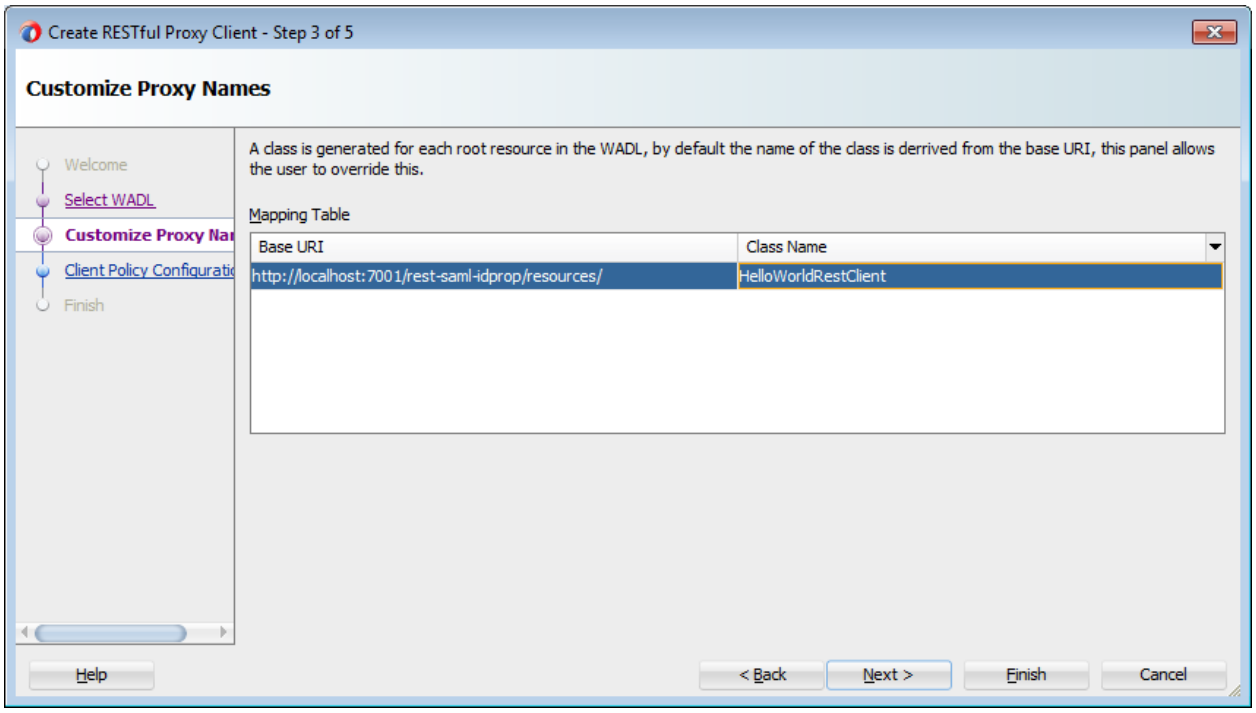
Click “Next”

To create a RESTful Proxy Client – you will need the WADL of the REST service.



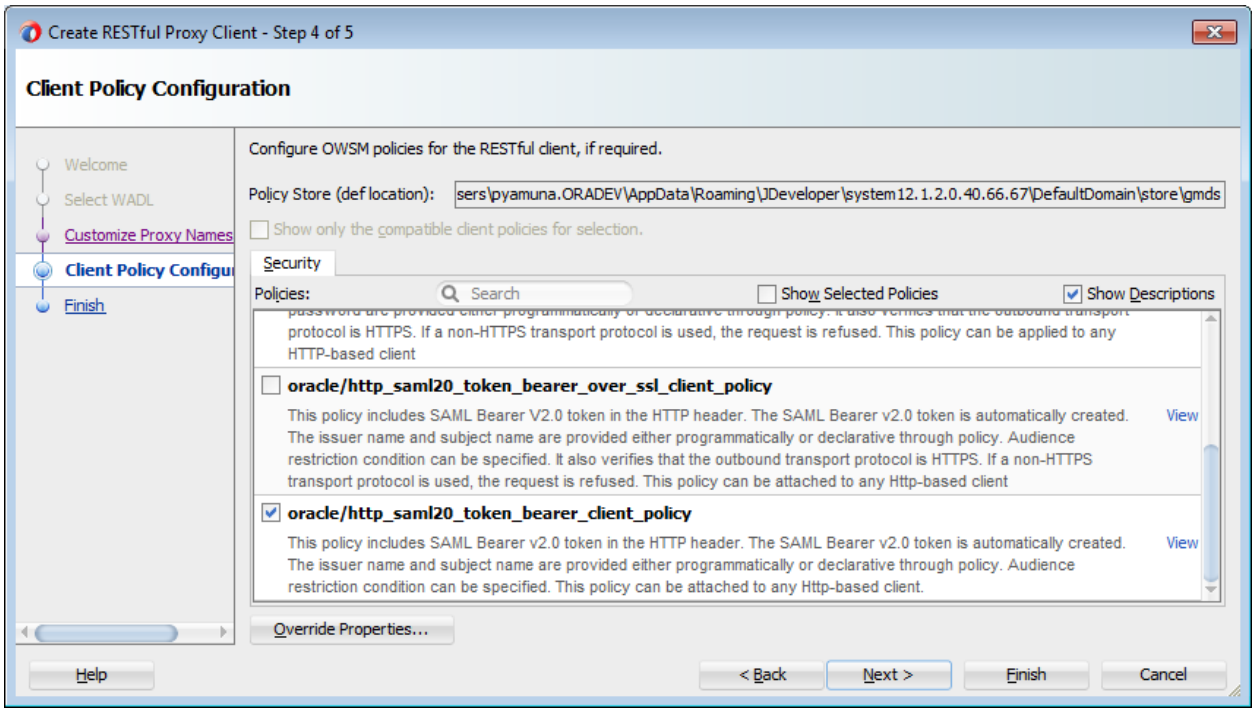
NOTE: Either you need to save the WADL of the service a priori or have the RESTful Service running in order to be able to provide the WADL url.

Click “Next”

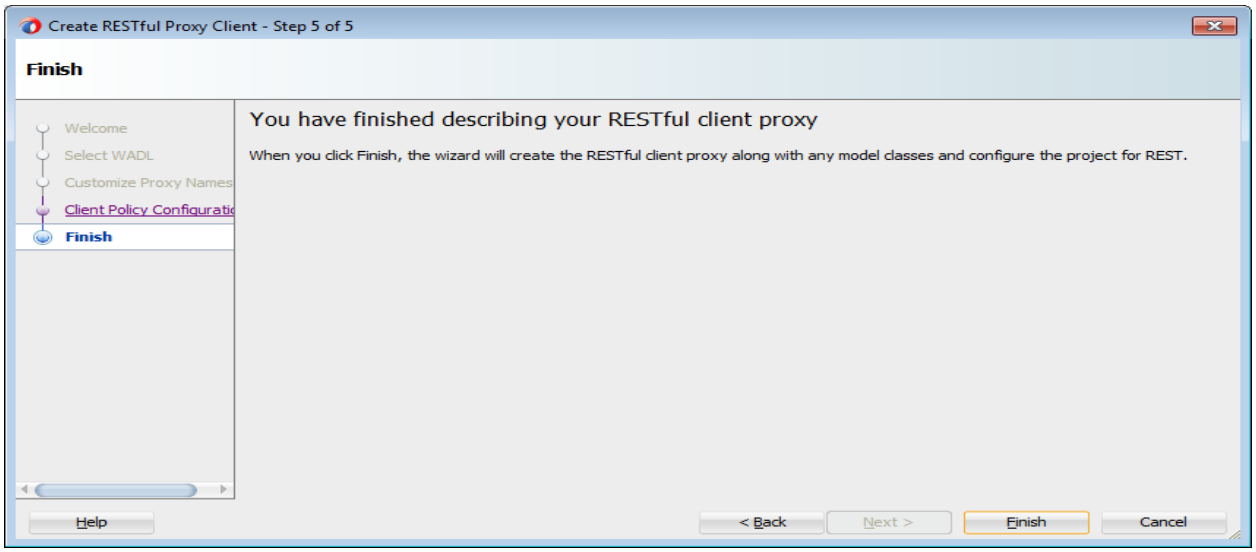


By default JDeveloper generates a classname for the Proxy client. Provide a more user friendly name in this dialog as shown in the screenshot above.

Click “Next”



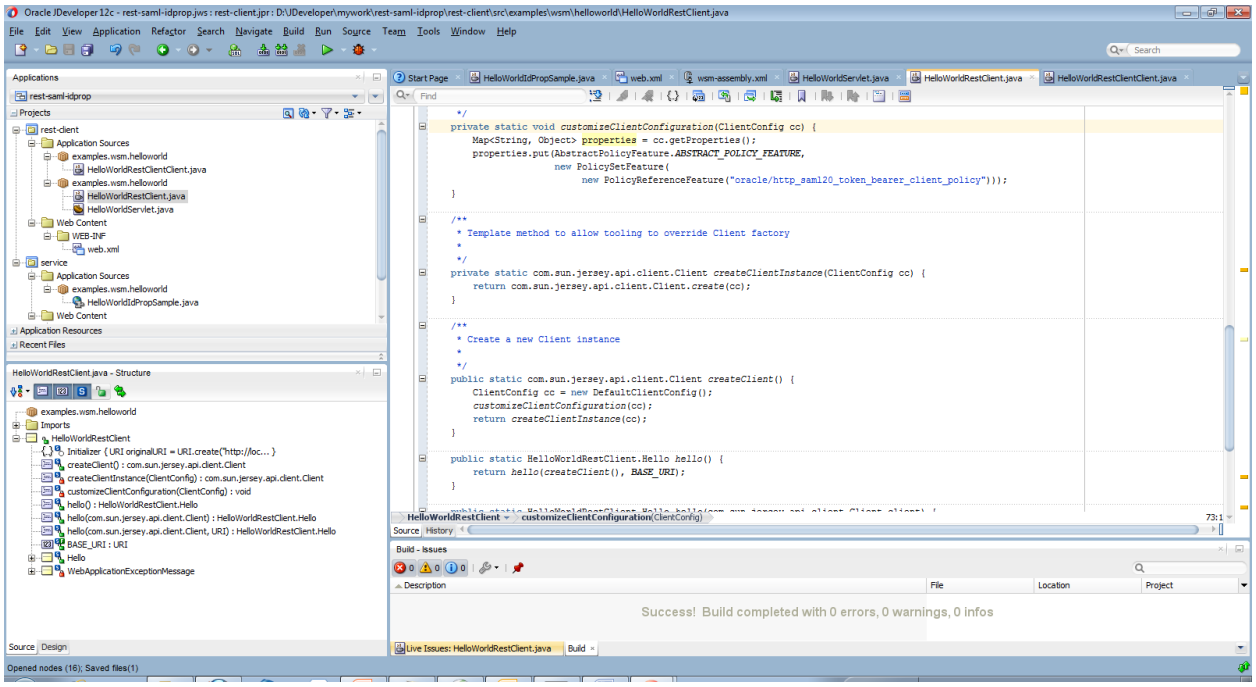
Select the OWSM policy you want to use to secure the client and click “Next”.



IMPORTANT: Although JDeveloper provides you a policy selection dialog to select the OWSM policy in the REST Client Proxy creation wizard – there is a bug in JDeveloper - due to which the Policy is not added to the Client Proxy code. You will need to manually add the following lines of code show below.

Click "Finish"

4.4 Secure the REST Client Proxy



```
import com.sun.jersey.api.client.GenericType;

import com.sun.jersey.api.client.config.ClientConfig;

import com.sun.jersey.api.client.config.DefaultClientConfig;

import oracle.wsm.metadata.feature.AbstractPolicyFeature;

import oracle.wsm.metadata.feature.PolicyReferenceFeature;

import oracle.wsm.metadata.feature.PolicySetFeature;

import oracle.wsm.metadata.feature.PropertyFeature;

import oracle.wsm.security.util.SecurityConstants;

import weblogic.jaxrs.api.client.Client;

//…

private static void customizeClientConfiguration(ClientConfig cc) {

      // Client policy features configuration.

      Map<String, Object> properties = cc.getProperties();

      properties.put(AbstractPolicyFeature.ABSTRACT_POLICY_FEATURE,

                  new PolicySetFeature(new
PolicyReferenceFeature("oracle/http_saml20_token_bearer_client_policy")));

   }

   public static com.sun.jersey.api.client.Client createClient() {

      ClientConfig cc = new DefaultClientConfig();

      customizeClientConfiguration(cc);

      return createClientInstance(cc);

   }
```

Pay special attention to the import statements in the code snippet above.

```
import oracle.wsm.metadata.feature.AbstractPolicyFeature;

import oracle.wsm.metadata.feature.PolicyReferenceFeature;

import oracle.wsm.metadata.feature.PolicySetFeature;

import oracle.wsm.metadata.feature.PropertyFeature;

import oracle.wsm.security.util.SecurityConstants;

import weblogic.jaxrs.api.client.Client;
```

Few other things to note:

- The oracle/ oracle/http_saml20_token_bearer_client_policy – creates a SAML bearer token from the Subject.

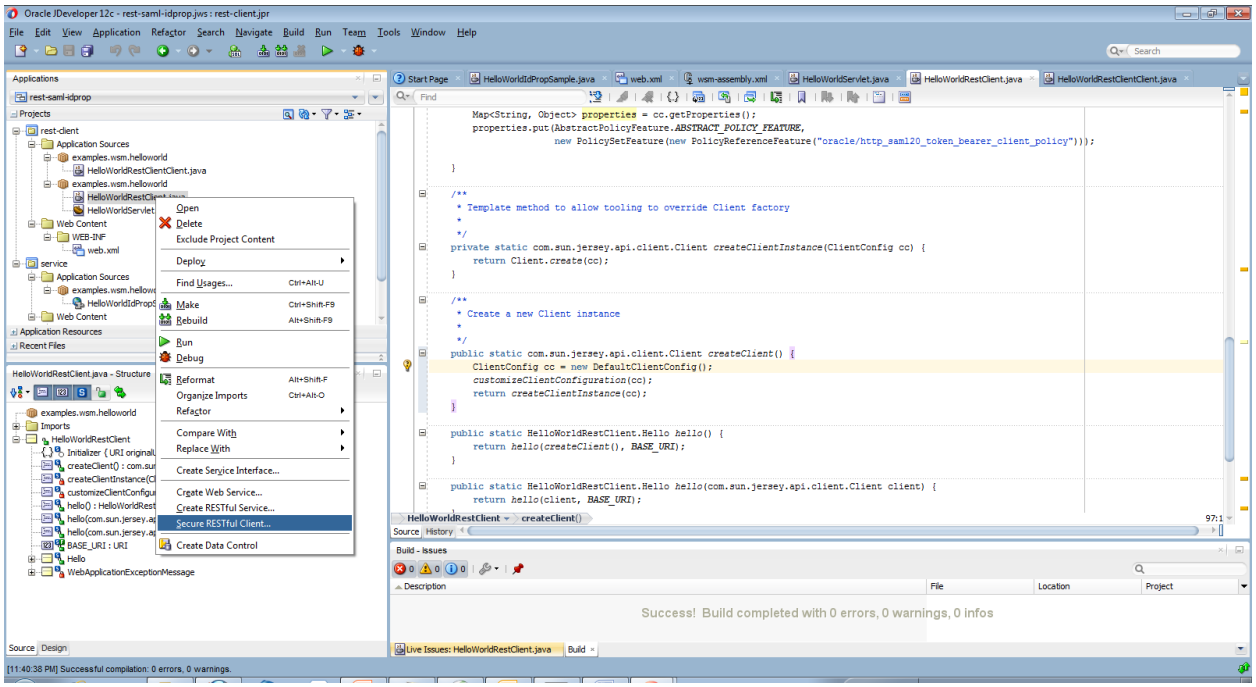
4.5 Modify the HelloWorldServlet to call the RESTful Client Proxy

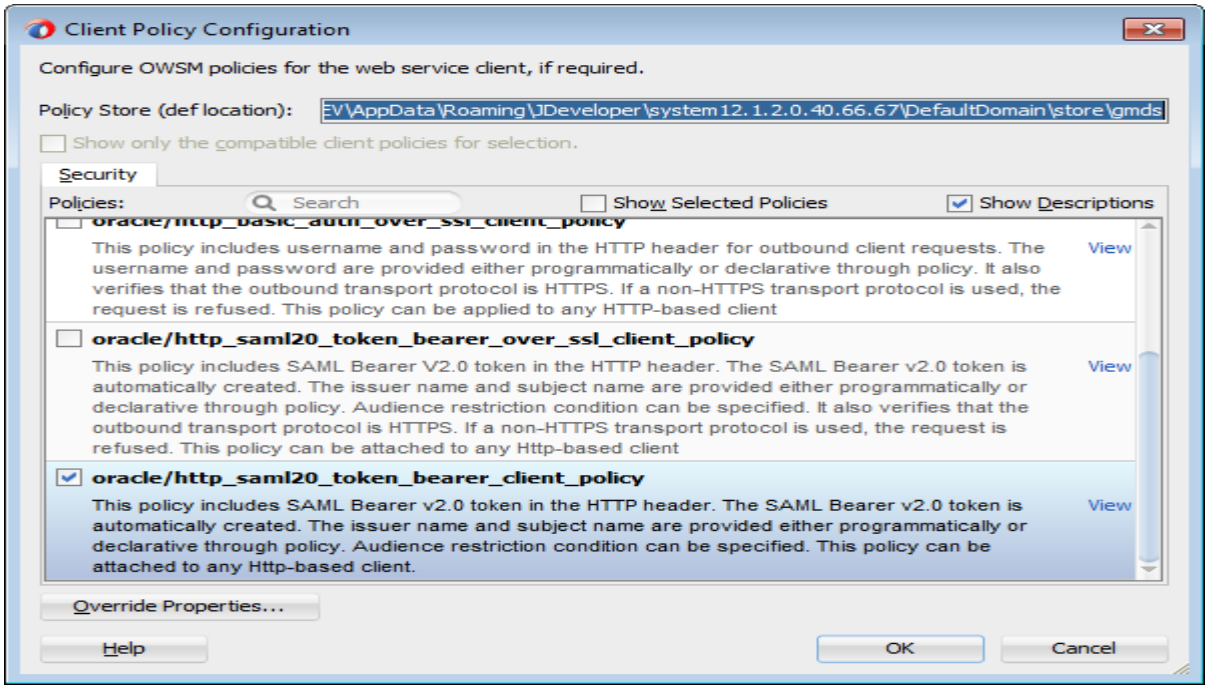
```
package examples.wsm.helloworld;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.*;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.*;

import com.sun.jersey.api.client.Client;

import examples.wsm.helloworld.HelloWorldRestClient.Hello;

@WebServlet(name = "HelloWorldServlet", urlPatterns = { "/helloworldclient" })

public class HelloWorldServlet extends HttpServlet {

//…

   public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

      response.setContentType(CONTENT_TYPE);

//…

      Client client = HelloWorldRestClient.createClient();

      HelloWorldRestClient.Hello hello = HelloWorldRestClient.hello(client);

      String output = hello.user().getAsTextPlain(String.class);

      out.println("Output from REST service:"+output);

      out.println();

//…

     out.close();

   }

}
```

For purposes of brevity I have provided the code snippet for how to call the generated RESTful Proxy Client code above.

4.6 Modifying the REST Client Security Policy

One can modify the OWSM client policy used to secure the client by right-clicking on the RESTful Client Proxy class and selecting "Secure RESTful Client" from the context menu as shown below.



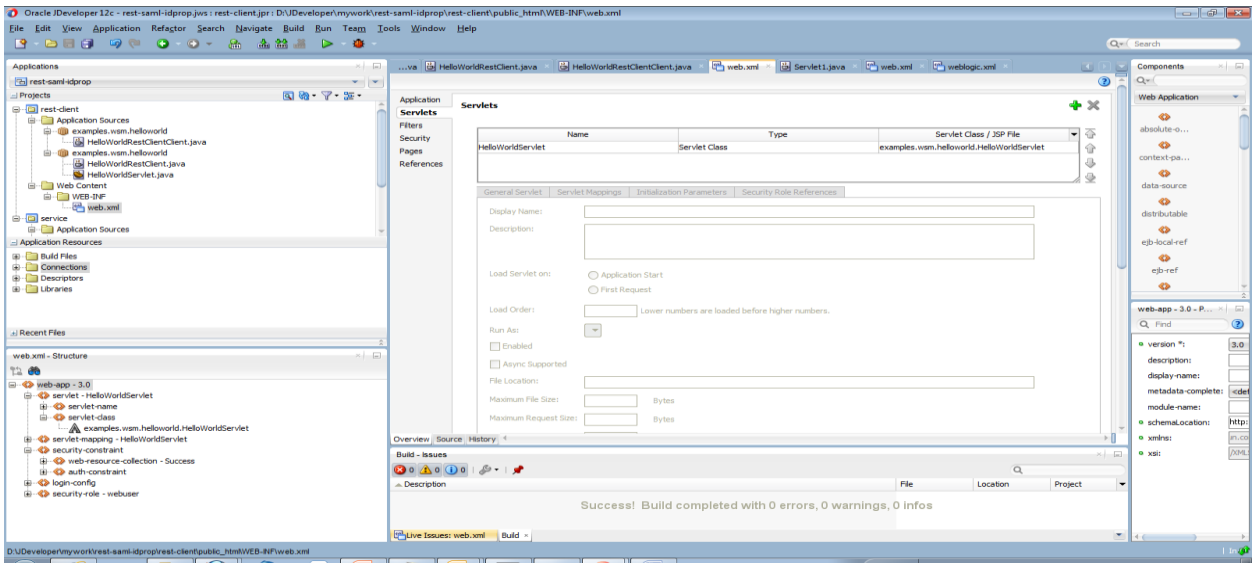


You can use the above dialog to make a different policy selection if required. In this case I did not modify the policy since I had already selected the correct policy.

