Oracle Utilities
Customer Self Service

Whitepaper:

Creating and Deploying the Sample Mobile Client Application

Release 2.2.0.0

E78239-01

November 2016

ORACLE

Oracle Utilities Customer Self Service
Whitepaper: Creating and Deploying the Sample Mobile Client Application

Release 2.2.0.0

E78239-01

November 2016

Table of Contents

Chapter 1

Overview

This guide describes the steps necessary for the deployment of a sample Mobile Web Application for Oracle Utilities Customer Self Service (OUCSS). This guide also describes the steps required to generate native apps for various mobile platforms using the sample Mobile Web Application and Apache Cordova.

The sample Web Application has the following features:

- Alerts
- Account Summary
- Enroll
- Financial History
- Make Payment
- Outage Map
- Report Outage
- Register
- Service Charges to Date
- Usage Overview
- View Bill

The sample Mobile Web Application uses jQuery Mobile and calls REST services that are part of OUCSS 2.2.0.0 to fetch and update data from CSS application. The Web Application can be accessed from a Mobile/Desktop browser.

The sample Mobile Web Application can be modified as needed in accordance with the client environment.

The sample application MobileWebApp2.2.0.0.zip can be downloaded from Oracle Utilities Customer Self Service section on the Oracle Technology Network (OTN) web site (http://docs.oracle.com/cd/E72219_01/documentation.html) Refer to the OUCSS 2.2.0.0 Installation and Implementation Guides for instructions on installation and configuration of the REST services, OUCSS Shared libraries which the REST services reference, and the InboundServices application, which is required to access the REST Services.

There are several fixes to REST Services which are part of Patch 24501061 (available from the Oracle Software Delivery Cloud at edelivery.oracle.com), This patch must be applied for features in this sample application to work correctly.

The Enroll and Register features in this sample Mobile Web Application do *not* support the CSS-CCB User Linking feature which was introduced in the OUCSS 2.2 Portal application.

Additional Resources

Resource	Location
JQuery	http://jquery.com
JQuery Mobile	http://jquerymobile.com
jqPlot	http://www.jqplot.com
Apache Cordova	http://cordova.apache.org
Oracle Utilities Customer Self Service Installation Guide	Check Oracle Utilities Customer Self Service on the Oracle Technology Network (OTN) web site (http://docs.oracle.com/cd/E78536_01/index.htm) for the latest version of the documents.
Oracle Utilities Customer Self Service Implementation Guide	Check Oracle Utilities Customer Self Service on the Oracle Technology Network (OTN) web site (http://docs.oracle.com/cd/E78536_01/index.htm) for the latest version of the documents.

Acronyms

OUCSS - Oracle Utilities Customer Self Service

iOS – Apple mobile operating system

Android – Google mobile operating system

Supported Environments

Platforms

Apache Cordova supports several platforms including iOS, Android, Blackberry, Windows, Tizen. This whitepaper covers steps related to iOS and Android platforms.

Refer to the *Platform Guides section of Apache Cordova Documentation*(http://cordova.apache.org) - for the list of supported platforms and versions.

Software Requirements

The following are the software requirements for generating native apps for iOS and Android.

Eclipse

Download Eclipse from http://www.eclipse.org/downloads/. Download a package which enables creation of Java EE and web applications (e.g., Eclipse IDE for Java EE Developers).

iOS

- A computer running Mac OS.
- Xcode.
- iOS SDK. (See the "iOS Platform Guide" section of the *Apache Cordova Documentation* at http://cordova.apache.org for information regarding the supported versions for Xcode and iOS SDK.).
- An Apple Developer ID.

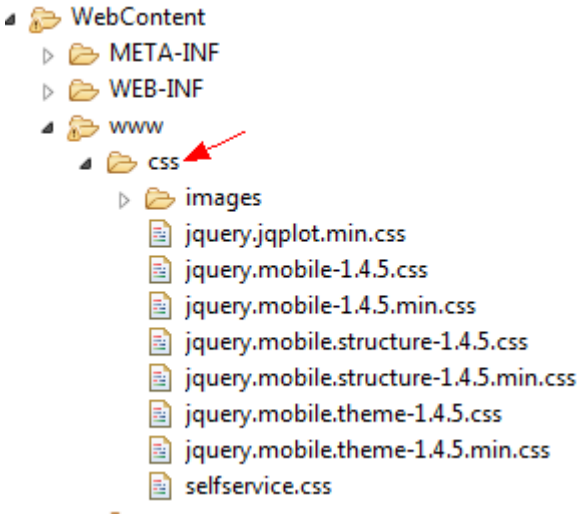
Android

Android SDK and its tools (see http://developer.android.com/sdk/) for supported versions.

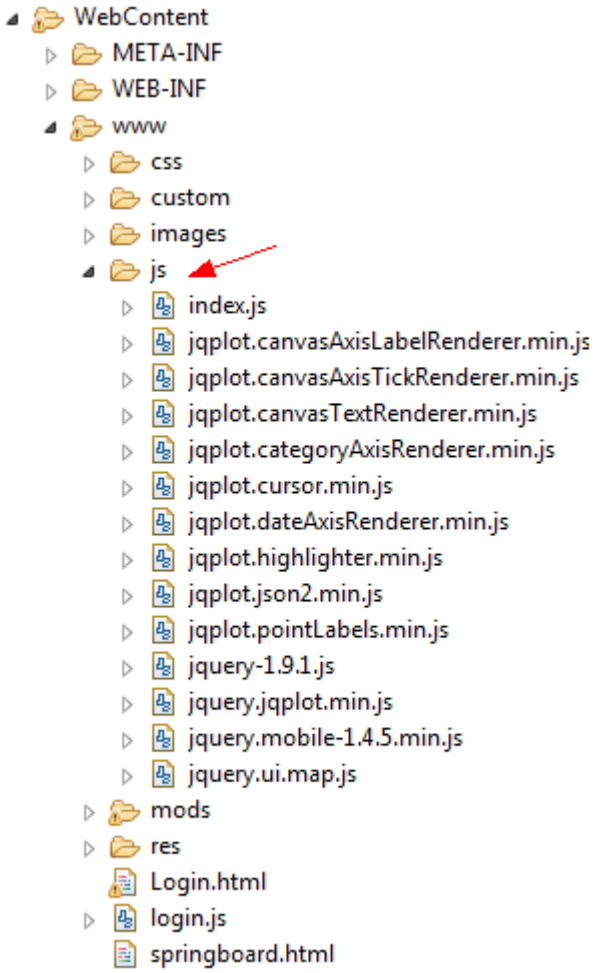
Third Party Software

The sample mobile application depends on the following third party software. Please download it from the suggested websites and place it in the folders specified.

CSS files should be placed under www/css folder



JavaScript files should be placed in the `www/js` folder.