NOTE: Any changes you do in terms of selecting a different policy via this dialog will be effective and the corresponding code in the RESTful Proxy Client will be modified.

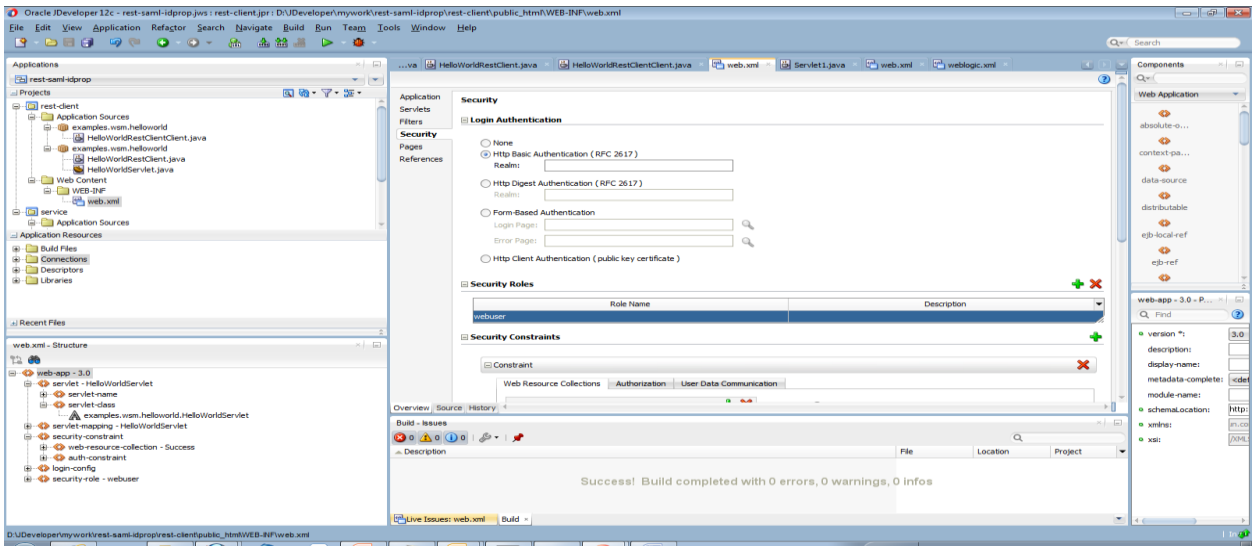
4.7 Secure the HelloWorldServlet web application

Open web.xml for the "rest-client" Project in JDeveloper and add an entry for the HelloWorldServlet as shown below.

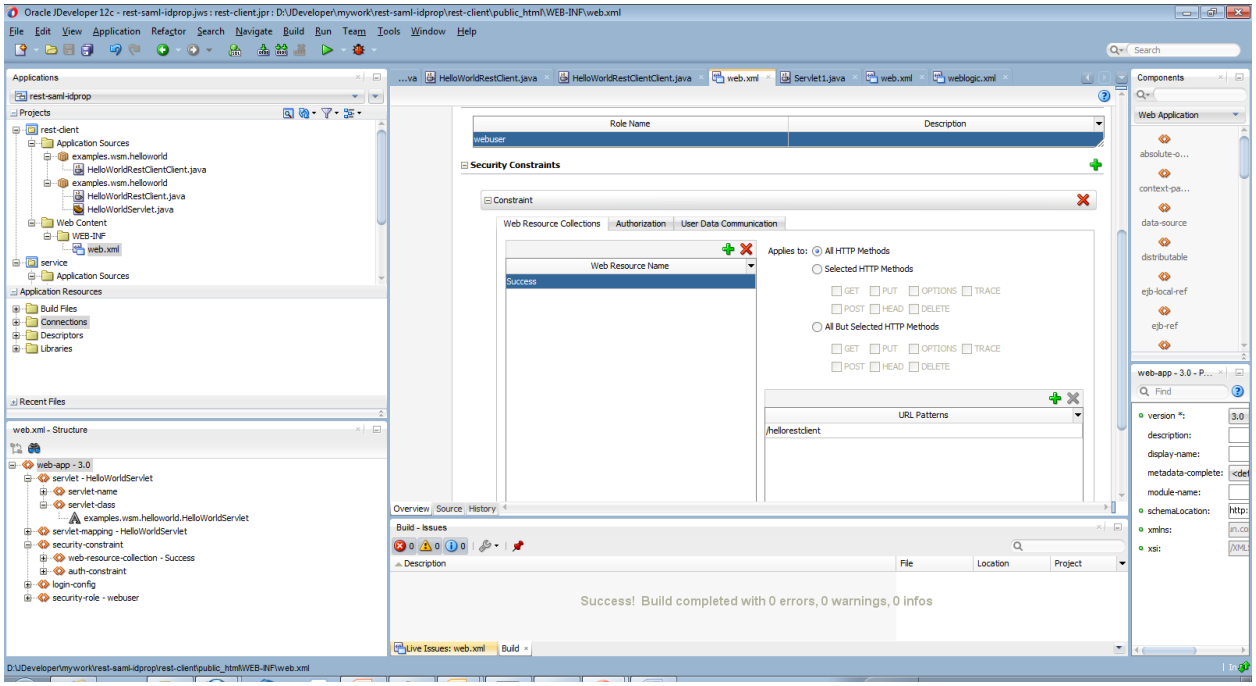


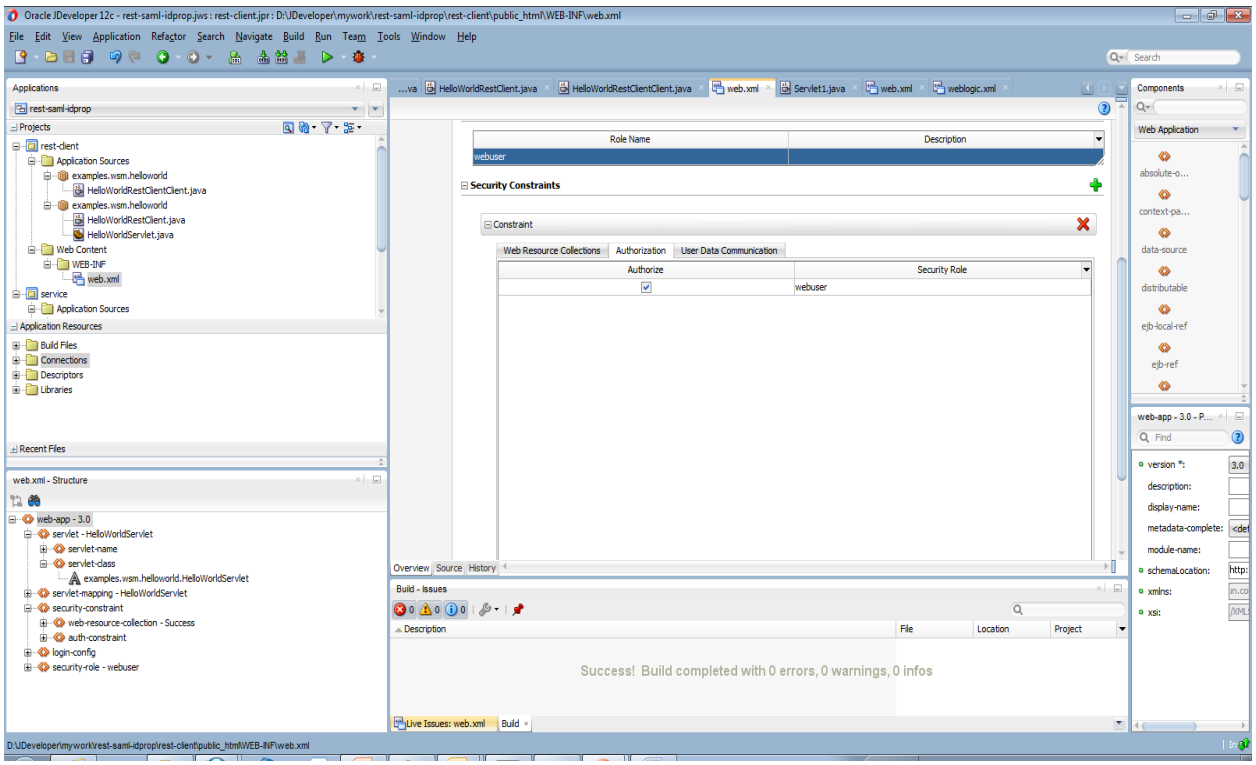
4.7.1 Select Authentication Mechanism

Select "HTTP Basic Authentication" mechanism.



4.7.2 Add Security Constraints



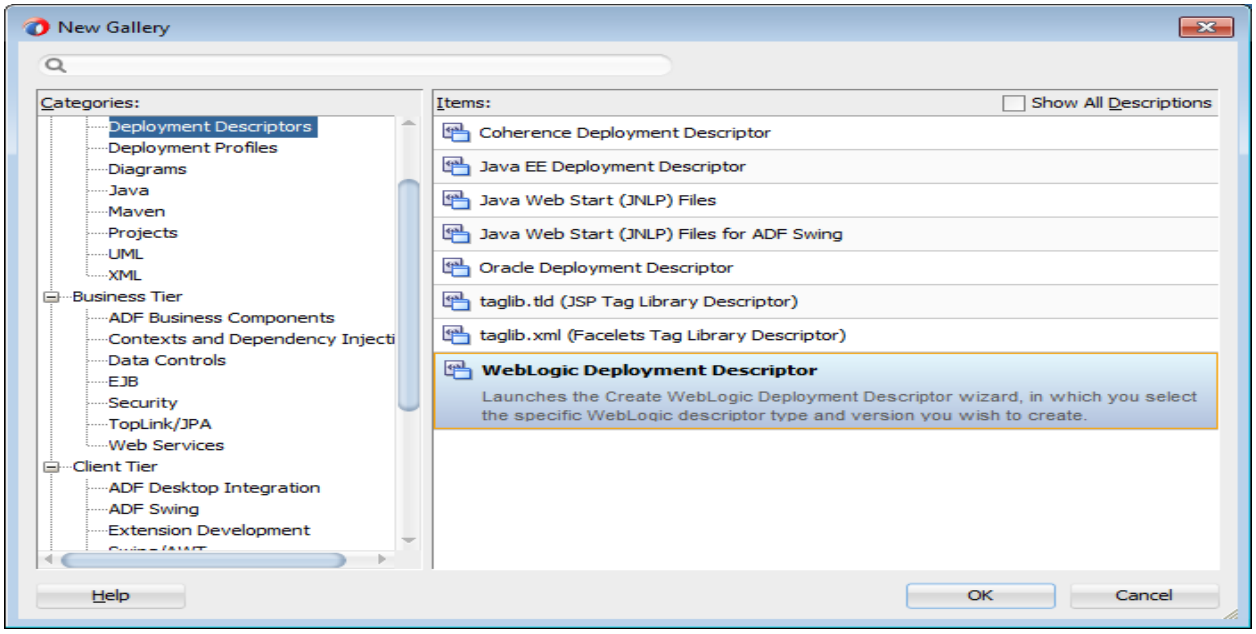


4.7.3 Final "web.xml" for the HelloWorldServlet

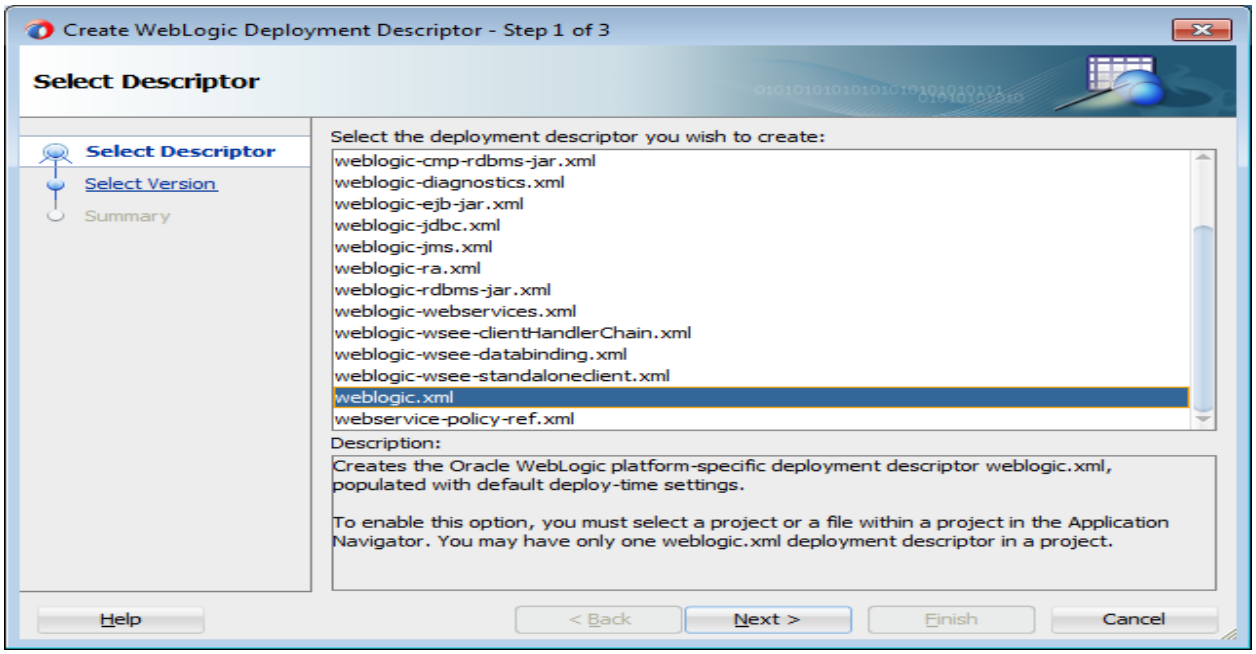
```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      version="3.0">
 <servlet>
  <servlet-name>HelloWorldServlet</servlet-name>
  <servlet-class>examples.wsm.helloworld.HelloWorldServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>HelloWorldServlet</servlet-name>
  <url-pattern>/hellorestclient</url-pattern>
 </servlet-mapping>
 <security-constraint>
  <web-resource-collection>
   <web-resource-name>Success</web-resource-name>
   <url-pattern>/hellorestclient</url-pattern>
  </web-resource-collection>
  <auth-constraint>
   <role-name>webuser</role-name>
  </auth-constraint>
 </security-constraint>
 <login-config>
  <auth-method>BASIC</auth-method>
 </login-config>
 <security-role>
  <role-name>webuser</role-name>
 </security-role>
</web-app>
```

4.7.4 Create weblogic.xml deployment descriptor

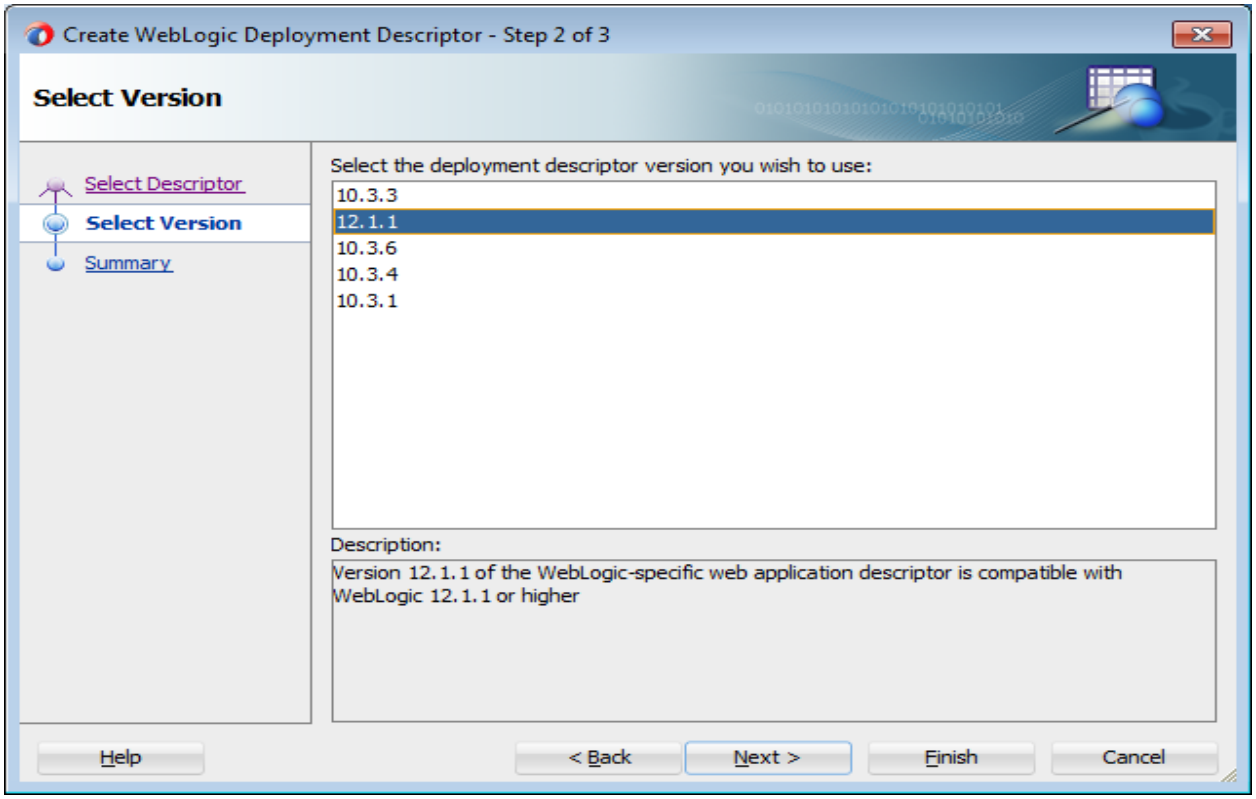
Right click on the project and select “New->From Gallery”.