The following files are referenced in the sample project:

- JQuery (http://www.jquery.com)
  - jquery-1.9.1.js
- JQuery Mobile (http://jquerymobile.com/)
  - jquery.mobile.structure-1.4.5.css
  - jquery.mobile.structure-1.4.5.min.css
  - jquery.mobile.theme-1.4.5.css
  - jquery.mobile.theme-1.4.5.min.css
  - jquery.mobile-1.4.5.css
  - jquery.mobile-1.4.5.min.css
  - jquery.mobile-1.4.5.min.js
- jqPlot (http://www.jqplot.com/) version 1.0.8r
  - jqplot.canvasAxisLabelRenderer.min.js
  - jqplot.canvasAxisTickRenderer.min.js
  - jqplot.canvasTextRenderer.min.js
  - jqplot.categoryAxisRenderer.min.js
  - jqplot.cursor.min.js

  - jqplot.dateAxisRenderer.min.js
  - jqplot.highlighter.min.js
  - jqplot.json2.min.js
  - jqplot.pointLabels.min.js
  - jquery.jqplot.min.js
  - jquery.jqplot.min.css
- Google Maps (http://code.google.com/p/jquery-ui-map/)
  - jquery.ui.map.js
- Apache Cordova
- Android OS and Android SDK
- iOS and Xcode

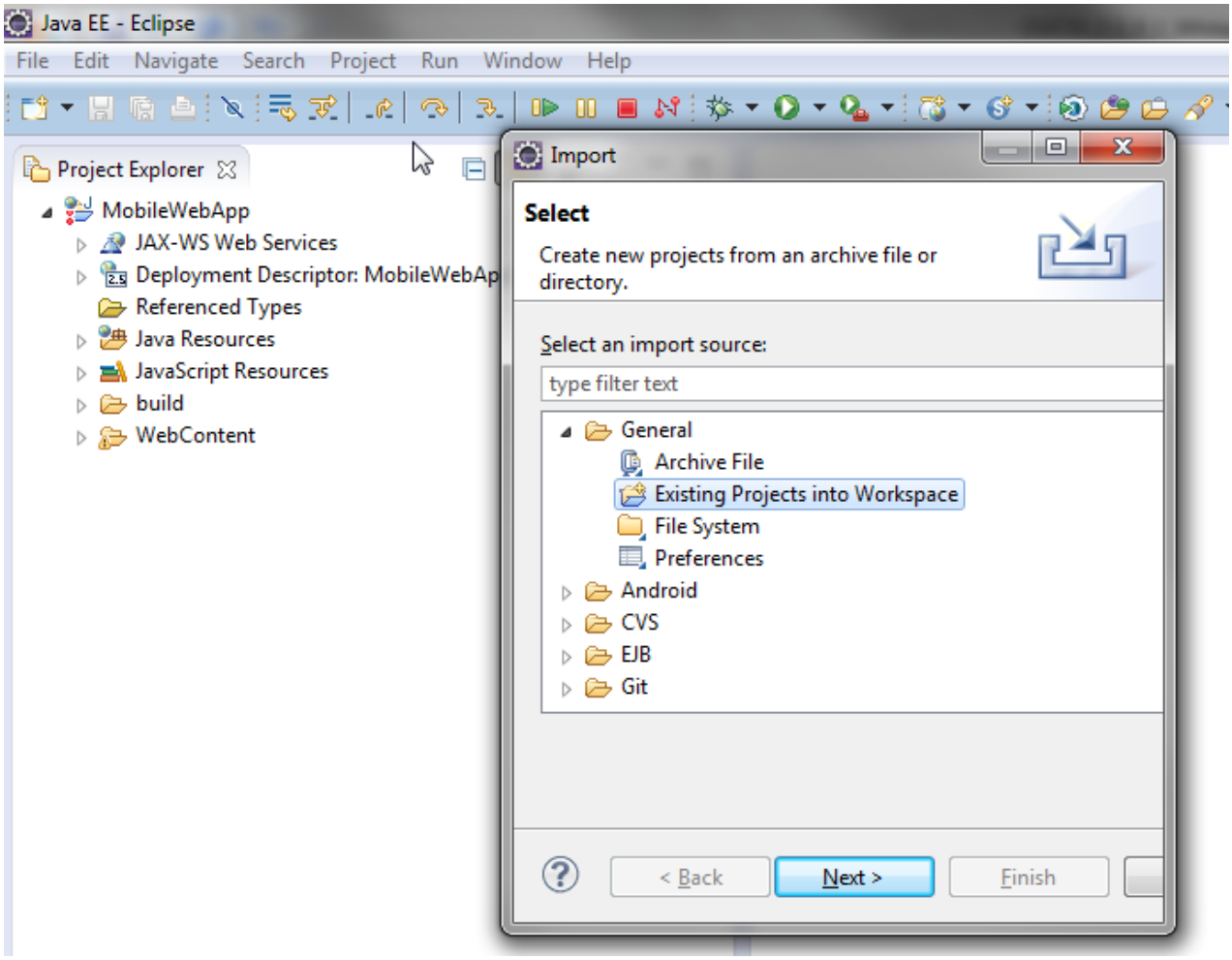
Chapter 2

Environment Setup

This section describes the settings and requirements for a successful installation of the Web Application and for generating native mobile apps.

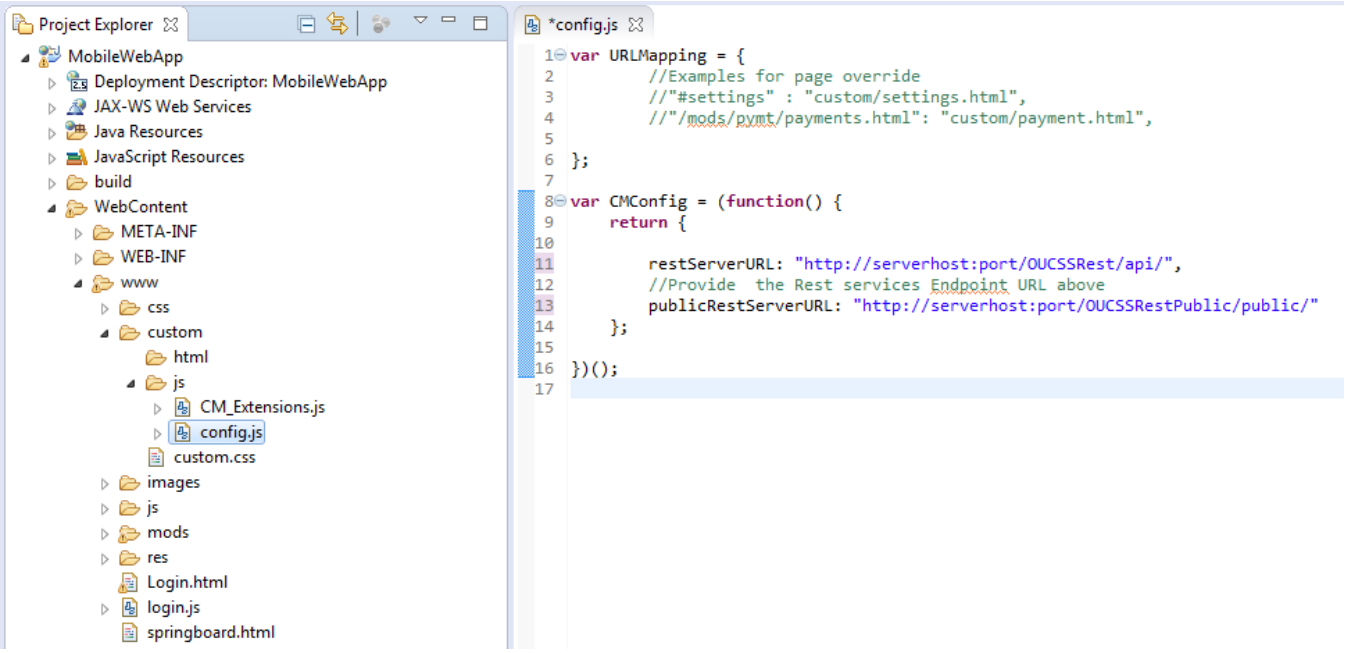
Setting Up Eclipse

1. Download Eclipse.
2. Extract the contents of MobileWebApp2.2.0.0.zip into a local folder.
3. Make sure all Third Party software is downloaded and copied to the respective folders as described in Third Party Software.
4. Create a workspace in Eclipse.
5. Import the MobileWebApp project into the Workspace: `File Menu > Import > General > Existing Projects into Workspace`.