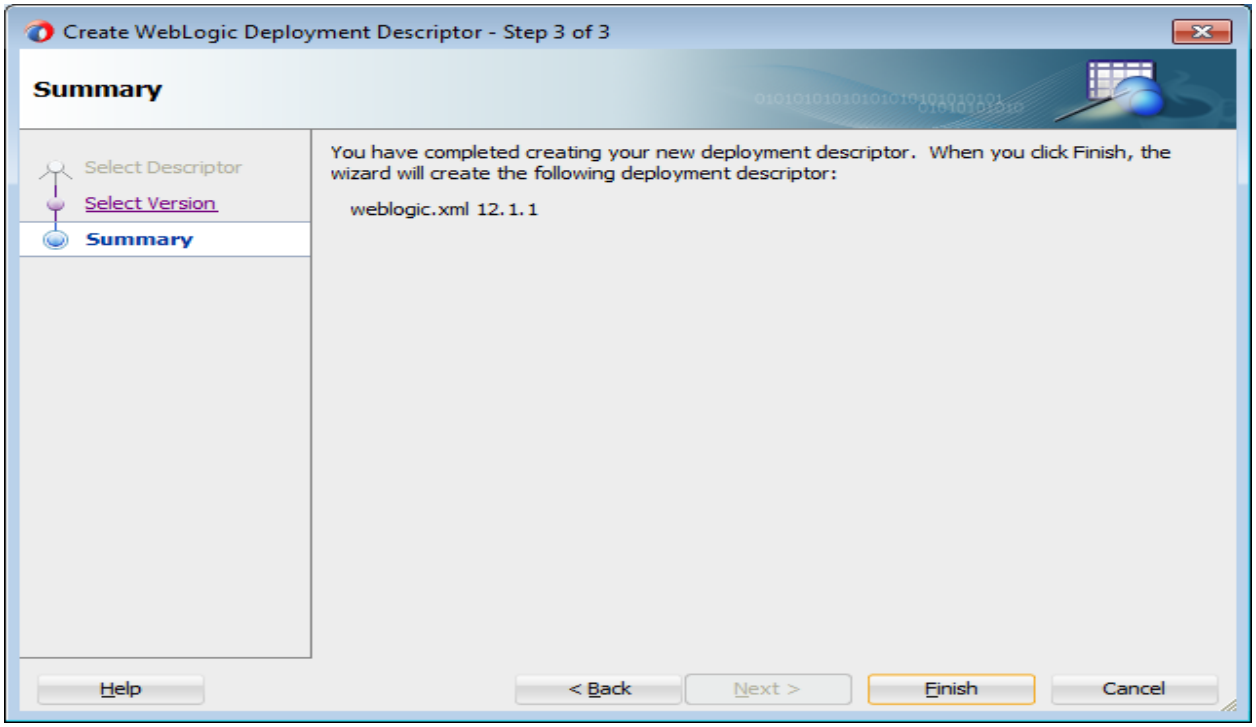
Click “OK”



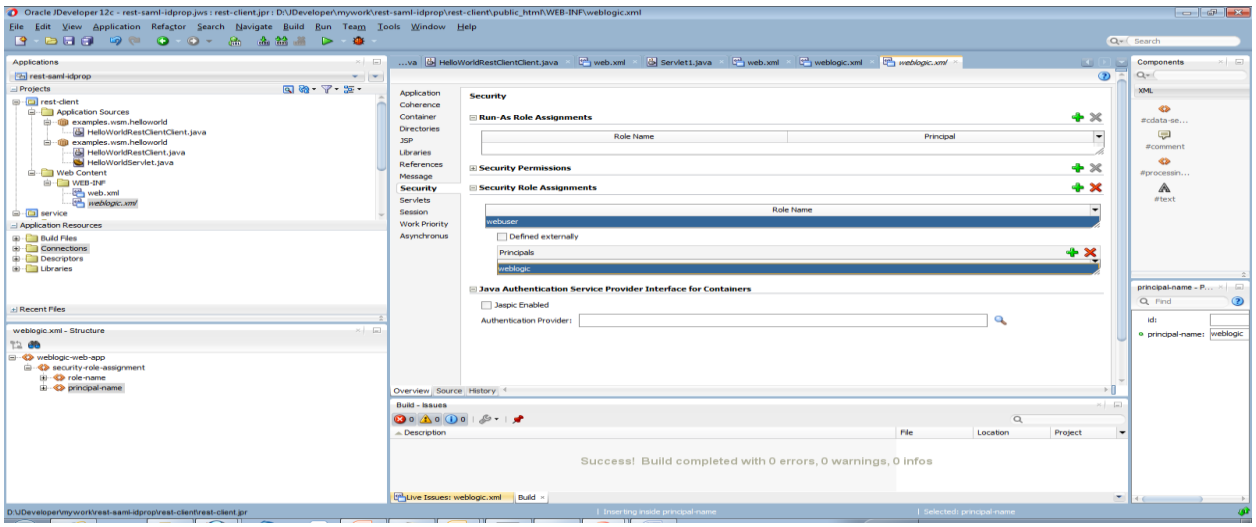
Click “Next”



Click “Next”



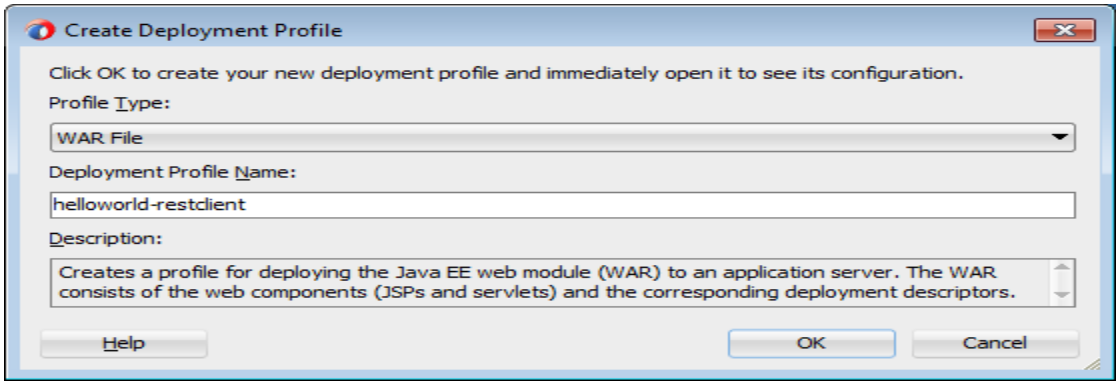
Click “Finish”



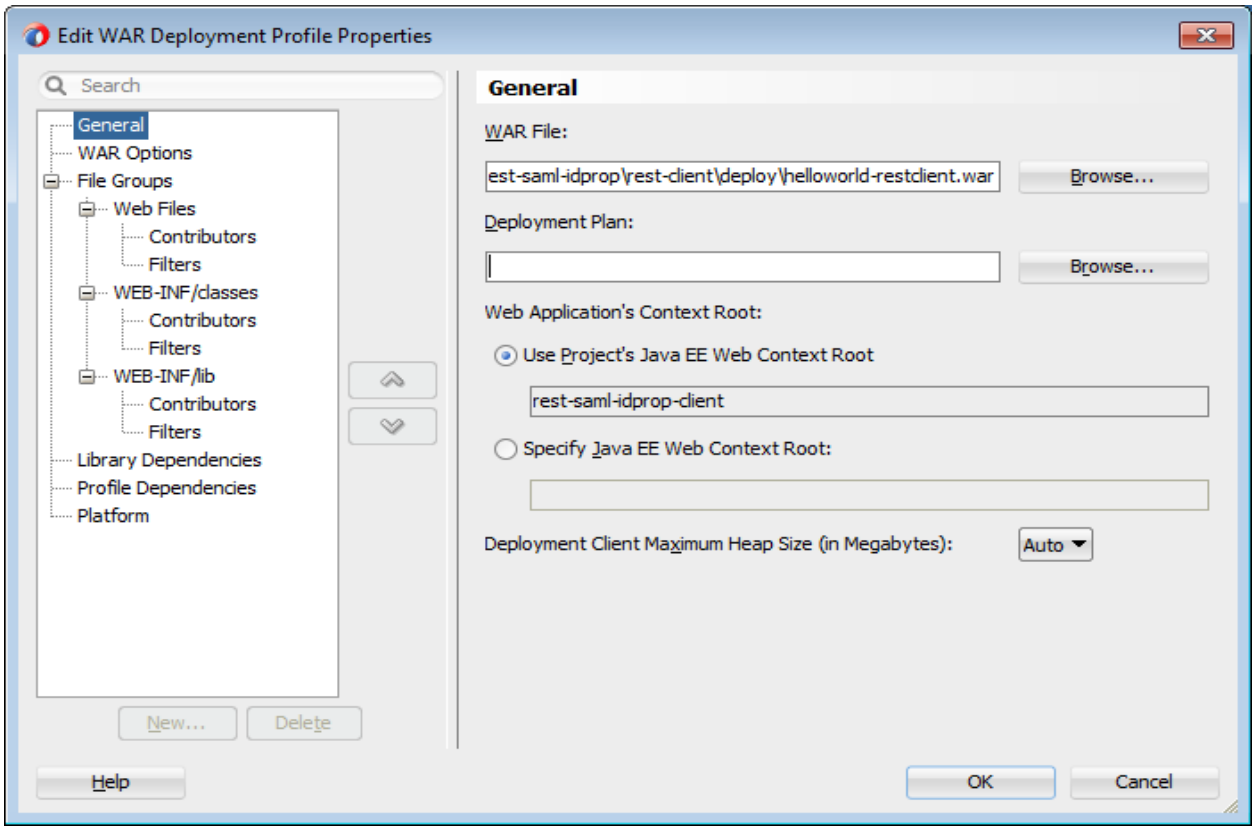
4.7.5 weblogic.xml for the HelloWorldServlet

```
<?xml version = '1.0' encoding = 'windows-1252'?>

<weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

                  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-web-app http://xmlns.oracle.com/weblogic/weblogic-web-app/1.4/weblogic-web-app.xsd"

                  xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app">

 <security-role-assignment>

   <role-name>webuser</role-name>

   <principal-name>weblogic</principal-name>

 </security-role-assignment>

</weblogic-web-app>
```

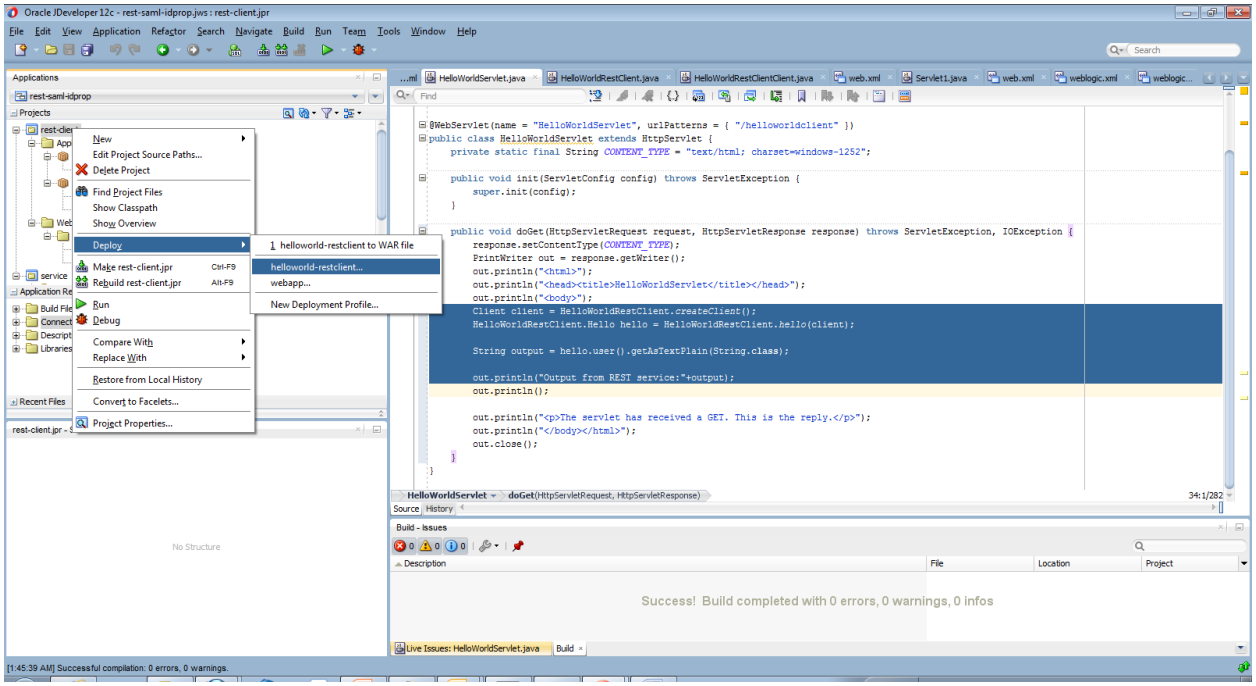
4.8 Create Deployment Profile and WAR for the Client

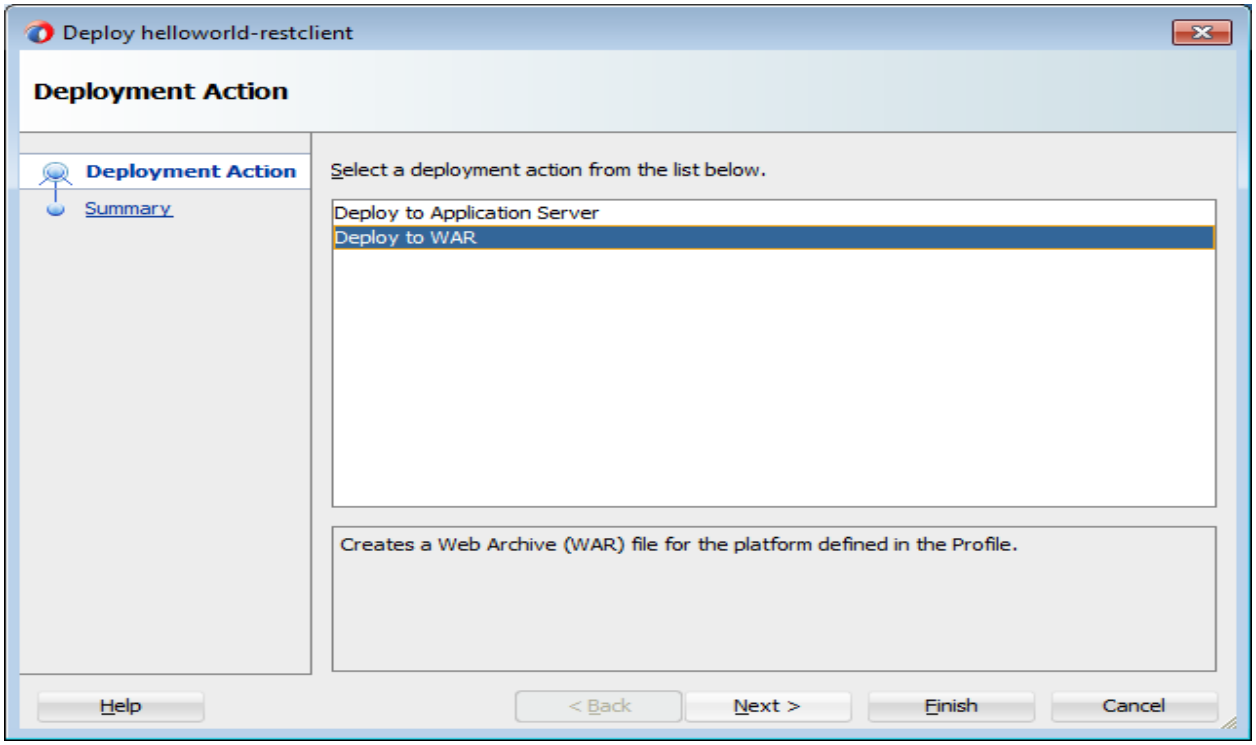


Click “OK”

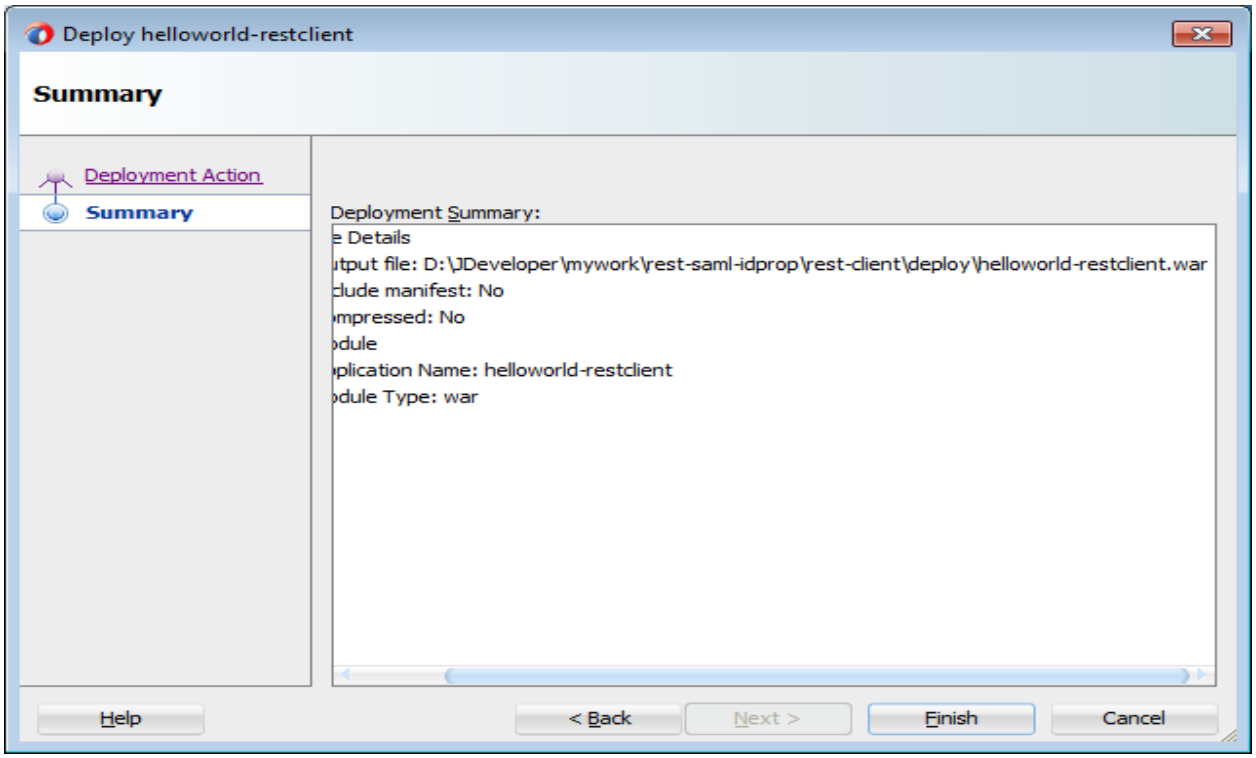

Edit WAR Deployment Profile Properties
Search
General
WAR Options
File Groups
Web Files
Contributors
Filters
WEB-INF/classes
Contributors
Filters
WEB-INF/lib
Contributors
Filters
Library Dependencies
Profile Dependencies
Platform
New...
Delete
General
WAR File:
est-saml-idprop\rest-client\deploy\helloworld-restclient.war
Browse...
Deployment Plan:
Browse...
Web Application's Context Root:
Use Project's Java EE Web Context Root
rest-saml-idprop-client
Specify Java EE Web Context Root:
Deployment Client Maximum Heap Size (in Megabytes):
Auto
Help
OK
Cancel

Click “OK”


Oracle JDeveloper 12c - rest-saml-idprop.jws : rest-client.jpr
File Edit View Application Refactor Search Navigate Build Run Team Tools Window Help
Applications
rest-saml-idprop
Projects
New
Edit Project Source Paths...
Delete Project
Find Project Files
Show Classpath
Show Overview
Deploy
Make rest-client.jpr Ctrl-F9
Rebuild rest-client.jpr Alt-F9
Run
Debug
Compare With
Replace With
Restore from Local History
Convert to Facelets...
Project Properties...
1 helloworld-restclient to WAR file
helloworld-restclient...
webapp...
New Deployment Profile...
HelloWorldServlet.java
HelloWorldRestClient.java
HelloWorldRestClientClient.java
web.xml
Servlet1.java
weblogic.xml
@WebServlet(name = "HelloWorldServlet", urlPatterns = { "/helloworldclient" })
public class HelloWorldServlet extends HttpServlet {
private static final String CONTENT_TYPE = "text/html; charset=windows-1252";
public void init(ServletConfig config) throws ServletException {
super.init(config);
}
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
response.setContentType(CONTENT_TYPE);
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>HelloWorldServlet</title></head>");
out.println("<body>");
Client client = HelloWorldRestClient.createClient();
HelloWorldRestClient.Hello hello = HelloWorldRestClient.hello(client);
String output = hello.user().getAsTextPlain(String.class);
out.println("Output from REST service:"+output);
out.println();
out.println(" The servlet has received a GET. This is the reply. ");
out.println("</body></html>");
out.close();
}
}
HelloWorldServlet doGet(HttpServletRequest, HttpServletResponse)
Source History
Build - Issues
Description File Location Project
Success! Build completed with 0 errors, 0 warnings, 0 infos
Live Issues: HelloWorldServlet.java
Build
[1:45:39 AM] Successful compilation: 0 errors, 0 warnings.