6 Select the location of the expanded MobileWebApp directory, then click Finish to create a project.

7 Expand the `WebContent > www > custom` folder and provide the URL for secured and public REST services in `config.js` as follows:



8 Save the files.

9 OUCSSRegisterService is a secured REST service. To access it without authentication, the REST service must be made public; for the procedure, see Chapter 4, Public REST Services).

Deploying the Web Application to a Server

To deploy the Web Application to a server and access it from a desktop/mobile browser:

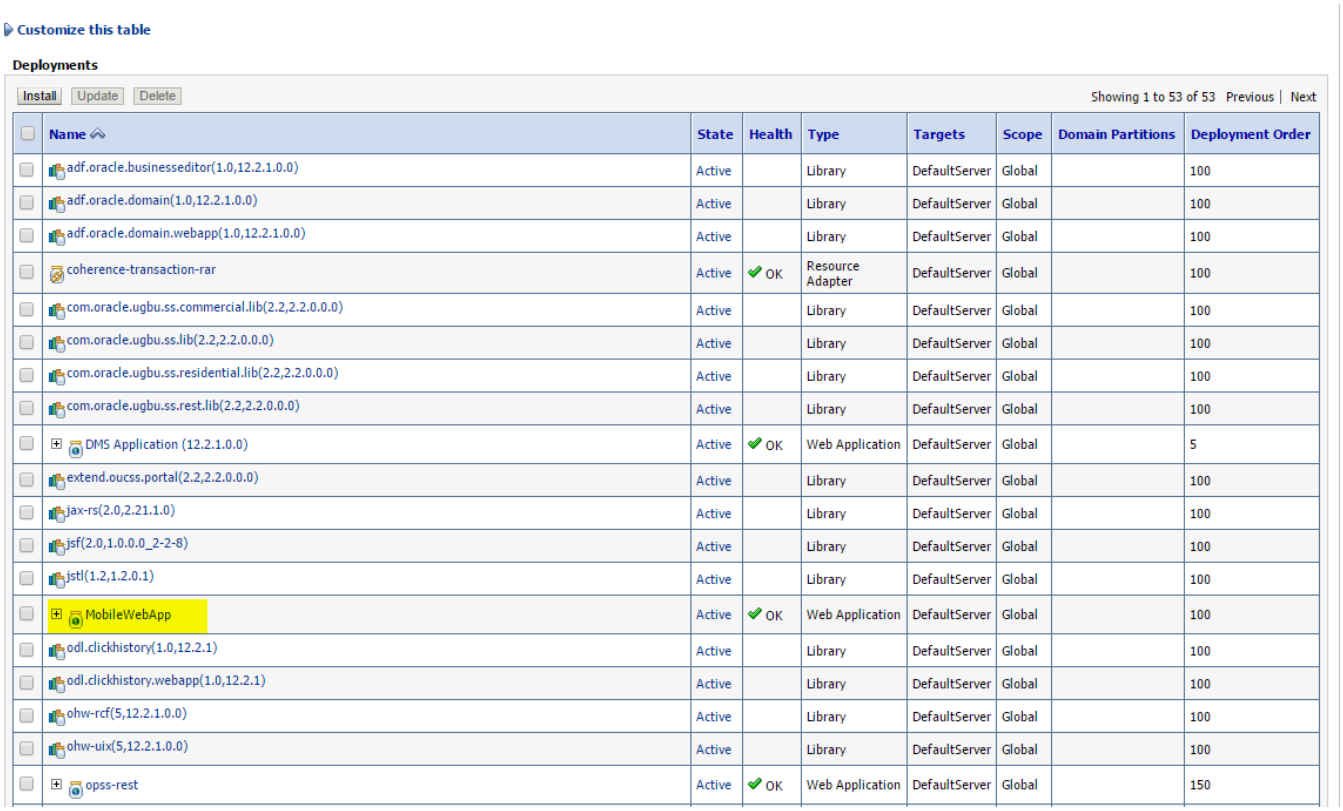
1 Right click on the MobileWebApp project, then choose `Export > WAR File`.

2 Specify the location of the war file.

Note: The Web Application should be deployed in the same server as OUCSS REST services to avoid Cross-Origin Resource Sharing (CORS) issues.

3 Copy the war file to the OUCSS server where REST services are deployed. The war file should be copied to the Admin server Upload folder. e.g., `xxx/user_projects/domains/xxx_domain/servers/AdminServer/upload`.

4 From the WebLogic Administrator console choose `Deployments > Install`.



5. Select MobileWebApp.war, then click Next.
6. Select Install this deployment as an application, then click Next.
7. Select the server on which the OUCSS REST services are deployed as Target.
8. Click Finish.

   The MobileWebApp should be deployed successfully.
9. Access the Web App from a desktop/mobile browser as follows:

```
http://server:port/MobileWebApp/www/Login.html
```

Setup for iOS Native App

1. Ensure you have the following installed and running:
   - A computer running Mac OS.
   - Xcode. After installing Xcode you must run it at least once and complete the Apple licensing and setup dialogs.
   - iOS SDK. (See the "iOS Platform Guide" section of the *Apache Cordova Documentation* at http://cordova.apache.org for information regarding the supported versions for Xcode and iOS SDK.)
2. Create an Apple developer ID.
3. Install Cordova by downloading Node.js. (For detailed instructions, see the "Command Line Interface" section of the *Apache Cordova Documentation* at http://cordova.apache.org.)
4. Create a Cordova project as follows. xxx below depends on the Apple developer ID configuration.

```
cordova create CSSMobileWebApp com.xxx. MobileWebApp CSSMobileWebApp
cd CSSMobileWebApp
```

5. Copy the **www** folder from the sample MobileWebApp project to CSSMobileWebApp.

6 Add platform support for iOS as follows:

```
cordova platform add ios
cordova build
```

7 Open the project in Xcode. Double-click to open **CSSMobileWebApp\platforms\ios\CSSMobileWebApp.xcodeproj.**

For detailed instructions on generating a native app for iOS, see the "iOS Platform Guide" section of the *Apache Cordova Documentation* at http://cordova.apache.org.

Setup for Android Native App

1 Ensure you have the following:

- A computer running Windows, Linux, or Mac OS.
- Android SDK with Platform 2.3 or later, and the corresponding tools, installed on the OS.

2 Download Eclipse from http://www.eclipse.org/downloads/. Download a package download which enables creation of Java EE and web applications e.g. Eclipse IDE for Java EE Developers.

3 Download and install the Android SDK from http://developer.android.com/sdk/index.html.

4 Install Eclipse and then install the Android development tools (ADT) plug-in as given in the instructions at http://developer.android.com/sdk/installing/installing-adt.html.

5 For the Apache Cordova, Eclipse and Android SDK setup follow instructions in Android Platform Guide section of Apache Cordova Documentation(http://cordova.apache.org) .

6 Create a Cordova project as follows.

```
cordova create CSSMobileWebApp com.xxx. MobileWebApp CSSMobileWebApp
cd CSSMobileWebApp
```

7 Copy the `www` folder from the sample MobileWebApp project to CSSMobileWebApp.

8 Change the content in config.xml to point to Login.html.

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.oracle.MobileWebApp" version="0.0.1" xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>"CSSMobileWebApp"</name>
    <description>
        A sample Apache Cordova application that responds to the deviceready event.
    </description>
    <author email="dev@cordova.apache.org" href="http://cordova.io">
        Apache Cordova Team
    </author>
    <content src="Login.html" />
    <plugin name="cordova-plugin-whitelist" version="1" />
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <platform name="android">
        <allow-intent href="market:*" />
    </platform>
    <platform name="ios">
        <allow-intent href="itms:*" />
        <allow-intent href="itms-apps:*" />
    </platform>
</widget>
```

9 Add platform support for Android using the following commands:

```
cordova platform add android
cordova build
```

10 Open the project in Eclipse and deploy it to Emulator or an Android device.

Chapter 3

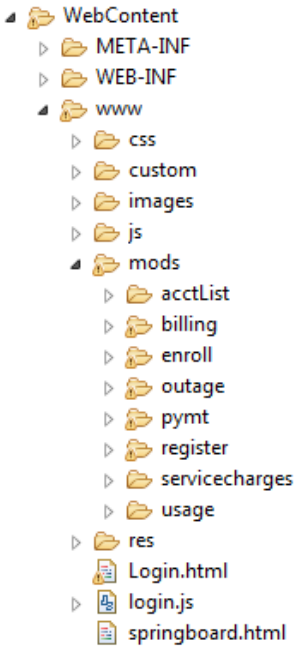
Sample Project

The sample Web Application described in this section (available for download from the Oracle Utilities Customer Self Service documentation section on the Oracle Technology Network at http://www.oracle.com/technetwork/apps-tech/utilities/documentation/index.html) invokes OUCSS REST services to fetch/update data. The sample project can be modified using the Eclipse IDE.

Note: See Third Party Software for information on the software required for the development and building of the Sample mobile app.

Architecture

MobileWebApp



- Login.html is the first page launched. This page is used to authenticate and to store credentials for future logins.
- springboard.html has the menu structure.
- js/index.js has the following base utility functions:
  - The js folder has all the required javascript files for jQuery, jQuery Mobile, cordova, other plugins like jQPlot, etc.
  - The mods folder contains html and javascript files for specific pages.
  - The css folder contains css files for jQuery Mobile and OUCSS.

Utility JavaScript Code

index.js

Base JavaScript with all the utility functions that are loaded when the CSS App starts. The data is retained for the lifetime of the application.

- OUCSS. Utilities: Has the utility functions to log to the console.
- OUCSS.Storage: Functions to store data to Local and Session storage per the HTML5 specification. This specification only supports key-value pairs. Local storage is persistent and will be retained until the user deletes the cache. Session storage is per session. Supported by all browsers and Mobile OSs. (**Not used. Web storage is used instead.)**
- setTheme: To change the theme for the current page.
- OUCSS.CacheMgr: Store Lookups and labels per session. This is used to get the data for Labels and Lookups.
- OUCSS.AJAX: Used to make AJAX calls to REST services.
- OUCSS.DB: Functions using the Web SQL feature to store Labels and Lookups.
  - Local persistent database implemented by browser and OS.
  - Supported by Safari, Chrome, iOS, Android.
  - This Util function creates the database if it does not exist, calls REST services to read Labels and Lookups, stores them in Web SQL, and populates OUCSS.CacheMgr.
  - In future invocations of the app, if data already exists , this function will read data from the database and populate OUCSS.CacheMgr.
- OUCSS.CSSApp: Core functions to handle Page context, page rendering, and form submit.
  - get/set PageContext.
  - get/set GlobalContext.
  - pageBeforeChange: Called when moving from one page to another. Used to call a custom page instead of the base page. Also passes page parameters.
  - renderPage: Render the page based on the "oucss-" tags on the elements. Supports table, div, select, label, input.
  - pageBeforeShow:
    - Displays Labels and Lookups based on “oucss-label” and “oucss-lookup” tags.
    - Invoke the REST service to load the data and call renderPage. If the page has the same function implemented as OUCSS.PageEvents.pageId.pageBeforeShow, then call that function instead.
    - If a custom JS function exists in the form of OUCSS.PageEvents.pageId.cmext.pageBeforeShow, then call that function.
  - pageSubmitForm:
    - If a custom JS function exists in the form of OUCSS.PageEvents. **pageId**.cmext.pageSubmitForm, then call that function.
    - Submit the form to the REST service based on the “oucss-service” tag on the page. If the page has implemented the same function, then call the page-specific function instead.

Anatomy of a CSS Mobile Page

HTML

- Every jQuery Mobile page should have divs with data-roles of page, header, content, and footer.

- Custom “oucss-” tags can be used to link the UI elements with the data:
  - “oucss-service” tag on the page element to indicate the name of the REST service for getting the page data.
  - “oucss-path” tag on any element represents the relative path of the data element in the REST service’s output.
  - “oucss-label” tag represents the label name.
  - “oucss-lookup” tag represents the lookup name.

Page Structure (Snippet)

```
<div oucss-form="paymentForm" data-role="page" id="makePaymentPage" data-mini="true" class="pageCls">

        <script src="payments.js"></script>
        <script src="payments_cmExt.js"></script>

        <div data-role="header" data-position="fixed" class="cssheader">
            <div class="ui-btn-left" data-role="controlgroup">
                <table id="menuId">
                    <tr>
                        <td><a data-icon="grid" data-iconpos="notext"
                            data-role="button" href="../../springboard.html" data-rel="back">Menu</a></td>
                        <td class="backBtn">Menu</td>

                    </tr>
                </table>
            </div>
            <h1>
                <span class="headerLargeFont">Pay Now</span>
            </h1>
        </div>

        <div data-role="content" id="payment-content" class="card">

            <div>
                <form name="paymentForm" id="paymentForm" method="post"
                    data-ajax="false">

                    <div class="center-wrapper">
                        <span class="textLargeFont"><b id="acctInfoPay">Account
                                : </b></span>
                    </div>

                    <div data-role="fieldcontain" id="paymentInfoDiv"
                        class="ui-hide-label">