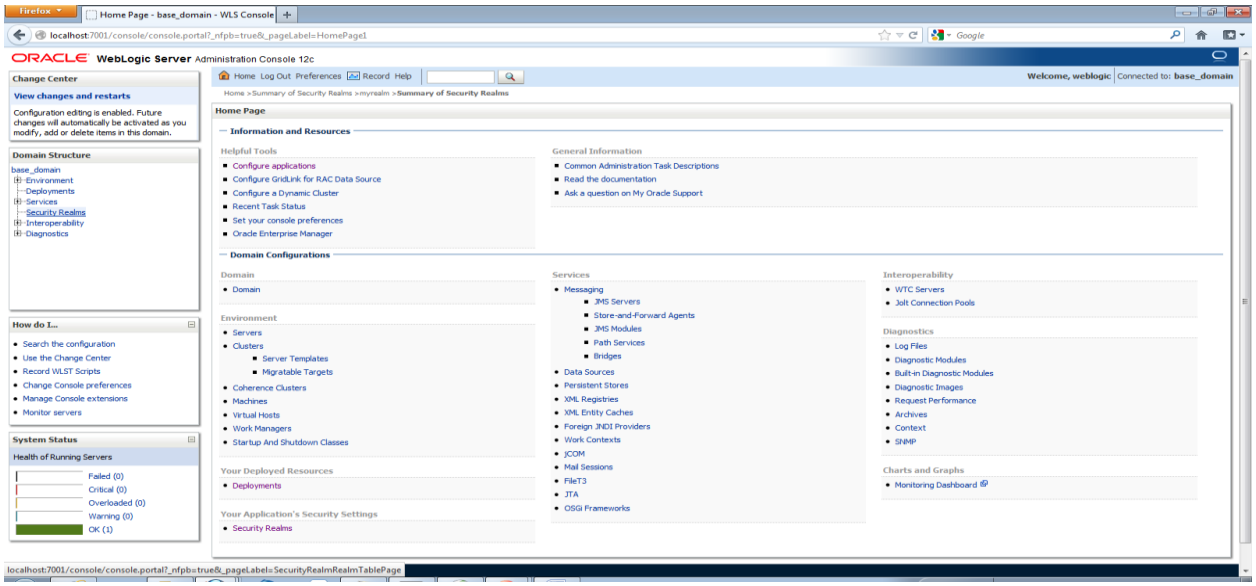
Click “Next”



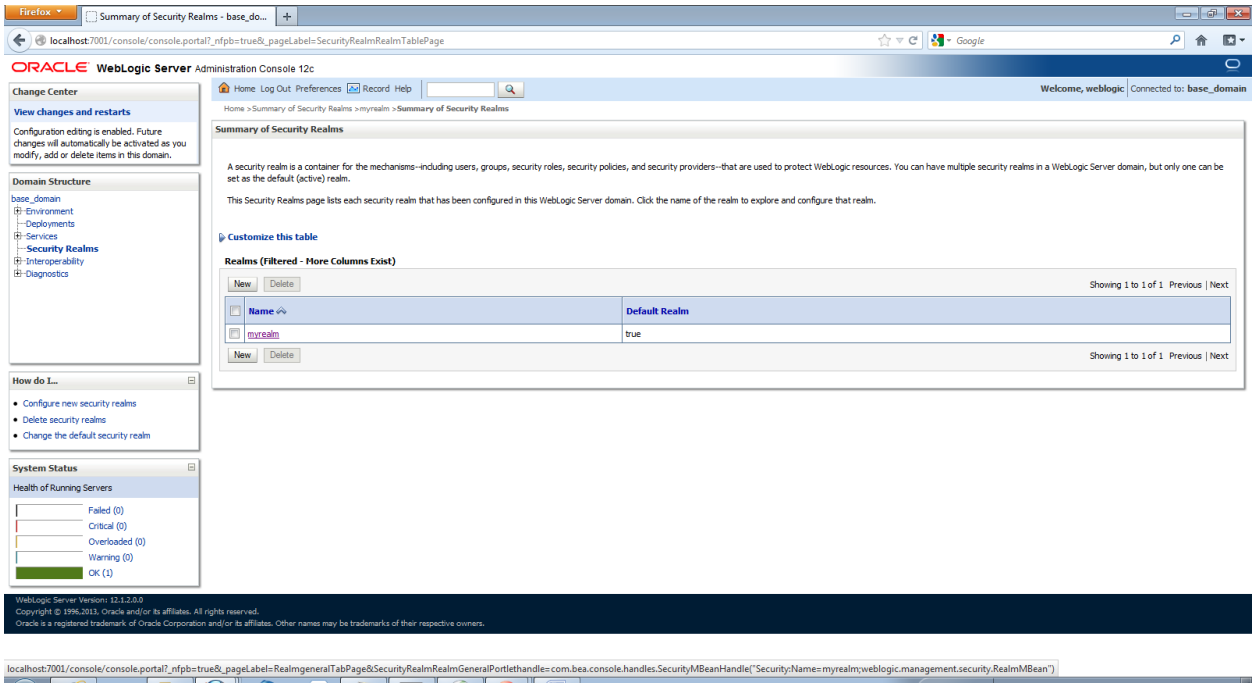
Click “Finish”

5 Create users in weblogic using WLS Console

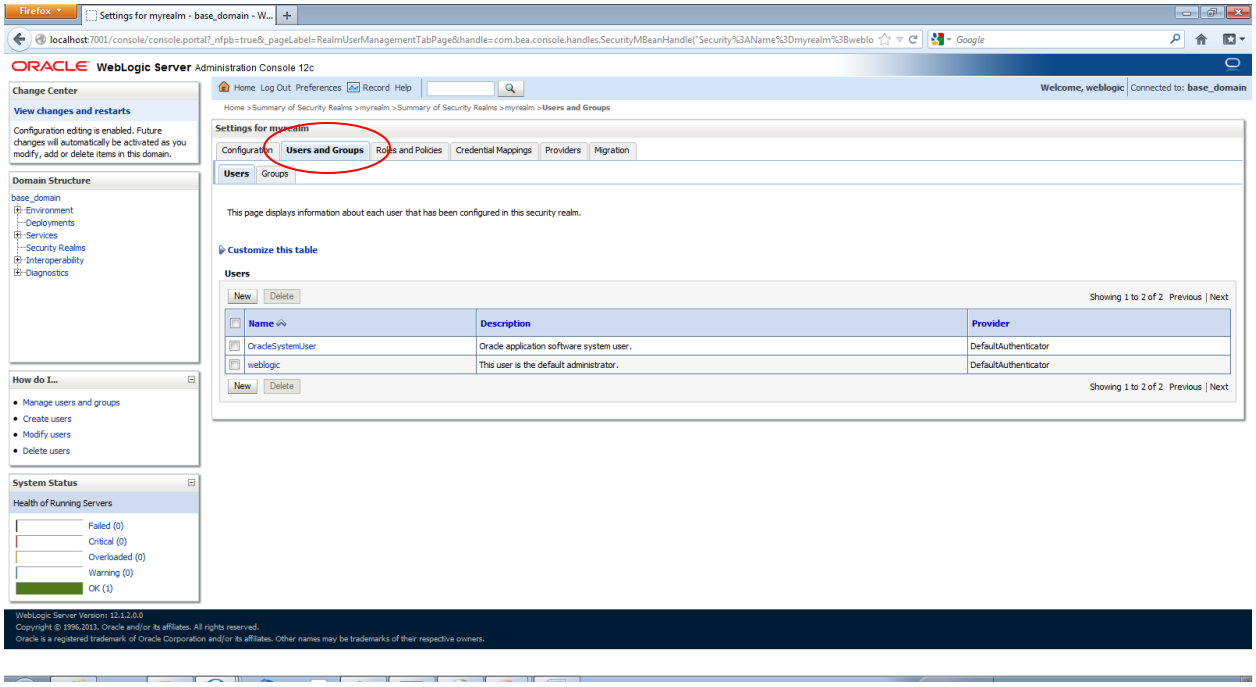
We will create some users for purposes of testing. These users will be created in embedded Ldap that ship with Weblogic. Log into WLS console as shown below.



Click on “Security Realms” on the LHS, this will open the Security Realms page on the RHS. By default Weblogic ships with a single realm called “myrealm”

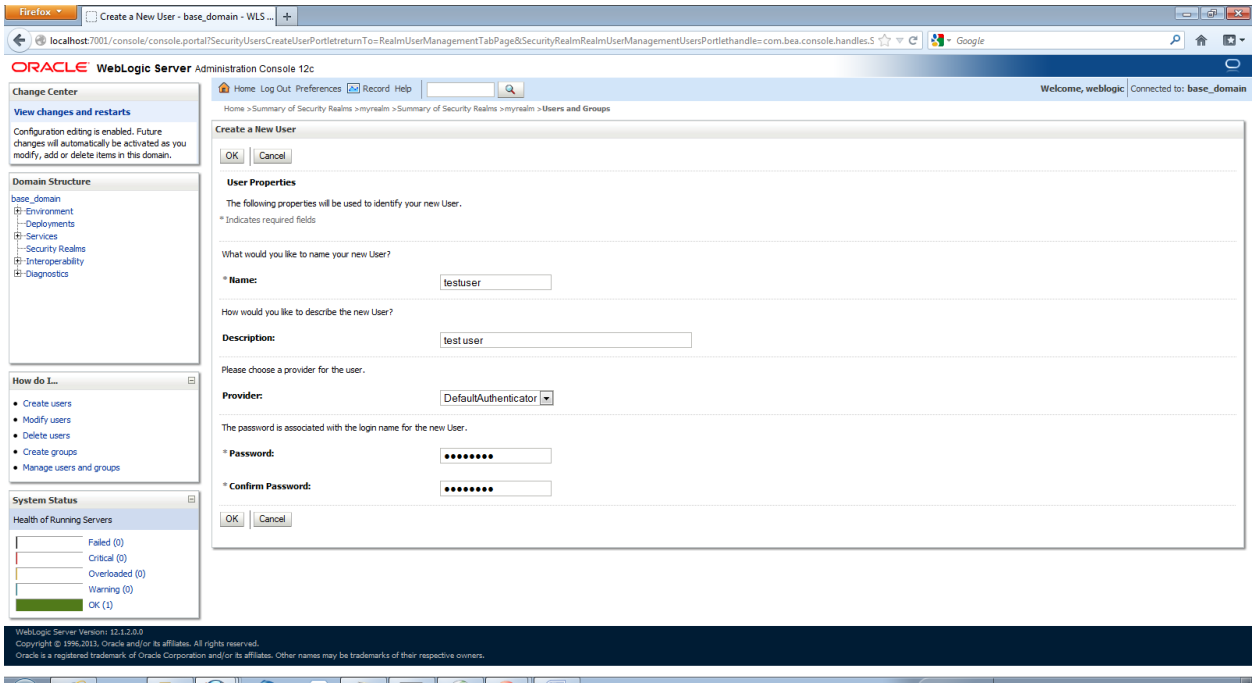


Click on “myrealm” in the above page. Click on the “Users and Groups” tab as shown below. This will list all the users/groups associated with that realm.

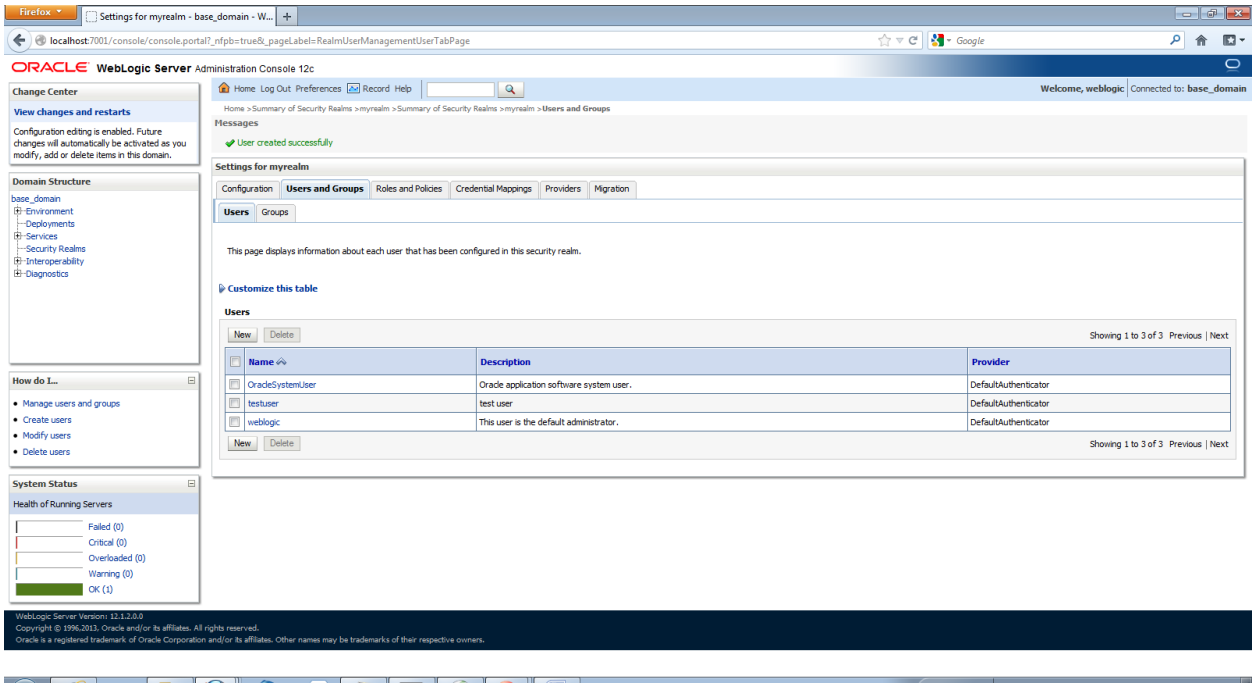


Click “New” to create a new user in the above page.

Create a user called “testuser”. Provide the password for “testuser” as shown below.

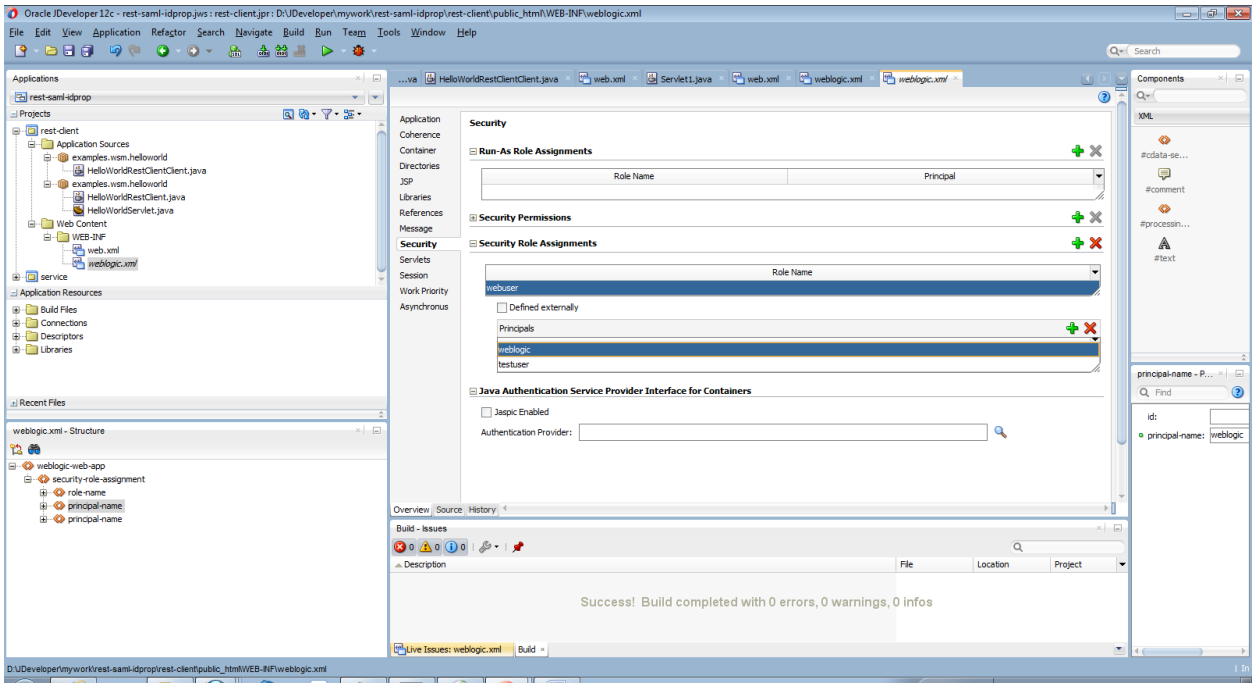


Click “OK” to complete creation of the new user “testuser”



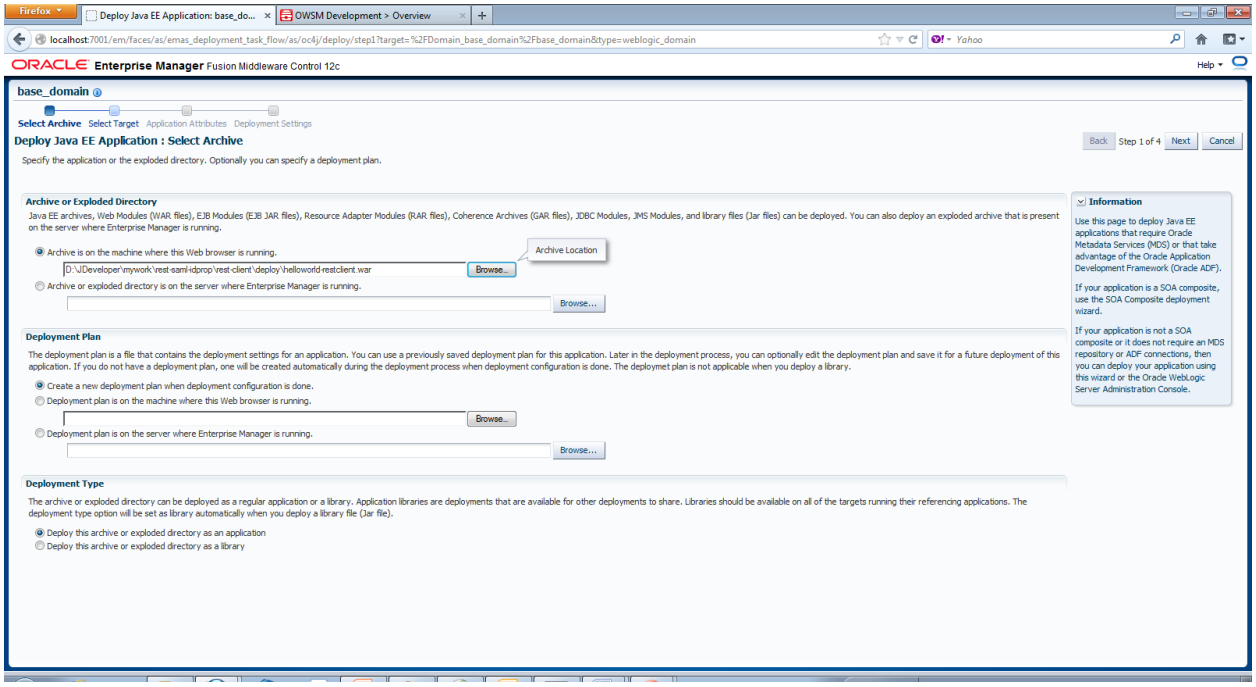
If user creation is successful – then testuser will be listed along with the prebuilt users that ship with Weblogic as shown in the screenshot above.

5.1 Mapping newly created "testuser" in weblogic.xml

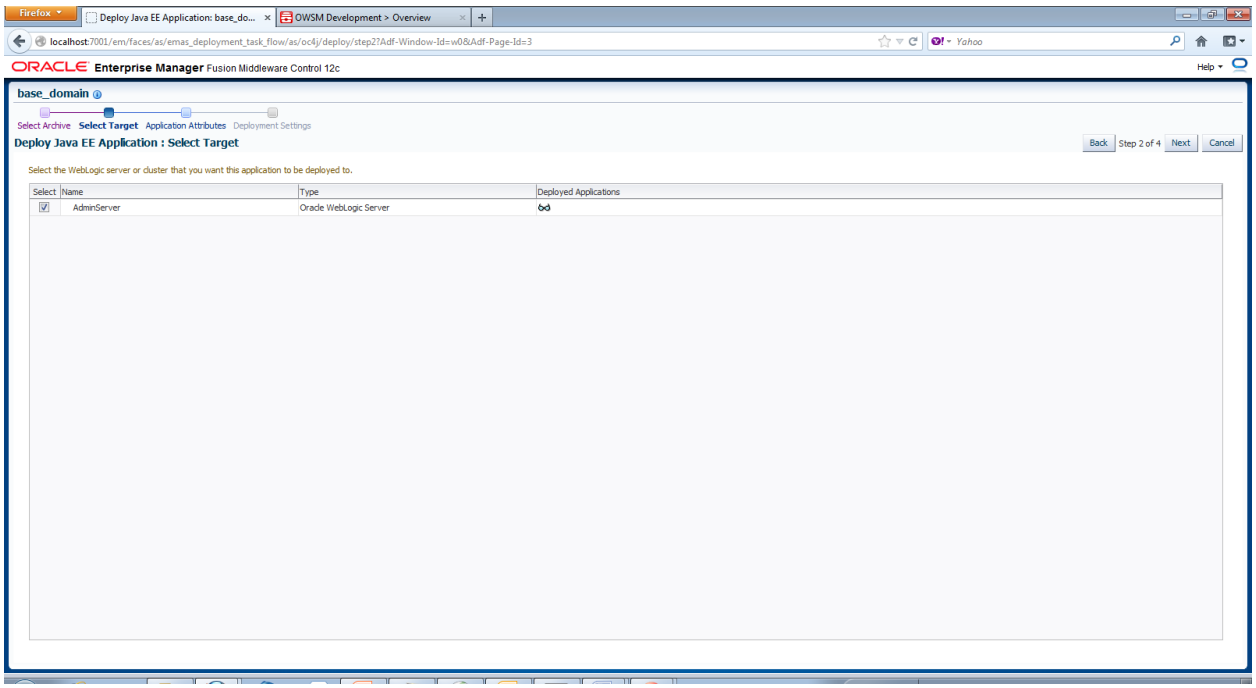


NOTE: Here I have mapped the role to individual Ldap users. It is better to map it to an Ldap Group.

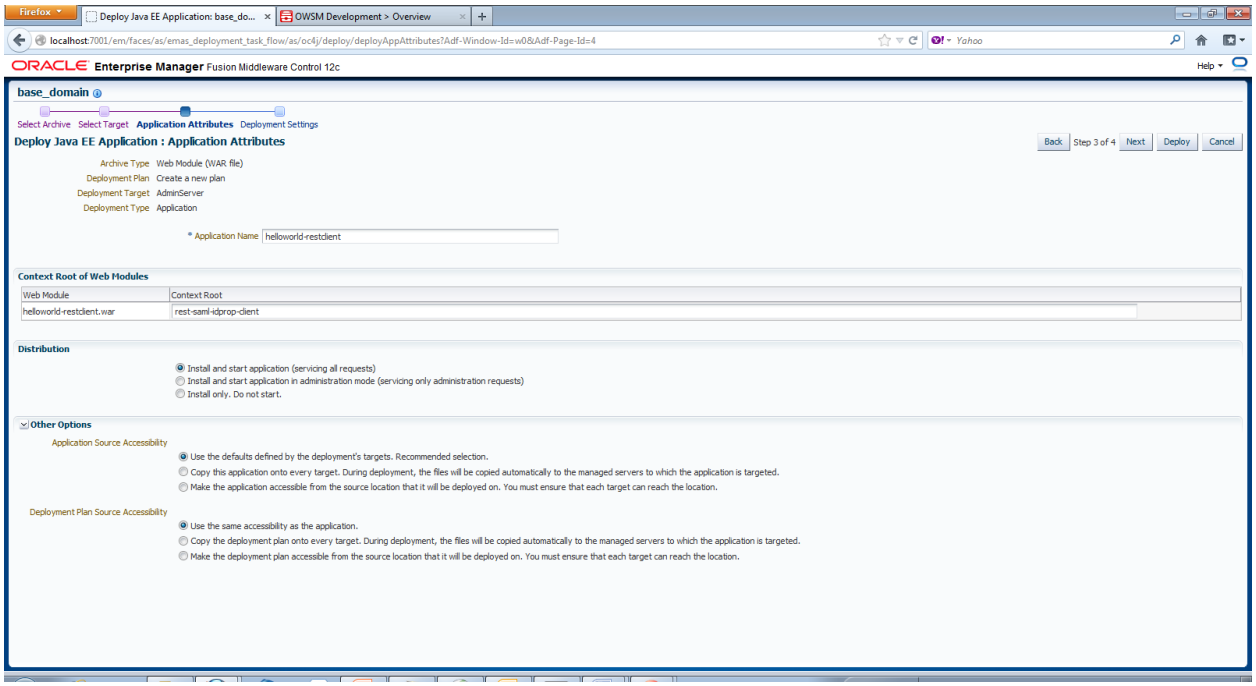
6 Deploy the client application to Weblogic using EM



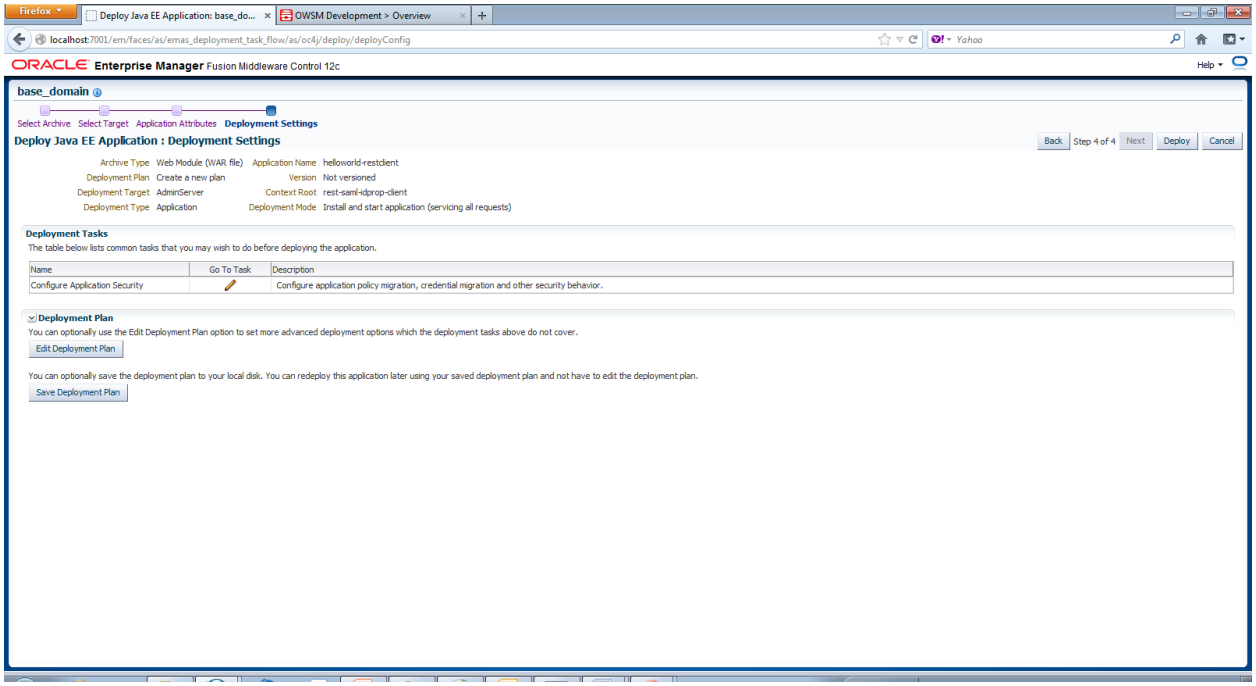
Click “Next”



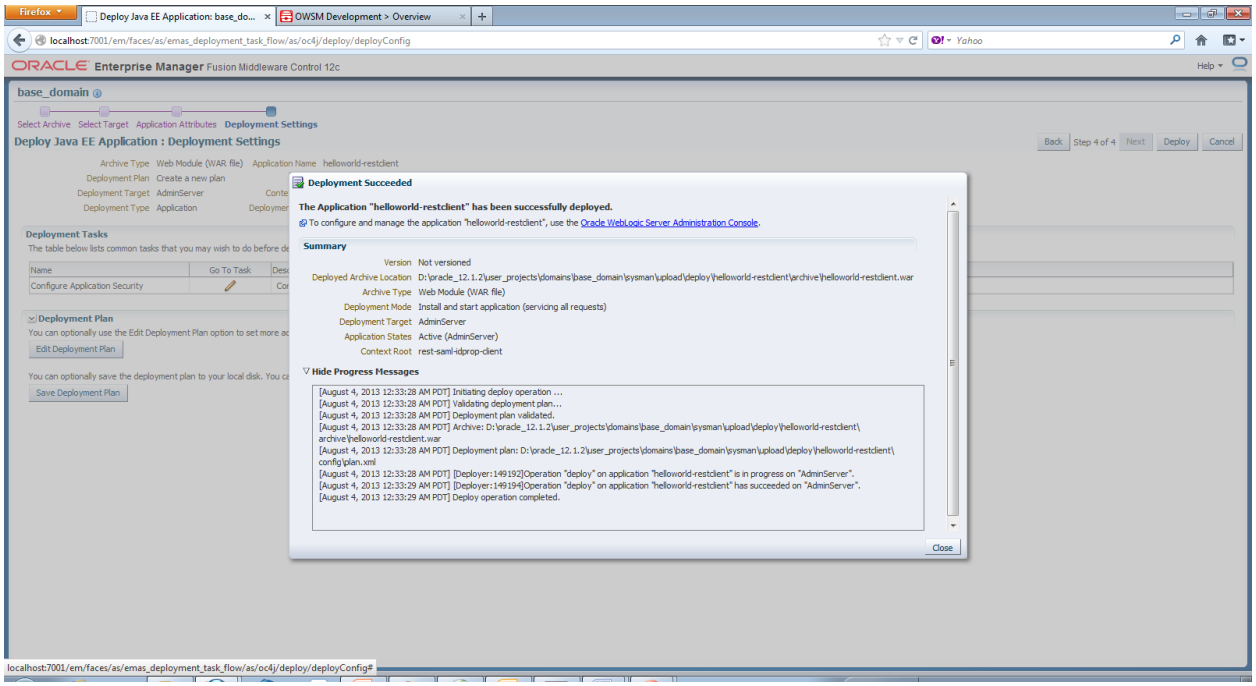
Click “Next”



Click “Next”



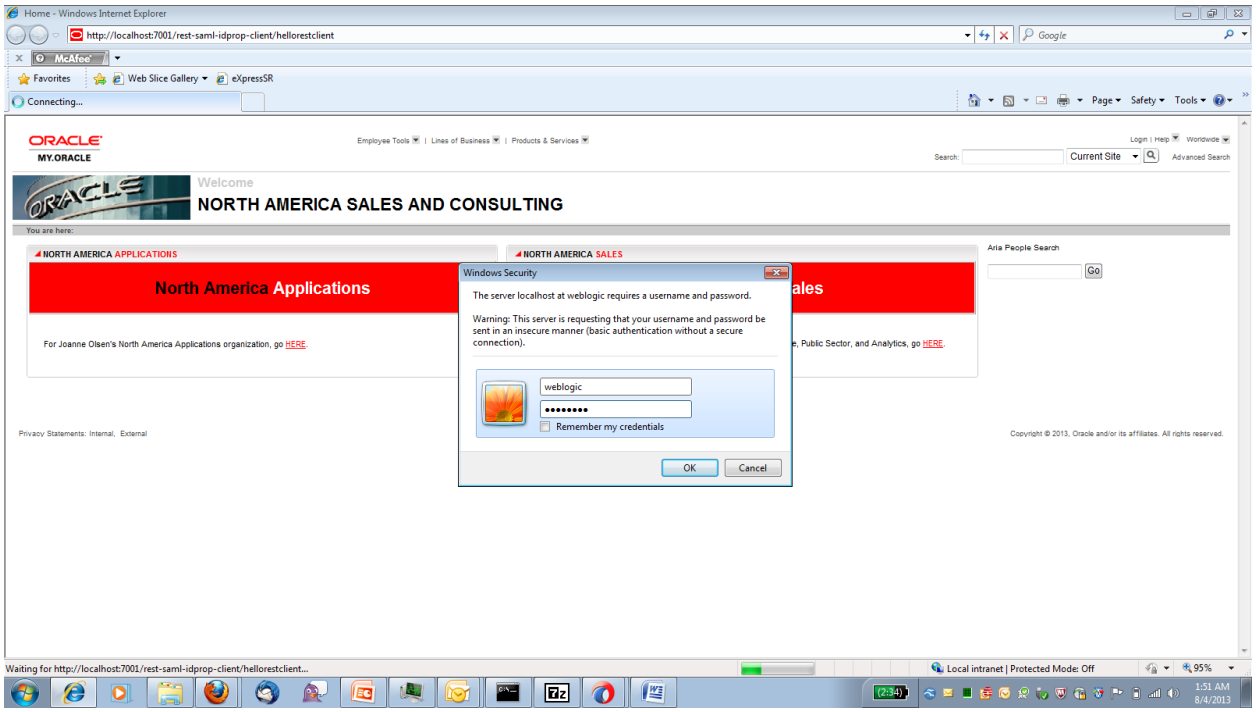
Click “Deploy”

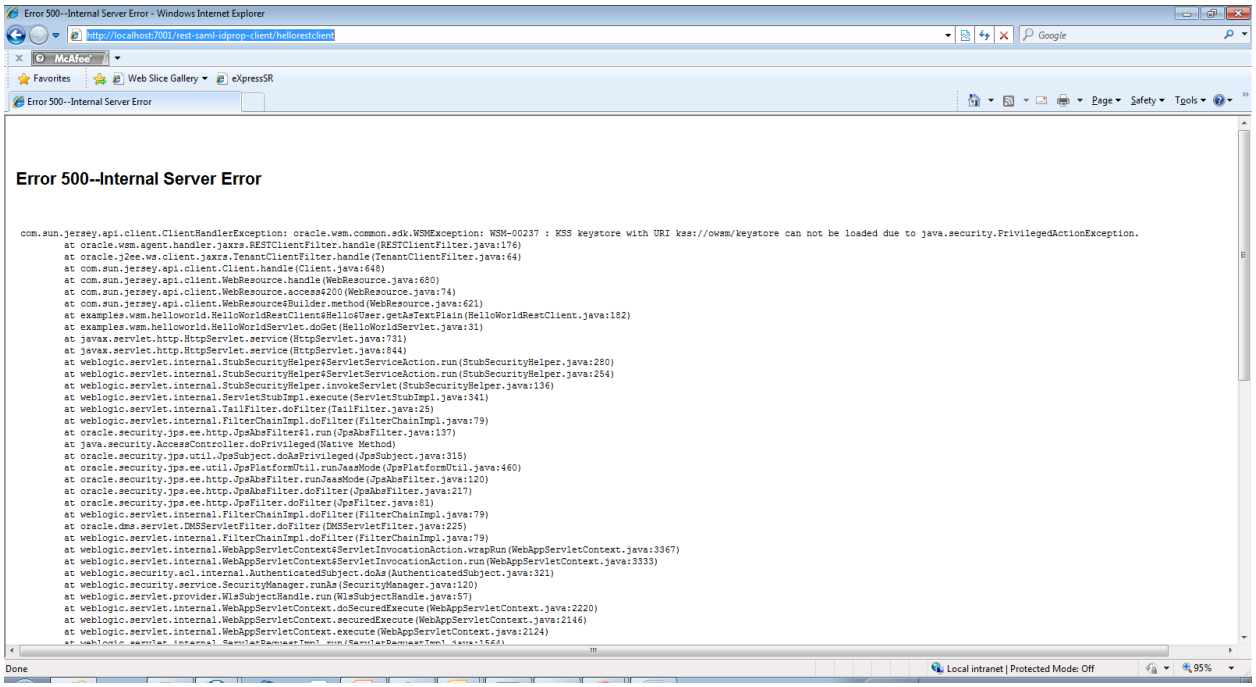


Click “Close”

6.1 Negative Testing

For the SAML token to work we need to set up the keystore. This is required as the SAML token is signed. Since we have not set up the keystore – testing will fail.





7 Set up Keystore

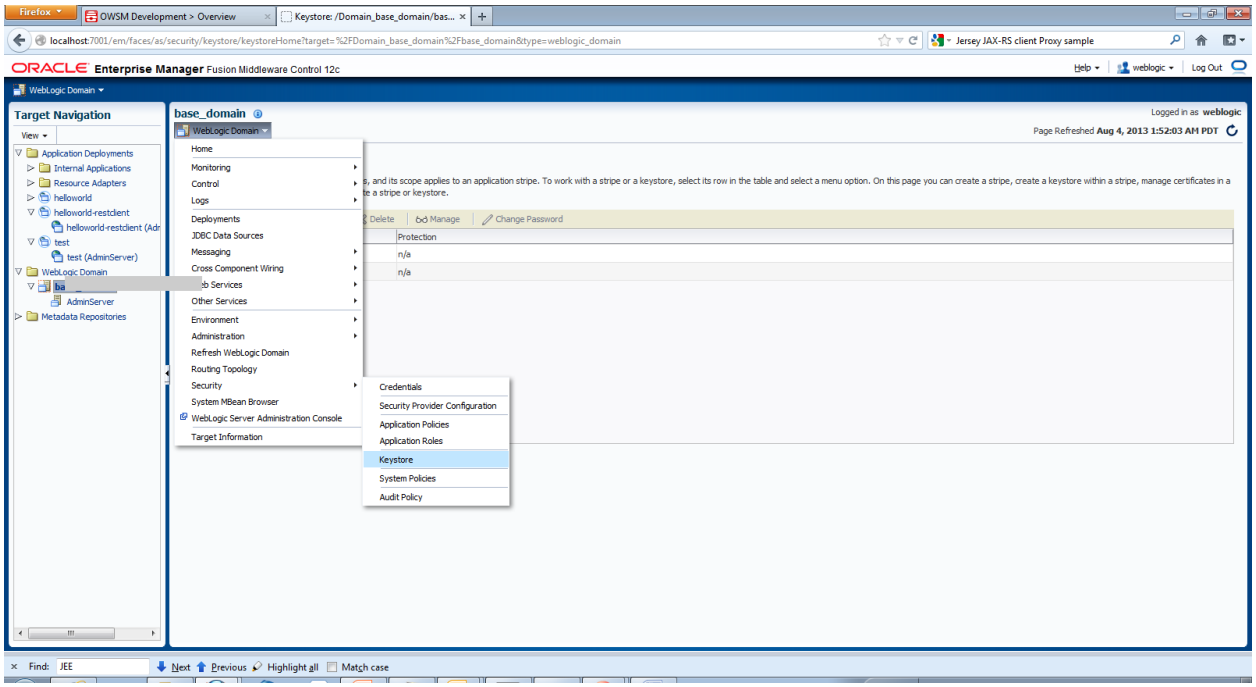
OWSM needs to signs the SAML bearer token, in order to do that it uses public key cryptography. Thus a keystore needs to be setup. In 12.1.2 the default keystore is KSS (Keystore Service) and this is different from 11g where the default keystore was JKS.

KSS is a service provided by Oracle Platform Security Services (OPSS) and there are a number of reasons for preferring KSS over JKS.

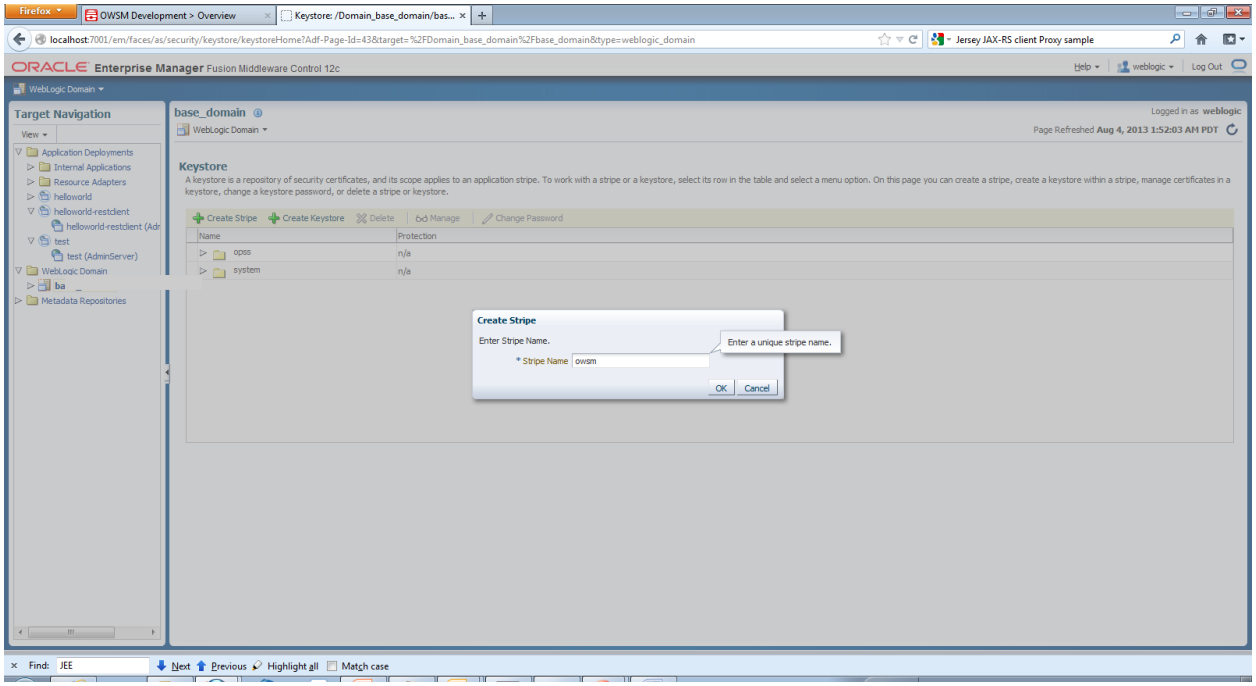
- KSS provides integrated tooling
  - EM and WLST support for creating/updating/import/exporting keys/certs
  - Internal CA for generating CA signed keys/certs
- KSS provides better lifecycle management.
  - Ability for multiple domains to share the same keystore is easier – since KSS provides centralized storage (ex: DB based storage)
  - Ability for segregation of keystores – ex: OWSM can have it's own keystore via the concept of "stripe" that exists in KSS.
  - Simplified management as it doesn't need passwords for accessing the private keys in the keystore.

7.1 Create "owsm" stripe and keystore

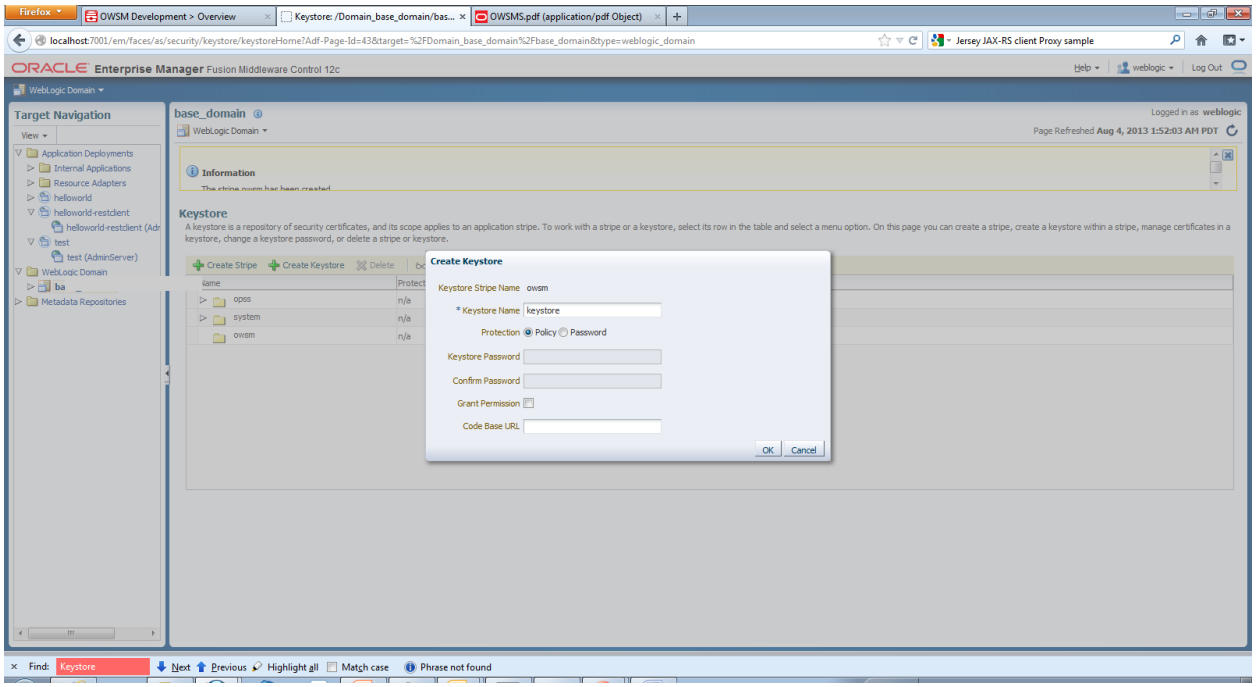
Click on the Weblogic domain on the LHS. Click on "Weblogic Domain->Security->Keystore" as shown in the screenshot below.