                        <label for="paymentType" class="select"
                            oucss-label="AUTOPAY_PAYMENT_TYPE_LBL" data-mini="true">Pymt
                            Type</label> <select data-theme="d" name="md_paymentType"
                            oucss-path="mainData.paymentType" id="paymentType"
                            oucss-lookup="PAYMENT_TYPES" data-mini="true" class="dropDownTxt">
                        </select> <label for="paymentAmount" oucss-label="AMOUNT_LBL">Amt</label> <input
                            type="text" oucss-path="mainData.paymentAmount"
                            name="md_paymentAmount" id="paymentAmount" value=""
                            data-mini="true" placeholder="Amount ($)" />
                    </div>
```

JavaScript

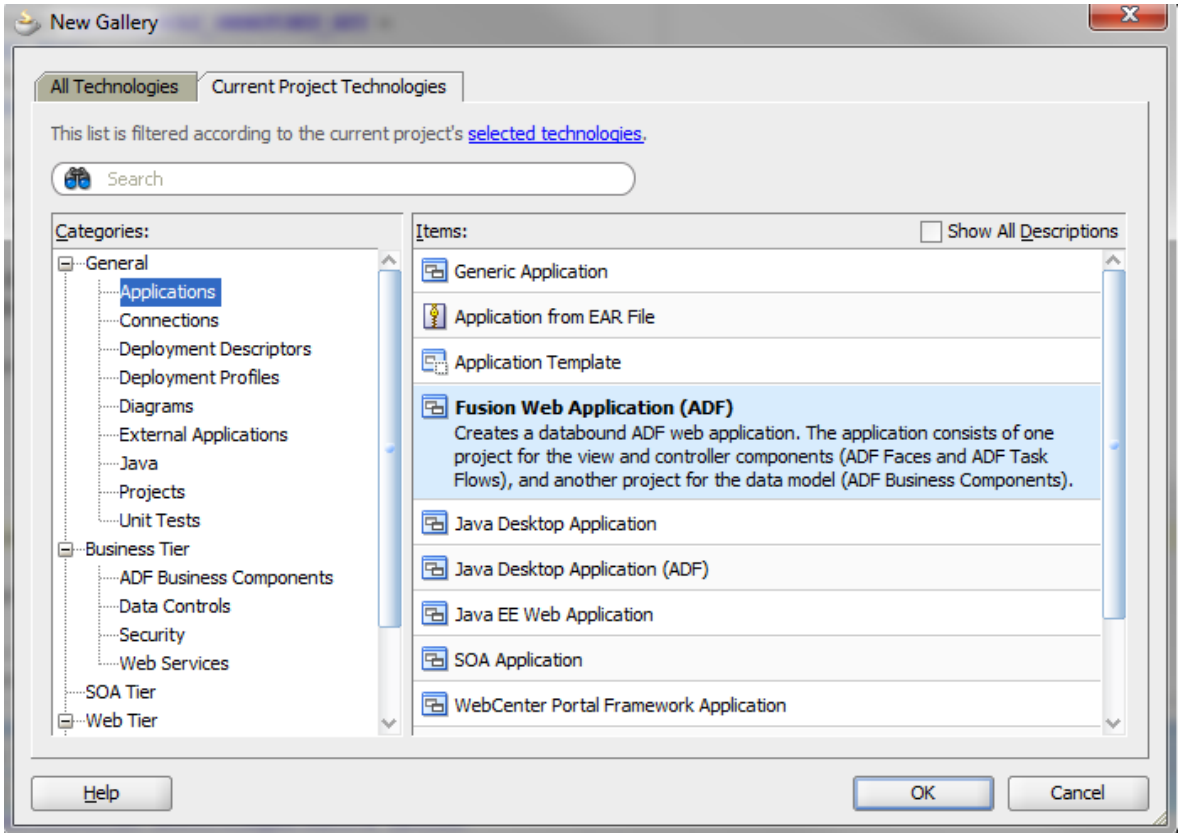
- Page-specific code to bind actions for buttons, etc.