Click on "Create Stripe" as shown in the screenshot below.

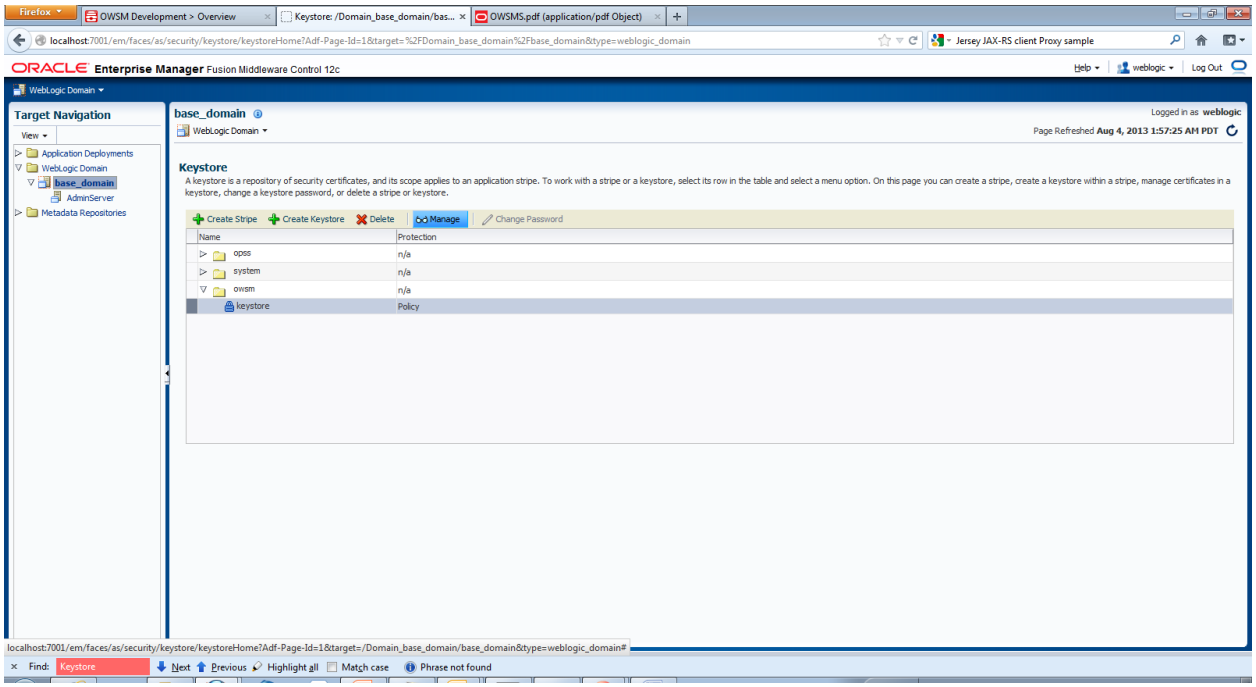


Create a Keystore under the “owsm” stripe by clicking on “Create Keystore” button. The keystore can be policy protected or password protected. Password protected will make it behave similar to JKS. I suggest you retain the default which is policy based and uncheck the “Grant Permission” checkbox as shown in the screenshot below.

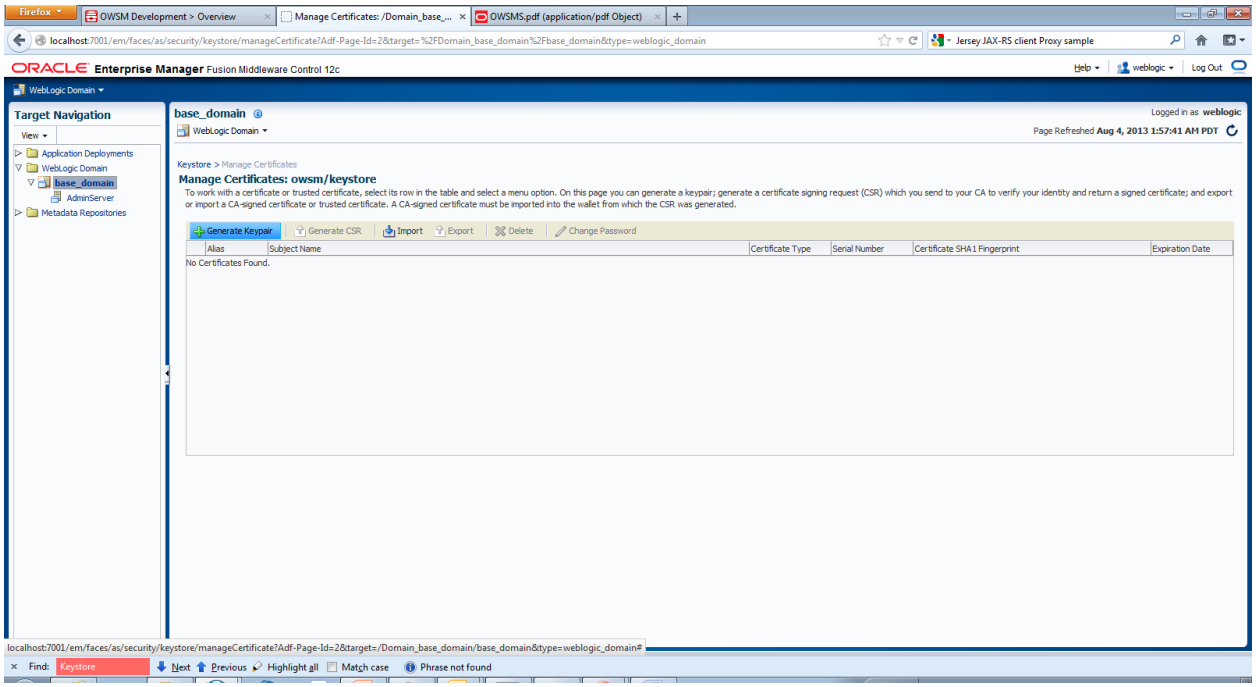


7.2 Generate keypair

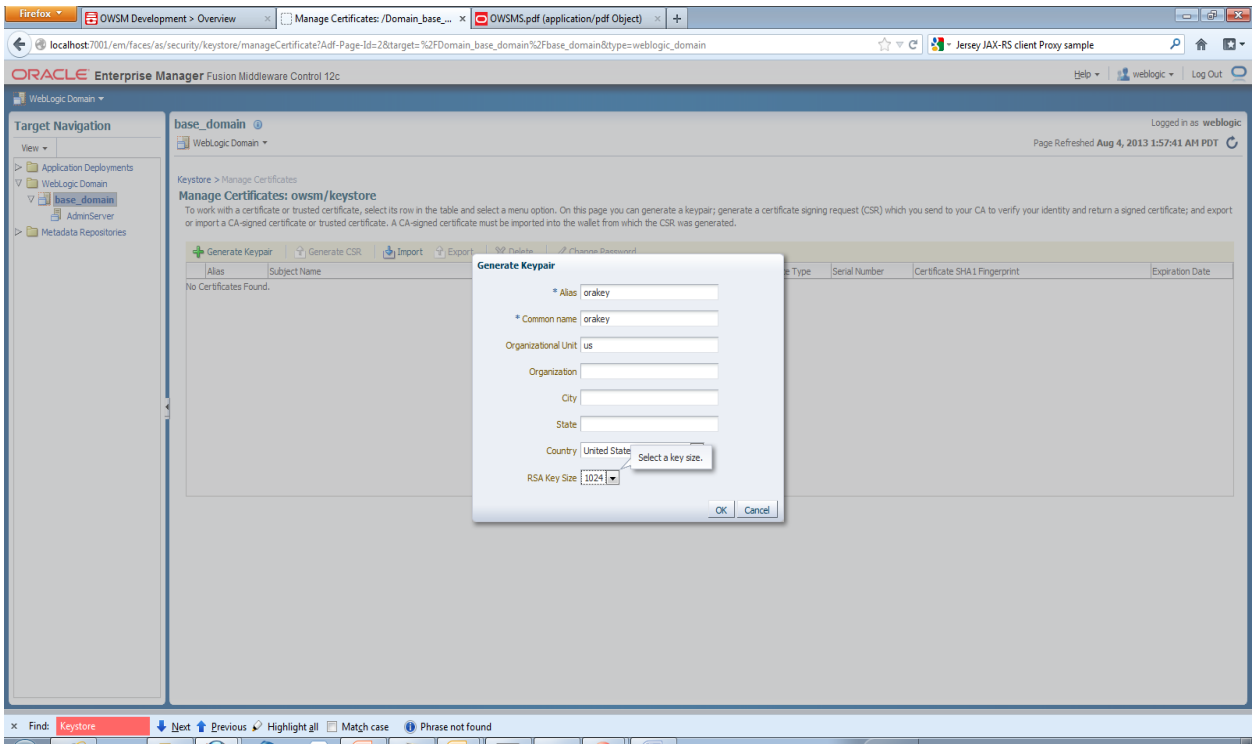
The next step is to generate a keypair. You can do this by selecting the “owsm->keystore” and clicking on “Manage” as shown in the screenshow below.

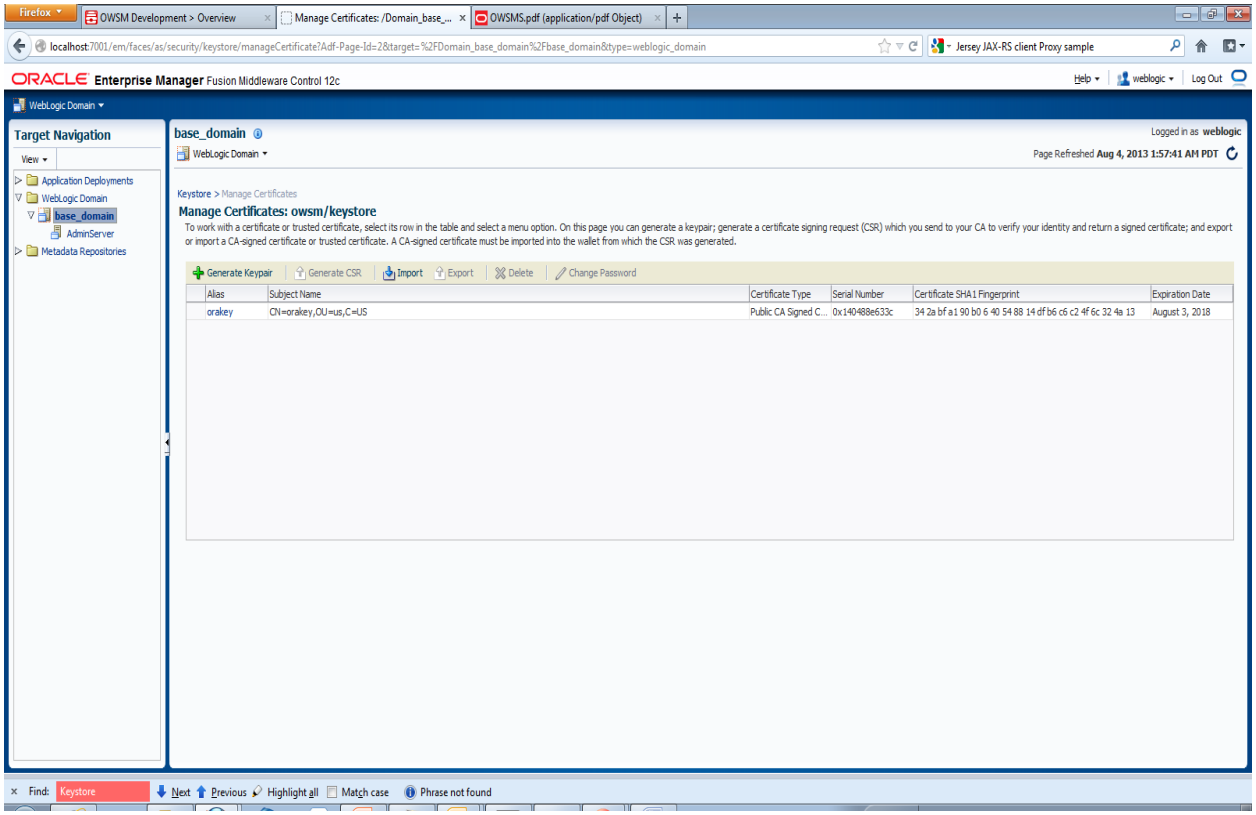


Generate a keypair by clicking on “Generate Keypair”.



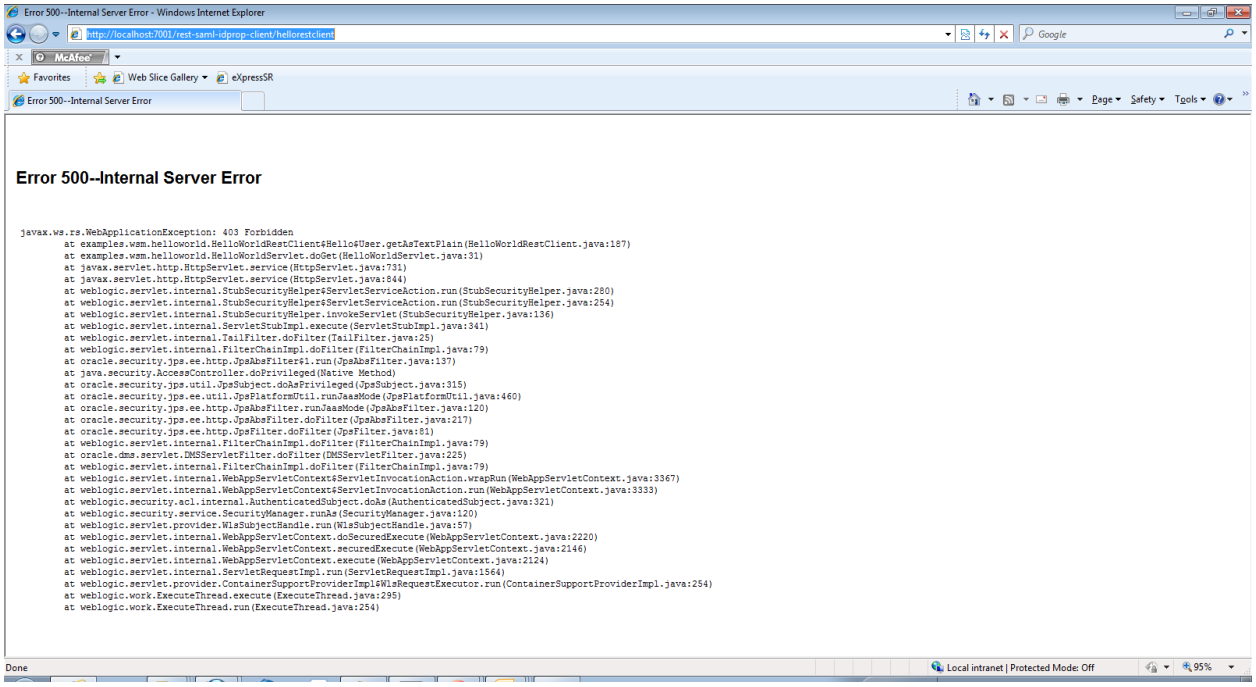
Use the default alias “orakey”. If you want to use a different alias – then additional steps will be required.


Firefox
OWSM Development > Overview
Manage Certificates: /Domain_base_...
OWSMS.pdf (application/pdf Object)
localhost:7001/em/faces/as/security/keystore/manageCertificate?Adf-Page-Id=2&target=%2FDomain_base_domain%2Fbase_domain&type=weblogic_domain
Jersey JAX-RS client Proxy sample
ORACLE Enterprise Manager Fusion Middleware Control 12c
Help
weblogic
Log Out
WebLogic Domain
Target Navigation
View
Application Deployments
WebLogic Domain
base_domain
AdminServer
Metadata Repositories
base_domain
WebLogic Domain
Logged in as weblogic
Page Refreshed Aug 4, 2013 1:57:41 AM PDT
Keystore > Manage Certificates
Manage Certificates: owsm/keystore
To work with a certificate or trusted certificate, select its row in the table and select a menu option. On this page you can generate a keypair; generate a certificate signing request (CSR) which you send to your CA to verify your identity and return a signed certificate; and export or import a CA-signed certificate or trusted certificate. A CA-signed certificate must be imported into the wallet from which the CSR was generated.
Generate Keypair
Generate CSR
Import
Export
Delete
Change Password
Alias
Subject Name
Certificate Type
Serial Number
Certificate SHA1 Fingerprint
Expiration Date
No Certificates Found.
Generate Keypair
* Alias orakey
* Common name orakey
Organizational Unit us
Organization
City
State
Country United State
Select a key size.
RSA Key Size 1024
OK
Cancel
Find: Keystore
Next
Previous
Highlight all
Match case
Phrase not found


Firefox
OWSM Development > Overview
Manage Certificates: /Domain_base_...
OWSMS.pdf (application/pdf Object)
localhost:7001/em/faces/as/security/keystore/manageCertificate?Adf-Page-Id=2&target=%2FDomain_base_domain%2Fbase_domain&type=weblogic_domain
Jersey JAX-RS client Proxy sample
ORACLE Enterprise Manager Fusion Middleware Control 12c
Help
weblogic
Log Out
WebLogic Domain
Target Navigation
View
Application Deployments
WebLogic Domain
base_domain
AdminServer
Metadata Repositories
base_domain
WebLogic Domain
Logged in as weblogic
Page Refreshed Aug 4, 2013 1:57:41 AM PDT
Keystore > Manage Certificates
Manage Certificates: owsm/keystore
To work with a certificate or trusted certificate, select its row in the table and select a menu option. On this page you can generate a keypair; generate a certificate signing request (CSR) which you send to your CA to verify your identity and return a signed certificate; and export or import a CA-signed certificate or trusted certificate. A CA-signed certificate must be imported into the wallet from which the CSR was generated.
Generate Keypair
Generate CSR
Import
Export
Delete
Change Password
Alias
Subject Name
Certificate Type
Serial Number
Certificate SHA1 Fingerprint
Expiration Date
orakey
CN=orakey,OU=us,C=US
Public CA Signed C...
0x140488e633c
34 2a bf a1 90 b0 6 40 54 88 14 df b6 c6 c2 4f 6c 32 4a 13
August 3, 2018
Find: Keystore
Next
Previous
Highlight all
Match case
Phrase not found

7.3 Negative Testing

Even though we created the keystore – if you test – you will see it results in a failure as shown below.



7.4 Import the "democa" CA certificate into "owsm" stripe

The reason it fails even after the keystore is set up is that the keypair we generated was signed by the Internal CA that ships with KSS and the validation of the cert on the service side will fail as OWSM Agent is unable to validate the Certificate path for the signing certificate.

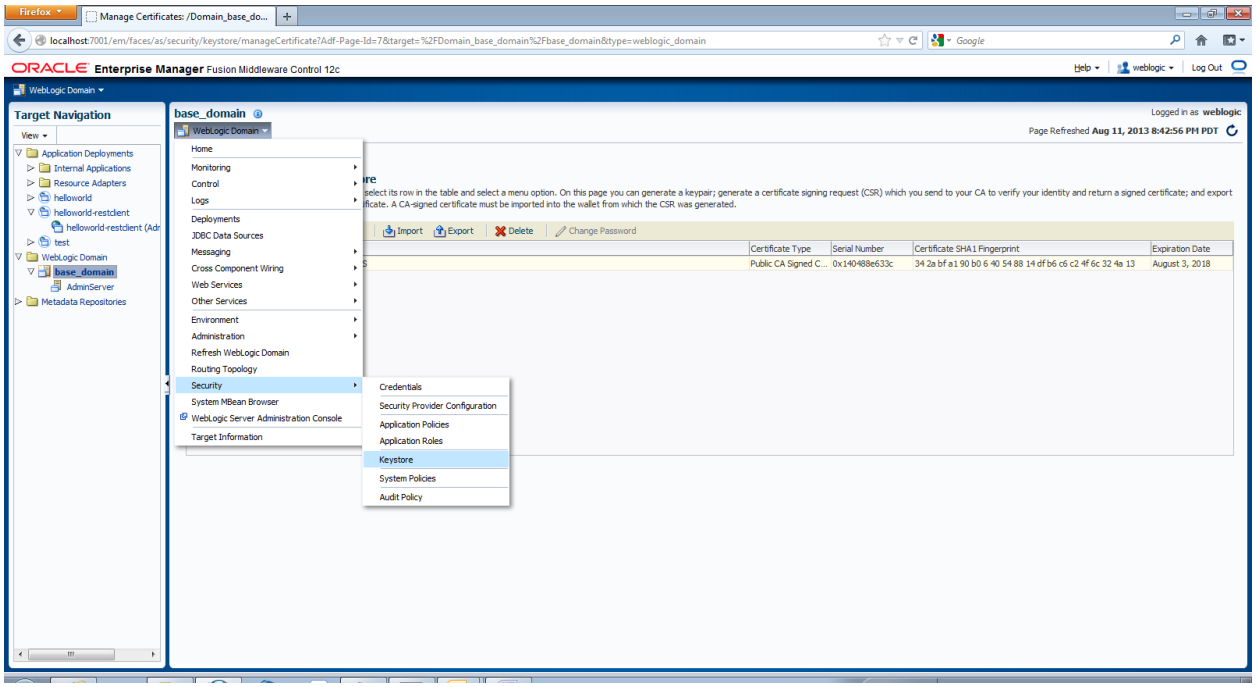
Note: You will not run into this issue if you use self-signed certificates. Since we have used CA signed (albeit an Internal CA signed certificate) – you need to import the CA cert into the OWSM keystore.

So you need to import the public certificate of the CA into the "owsm" stripe. Importing CA cert is a three step process:

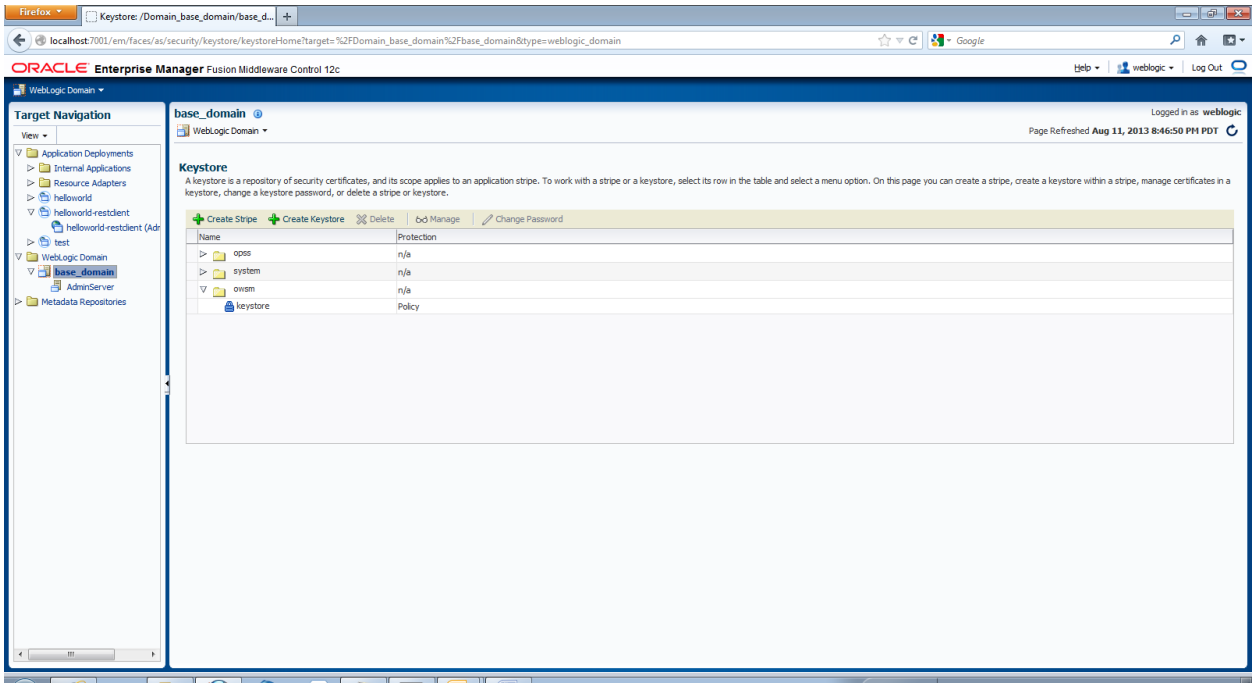
a) Identify the Certificate Authority (CA)
b) Export the CA cert from the "system" stripe in KSS.
c) Import the CA cert into the "owsm" stripe in KSS

7.4.1 Identify the Issuer of the "orakey" key

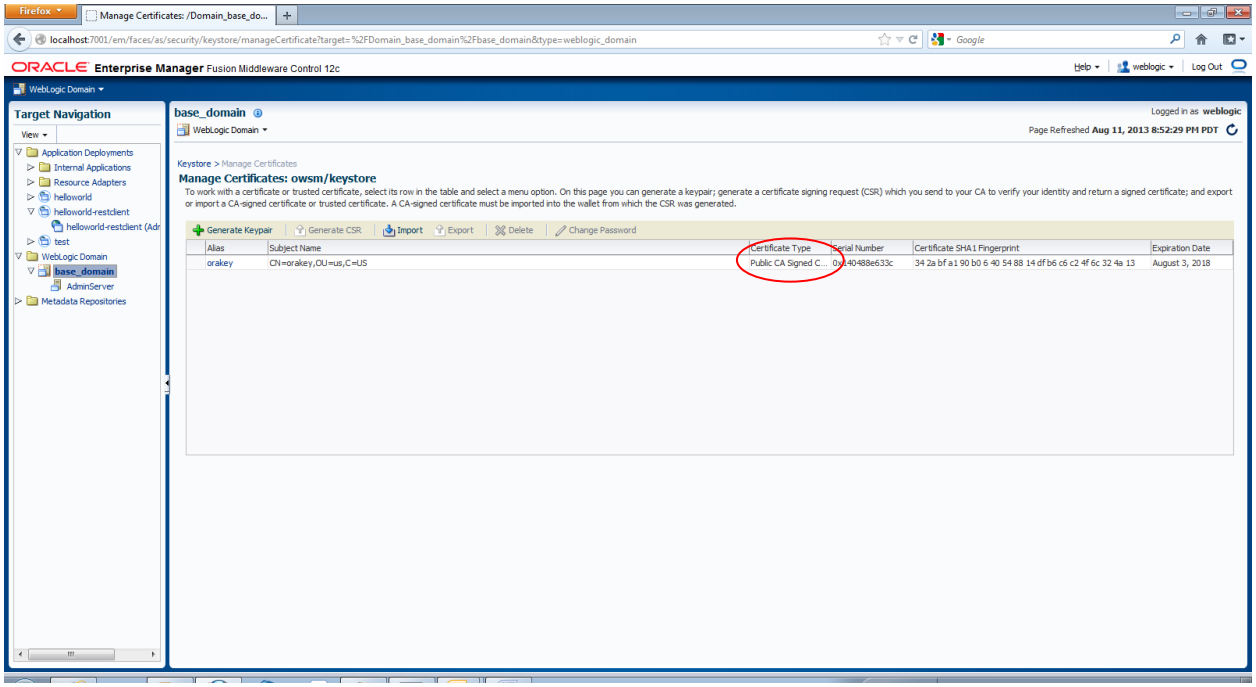
Navigate to the Keystore service in EM as shown in the screenshot below. "Weblogic Domain->Security->Keystore"



Expand the "owsm" stripe as shown in the screenshot below.

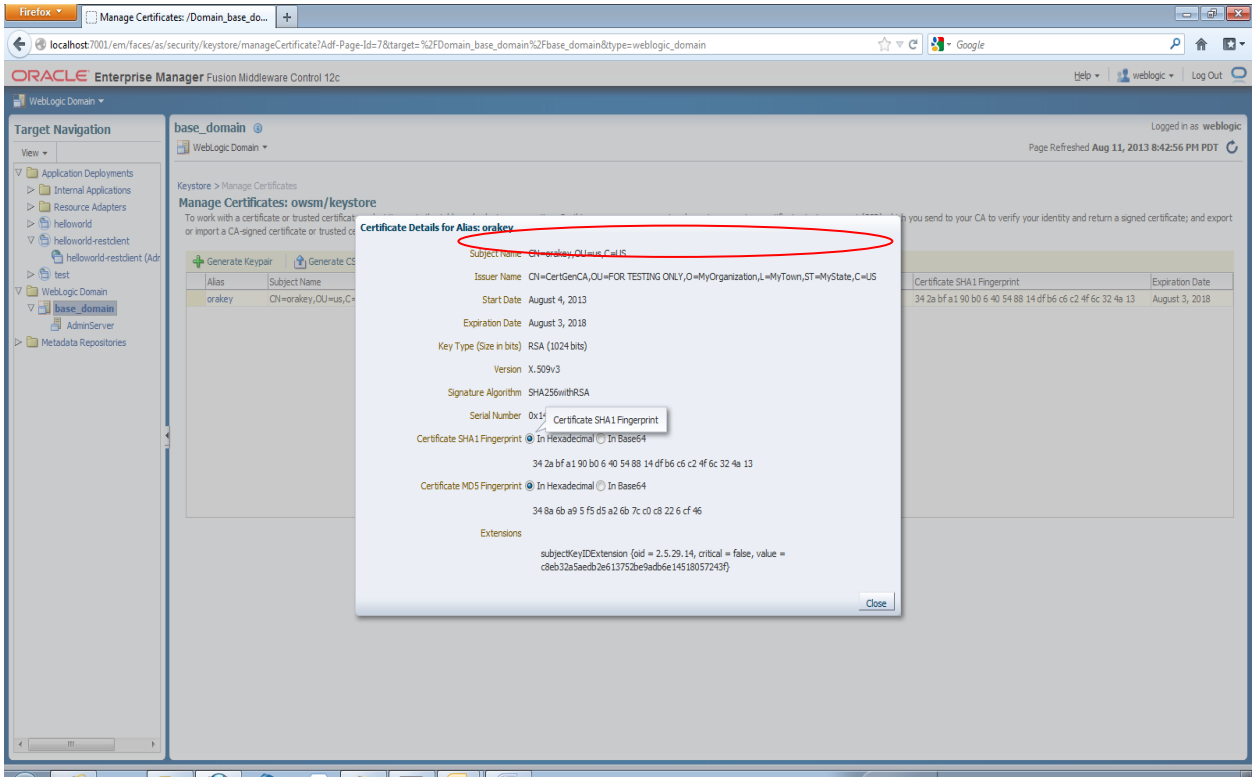


Select "keystore" and click on "Manage". This will list all the keys/certs in the keystore as shown in the screenshot below.



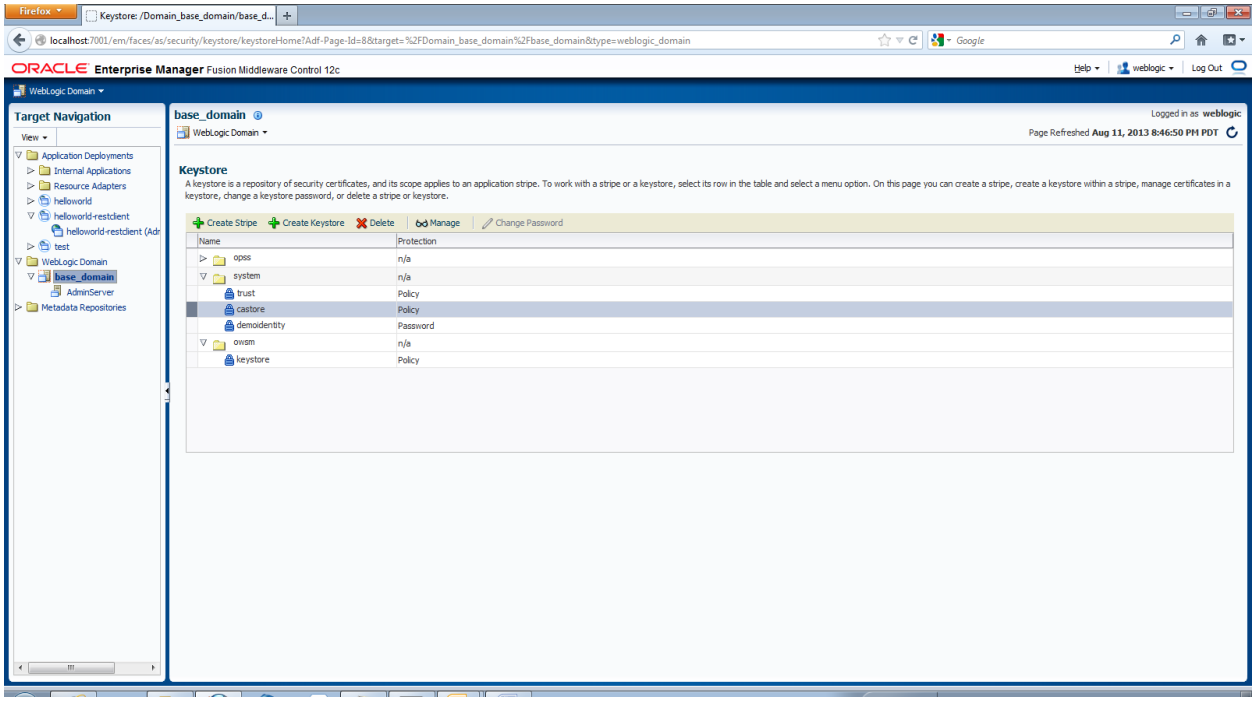
Click on “orakey” for details.

You will notice that the key pair that was generated was CA signed as shown in the screenshot below.

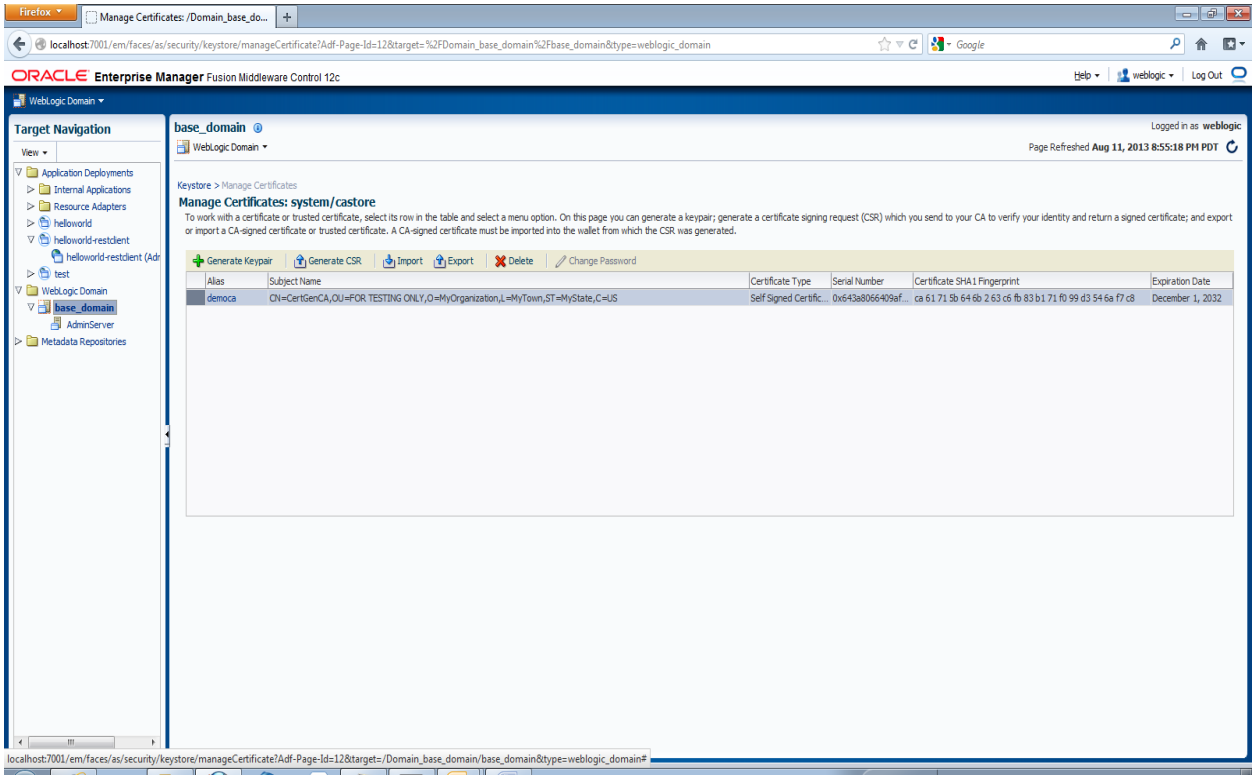


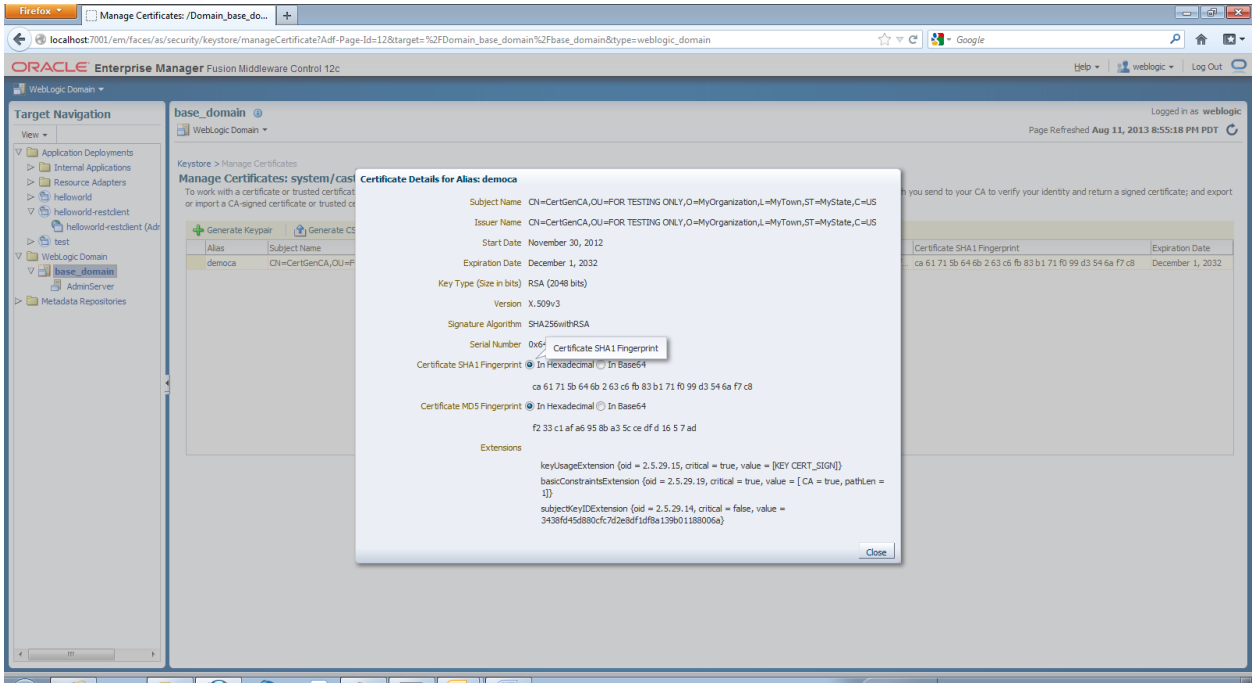
7.4.2 Export "democa" CA cert from "system" stripe

Expand the "system" stripe. Select "castore" and click on "Manage".

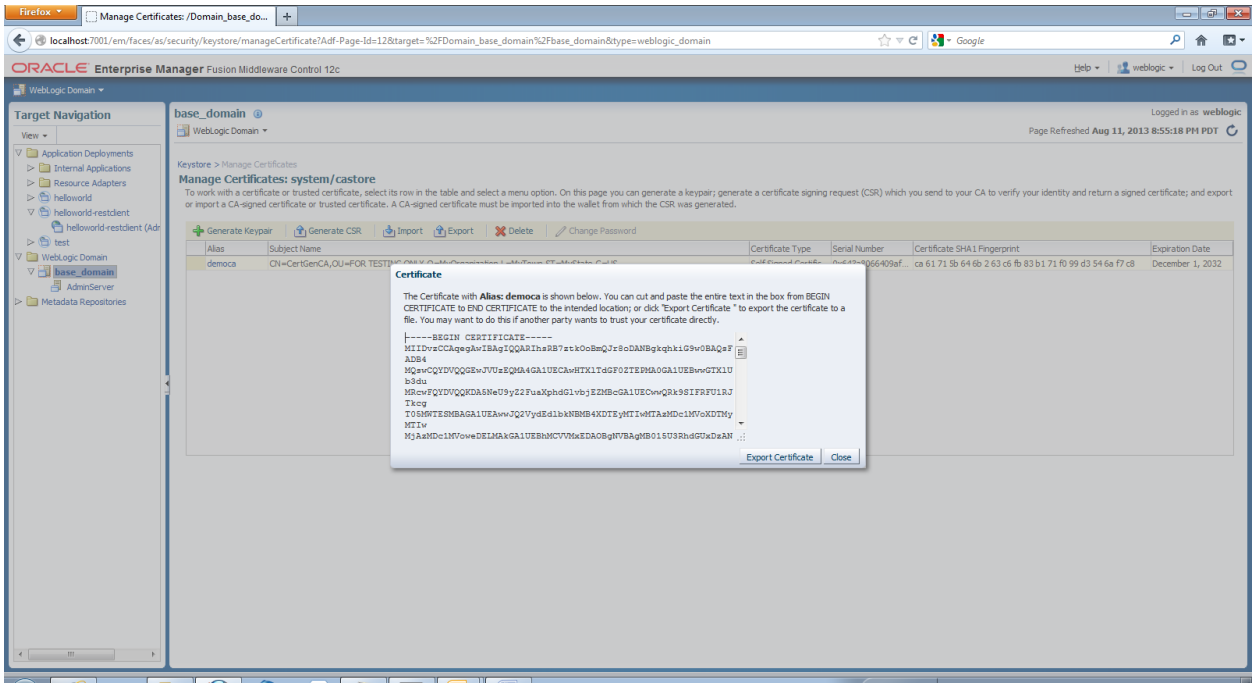


You will see a "democa" alias.