- OUCSS.PageEvents.**pageId:** Functions can be written for specific Page Events for non-standard code.
  - **pageBeforeShow**: Called instead of the base function in index.js. Can be used to have page specific code to call the REST service and render the page.
  - **pageShow**: Called after jQuery enhances the page. Can be used to have page-specific code to call the REST service to get data and to change the page layout.
  - **pageSubmitForm**: Called instead of base function in index.js. Can be used to have page-specific code to submit the page.

Chapter 4

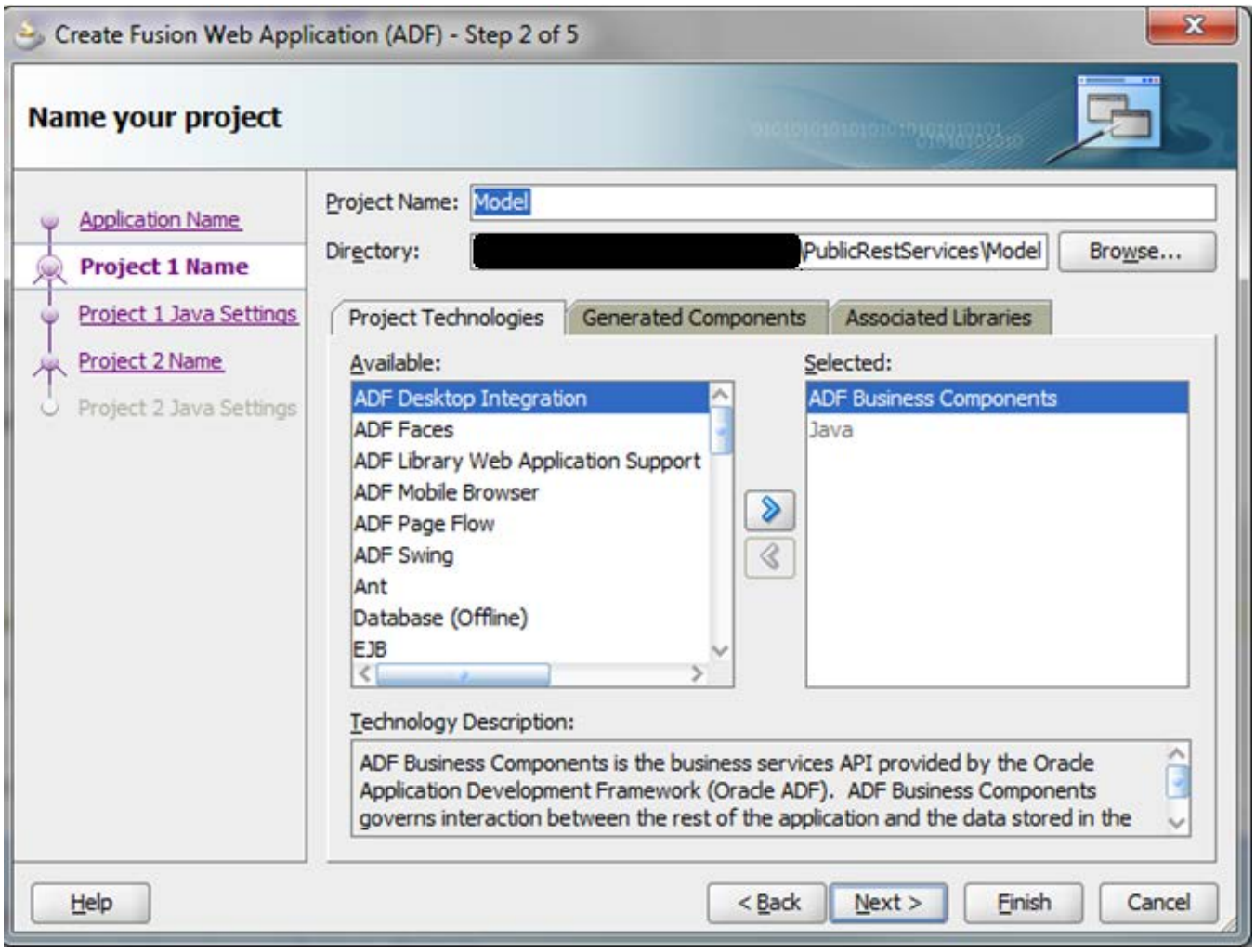
Public REST Services

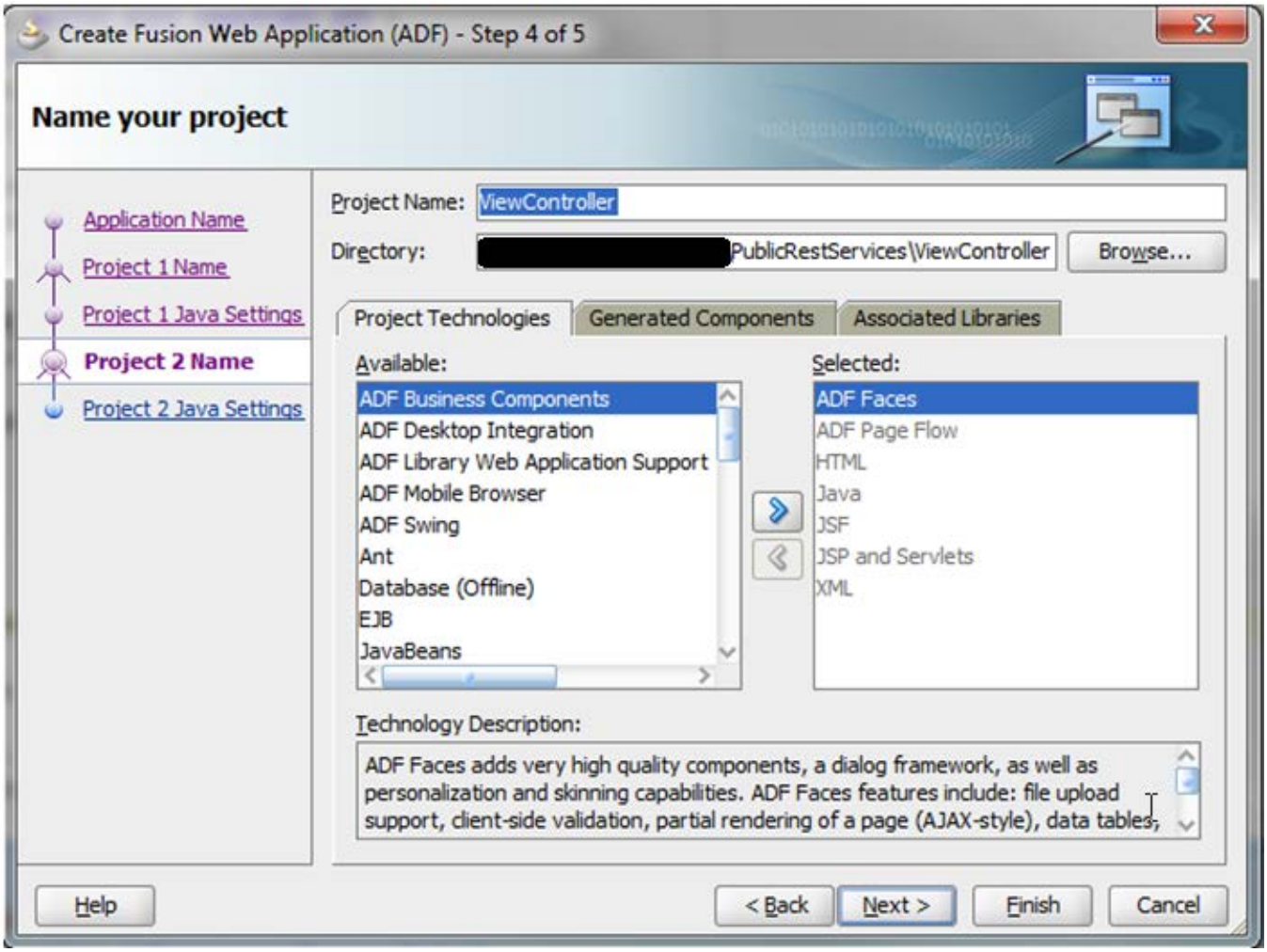
All OUCSS REST services are secured by an OWSM policy. If a REST service must be accessed without authentication (e.g., OUCSSRegisterService in the sample app) then that REST service can be made public as described in the following procedure.