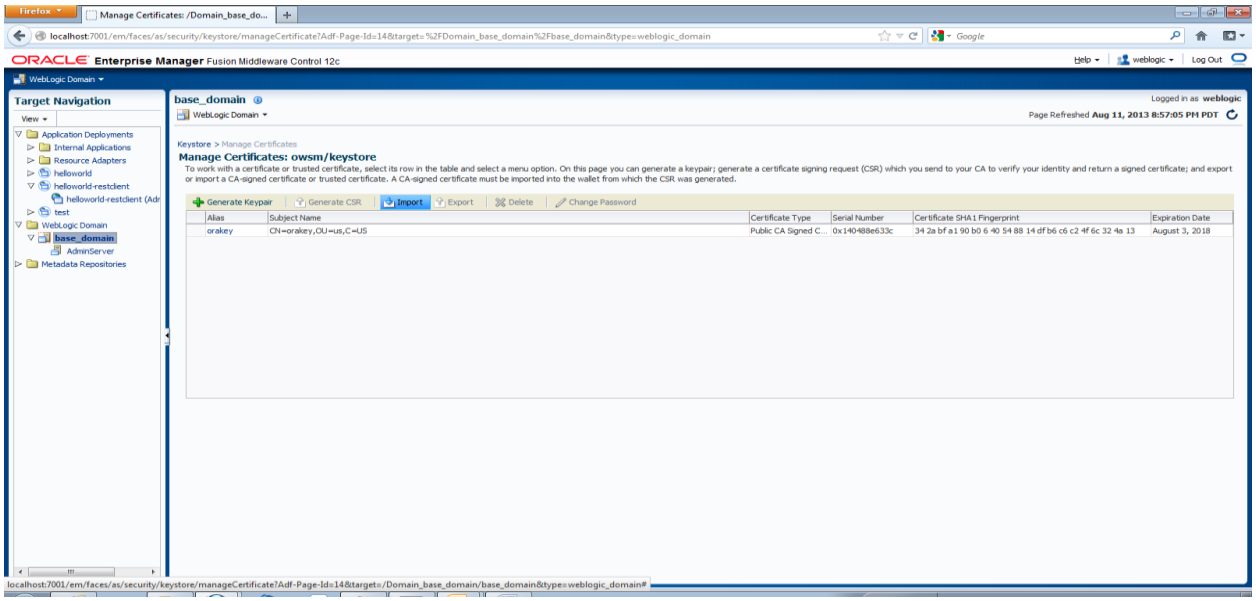


Click on “Export” and click on “Export Certificate” as shown in the screenshot below. You will be prompted to save the cert to a file. Save it on the file system.

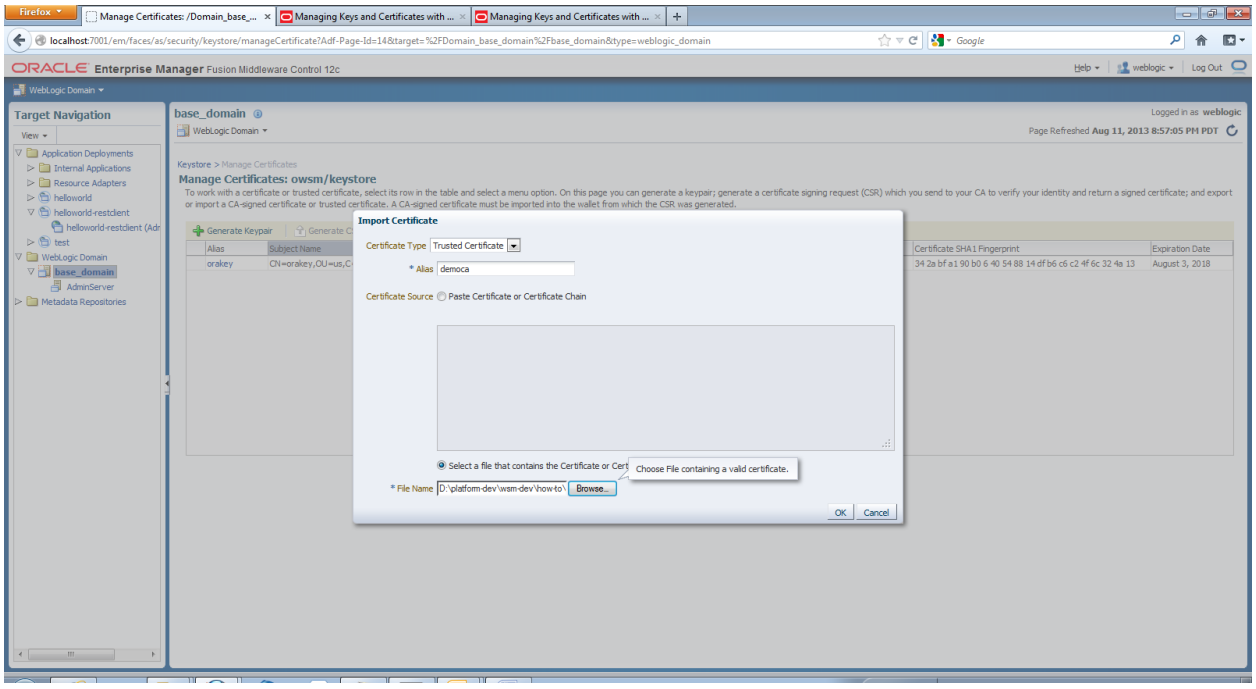


7.4.3 Import "democa" CA cert to "owsm" stripe

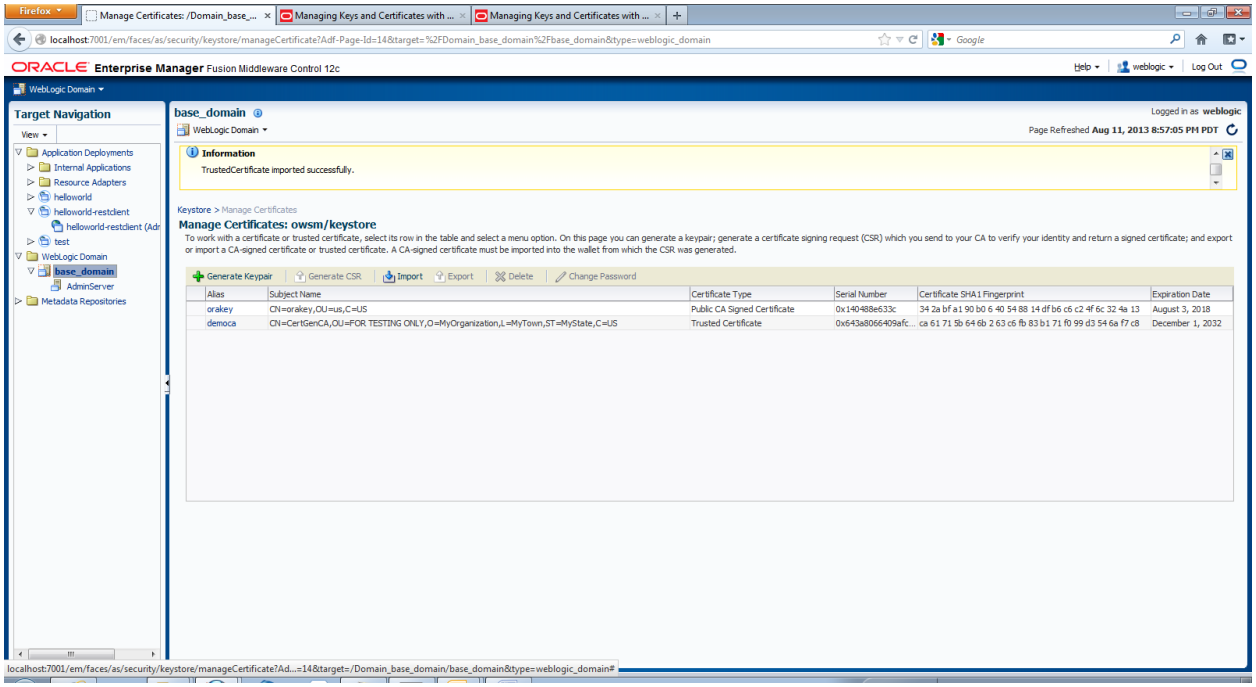
To import the "democa" cert, navigate to the "owsm" strip and click on "Manage". This will open the owsm/keystore page. Click on "Import" as shown in the screenshot below.



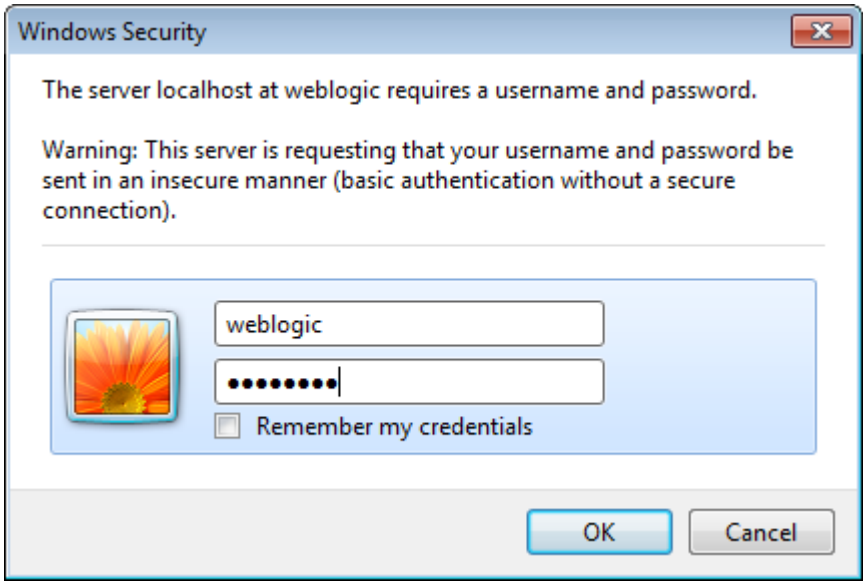
Clicking on "Import" will open a dialog box as shown in the screenshot below. Select the Certificate Type as "Trusted Certificate". Provide the alias as "democa" and select the file from the file system saved from section 7.4.2.

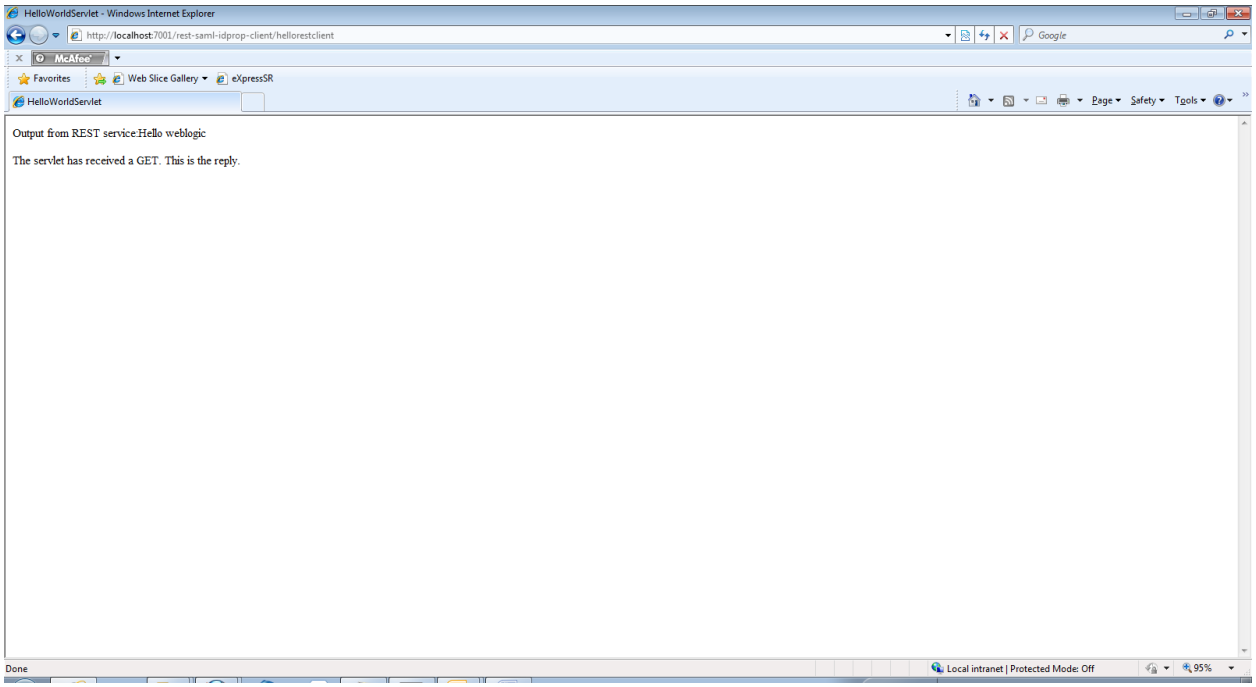


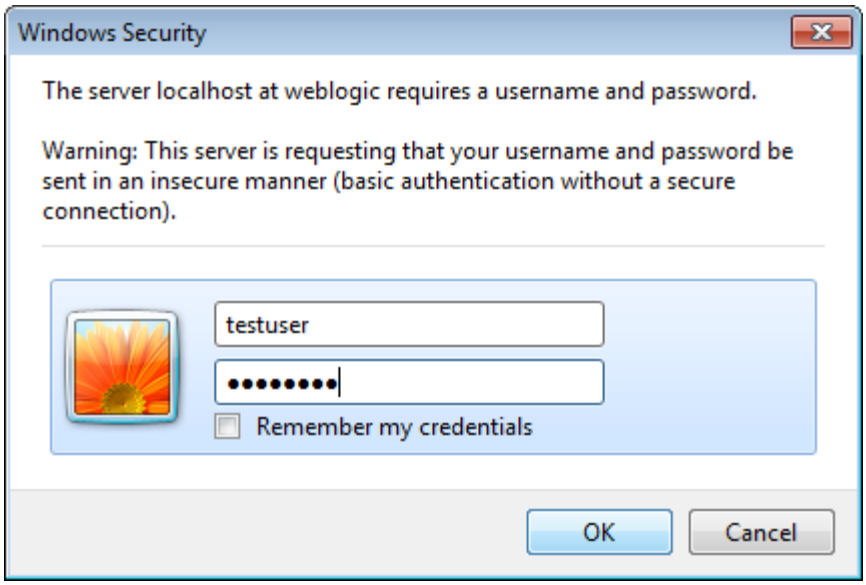
Click "OK"

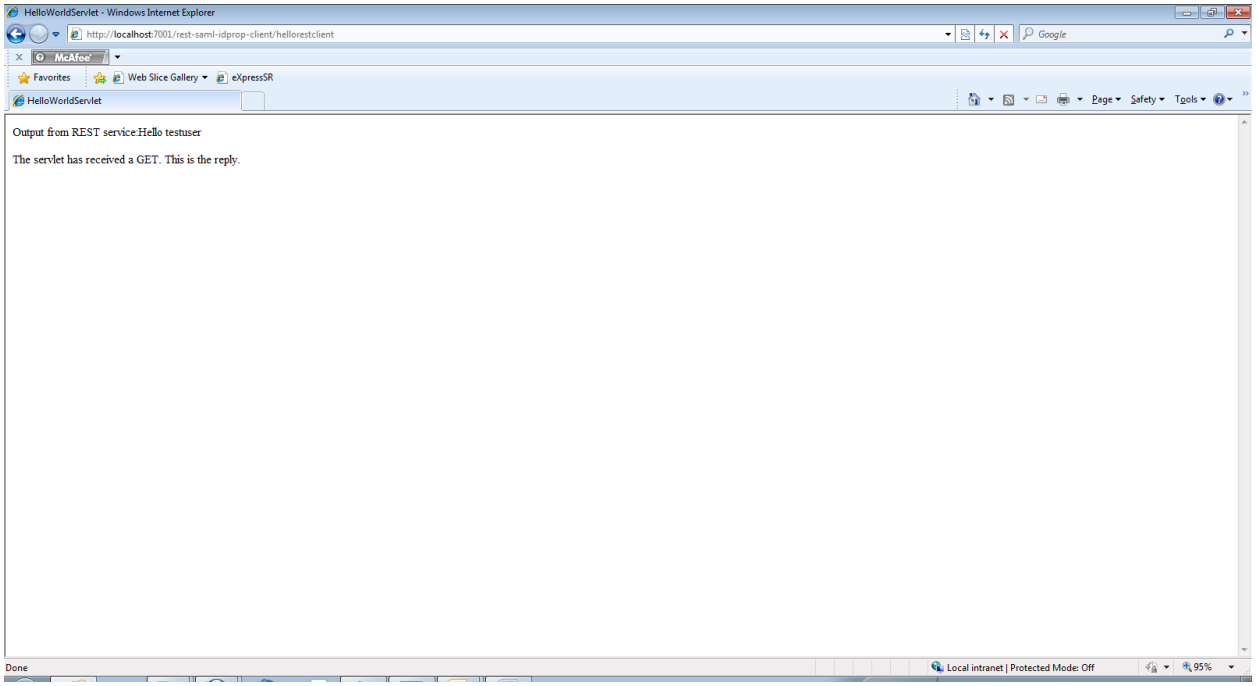

Firefox
Manage Certificates: /Domain_base_...
Managing Keys and Certificates with ...
Managing Keys and Certificates with ...
localhost:7001/em/faces/as/security/keystore/manageCertificate?Adf-Page-Id=14&target=%2FDomain_base_domain%2Fbase_domain&type=weblogic_domain
Google
ORACLE Enterprise Manager Fusion Middleware Control 12c
Help
weblogic
Log Out
WebLogic Domain
Target Navigation
View
Application Deployments
Internal Applications
Resource Adapters
helloworld
helloworld-restclient
helloworld-restclient (Adf
test
WebLogic Domain
base_domain
AdminServer
Metadata Repositories
base_domain
WebLogic Domain
Logged in as weblogic
Page Refreshed Aug 11, 2013 8:57:05 PM PDT
Information
TrustedCertificate imported successfully.
Keystore > Manage Certificates
Manage Certificates: owsm/keystore
To work with a certificate or trusted certificate, select its row in the table and select a menu option. On this page you can generate a keypair; generate a certificate signing request (CSR) which you send to your CA to verify your identity and return a signed certificate; and export or import a CA-signed certificate or trusted certificate. A CA-signed certificate must be imported into the wallet from which the CSR was generated.
Generate Keypair
Generate CSR
Import
Export
Delete
Change Password
Alias
Subject Name
Certificate Type
Serial Number
Certificate SHA1 Fingerprint
Expiration Date
orakey
CN=orakey,OU=us,C=US
Public CA Signed Certificate
0x140488e633c
34 2a bf a1 90 b0 6 40 54 88 14 df b6 c6 c2 4f 6c 32 4a 13
August 3, 2018
democa
CN=CertGenCA,OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US
Trusted Certificate
0x643a8066409afc...
ca 61 71 5b 64 6b 2 63 c6 fb 83 b1 71 f0 99 d3 54 6a f7 c8
December 1, 2032
localhost:7001/em/faces/as/security/keystore/manageCertificate?Ad...=14&target=/Domain_base_domain/base_domain&type=weblogic_domain#

8 Testing


Windows Security
The server localhost at weblogic requires a username and password.
Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).
weblogic
Remember my credentials
OK
Cancel


HelloWorldServlet - Windows Internet Explorer
http://localhost:7001/rest-saml-idprop-client/hellorestclient
Output from REST service:Hello weblogic
The servlet has received a GET. This is the reply.
Done


Windows Security
The server localhost at weblogic requires a username and password.
Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).
testuser
Remember my credentials
OK
Cancel


HelloWorldServlet - Windows Internet Explorer
http://localhost:7001/rest-saml-idprop-client/hellorestclient
Favorites
Web Slice Gallery
eXpressSR
HelloWorldServlet
Output from REST service:Hello testuser
The servlet has received a GET. This is the reply.
Done
Local intranet | Protected Mode: Off