1 Open JDeveloper and create a new application by selecting **New** and choosing the **Fusion Web Application(ADF)** template as shown in the following image:

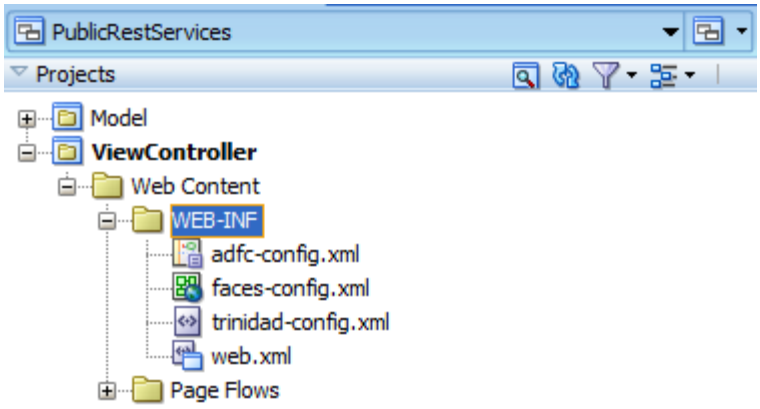


2 Provide an appropriate name and directory for the application in the wizard. Follow the wizard, retaining all default values to completion.

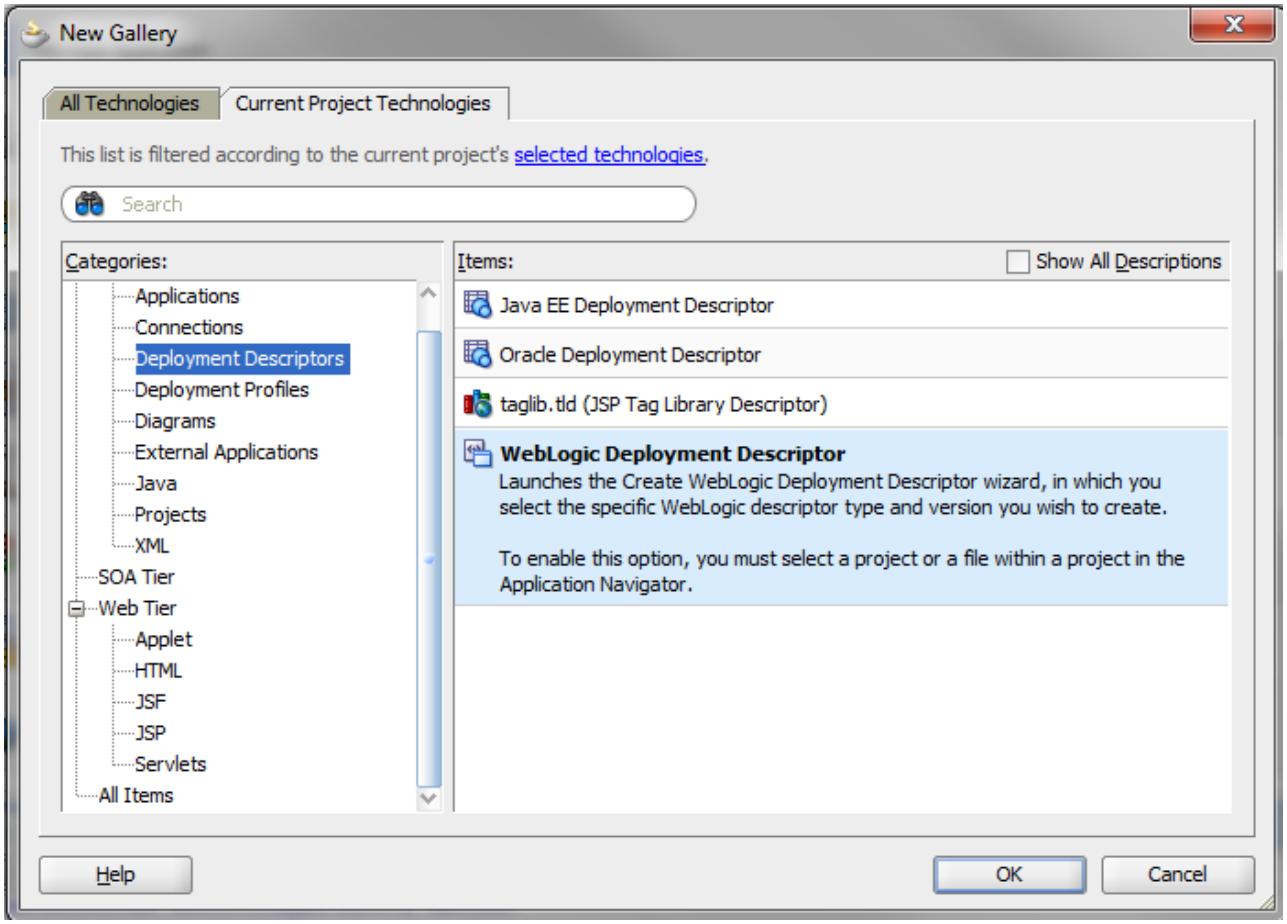




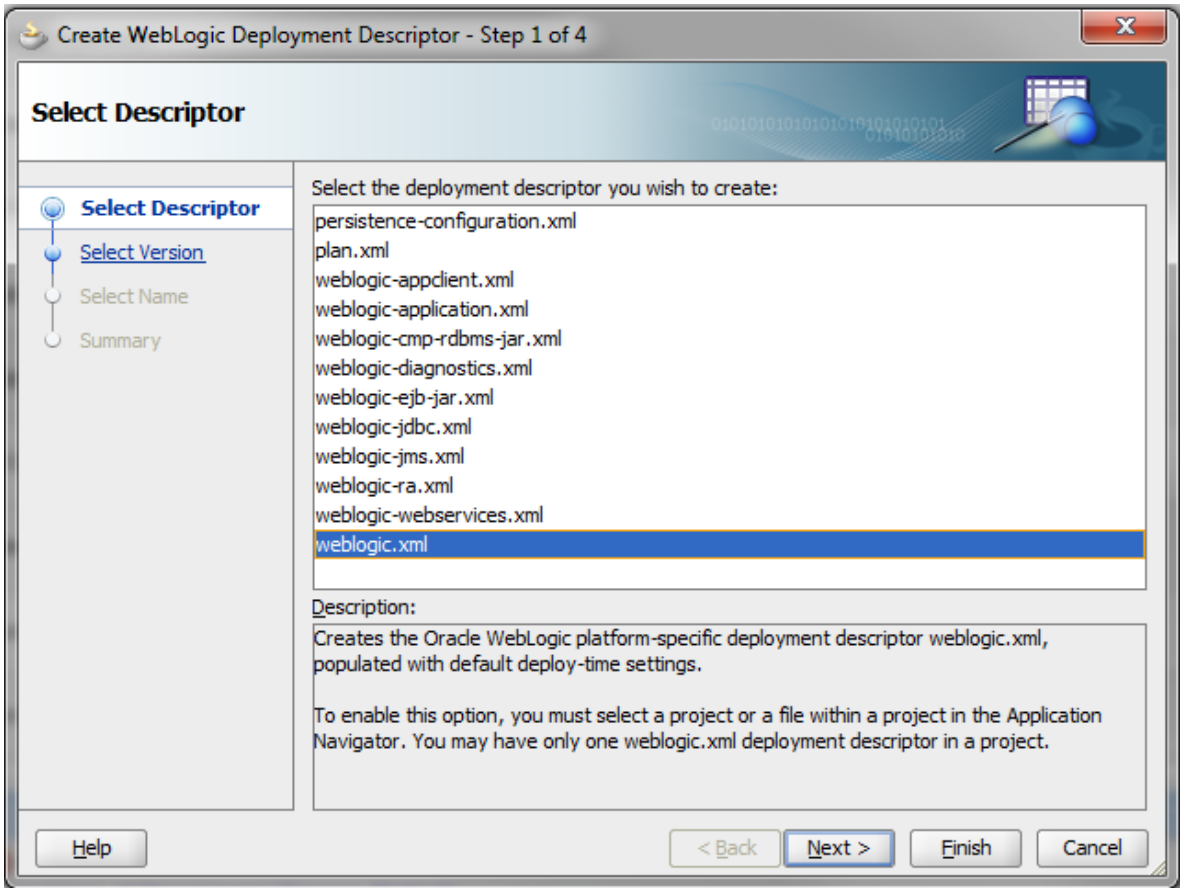
3 Verify that the application in JDeveloper has the following structure:



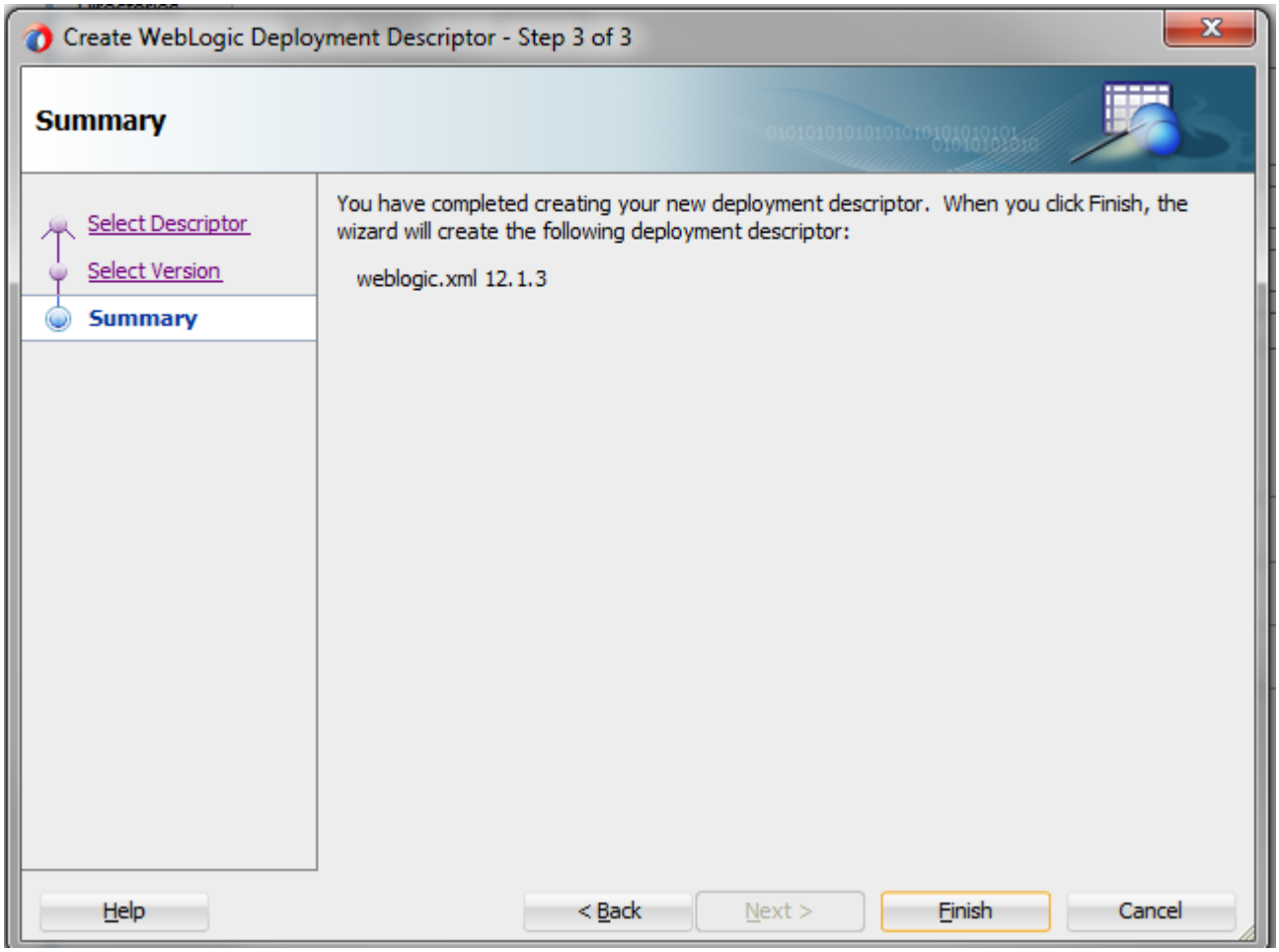
4 To begin the process of creating the weblogic.xml descriptor, **r**ight-click on the **WEB-INF** folder and select **New**. In the next window, select `Deployment Descriptors > Weblogic Deployment Descriptor`.



5 Select **weblogic.xml**, as shown in the following image.

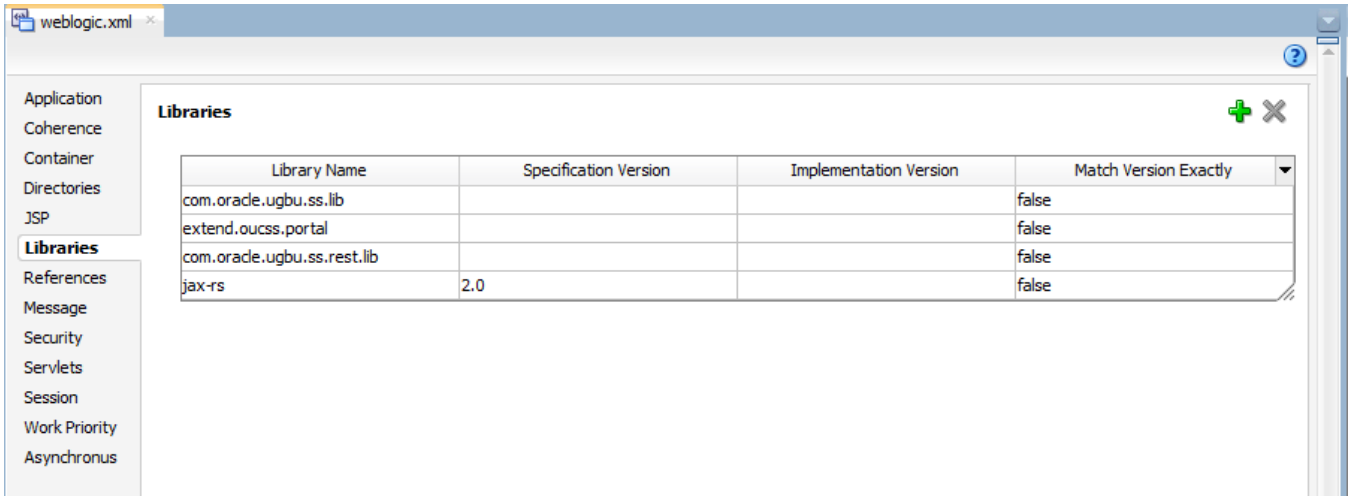


6 Follow the wizard, retaining all defaults, then press **Finish** to create the weblogic.xml descriptor.

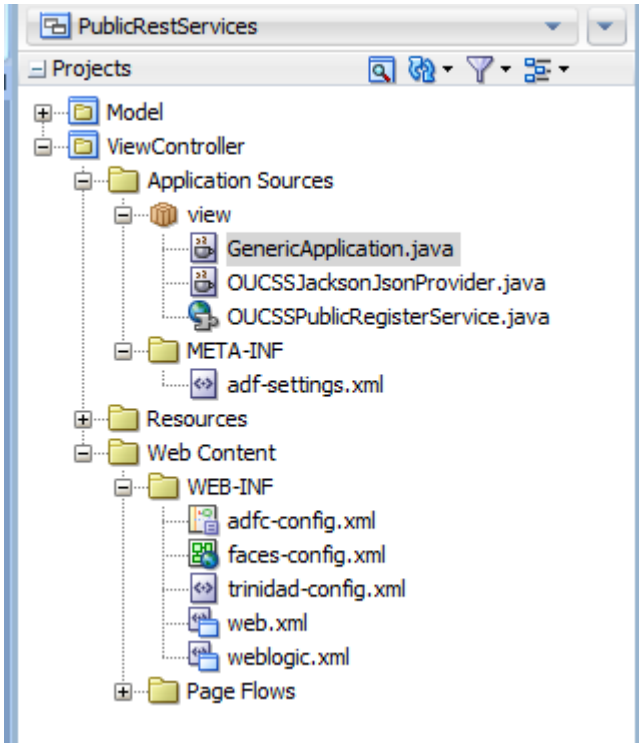


7 Select the Libraries tab in the weblogic.xml and add the following entries to refer to the OUCSS shared libraries:

- com.oracle.ugbu.ss.lib
- extend.oucss.portal
- com.oracle.ugbu.ss.rest.lib
- jax-rs 2.0



8 Create a RESTful service invoking OUCSSRegisterService with these sample Java files:



GenericApplication.java

```
package view;

import java.util.HashSet;
import java.util.Set;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("public")
public class GenericApplication
  extends Application
{
  public Set<Class<?>> getClasses()
  {
    Set<Class<?>> classes = new HashSet<Class<?>>();

    // Register feature classes.
    classes.add(org.glassfish.jersey.jackson.JacksonFeature.class);

    // Register OUCSS Register Service Resource.
    classes.add(OUCSSPublicRegisterService.class);

    // Register provider classes.
    classes.add(OUCSSJacksonJsonProvider.class);


    return classes;
  }
}
```

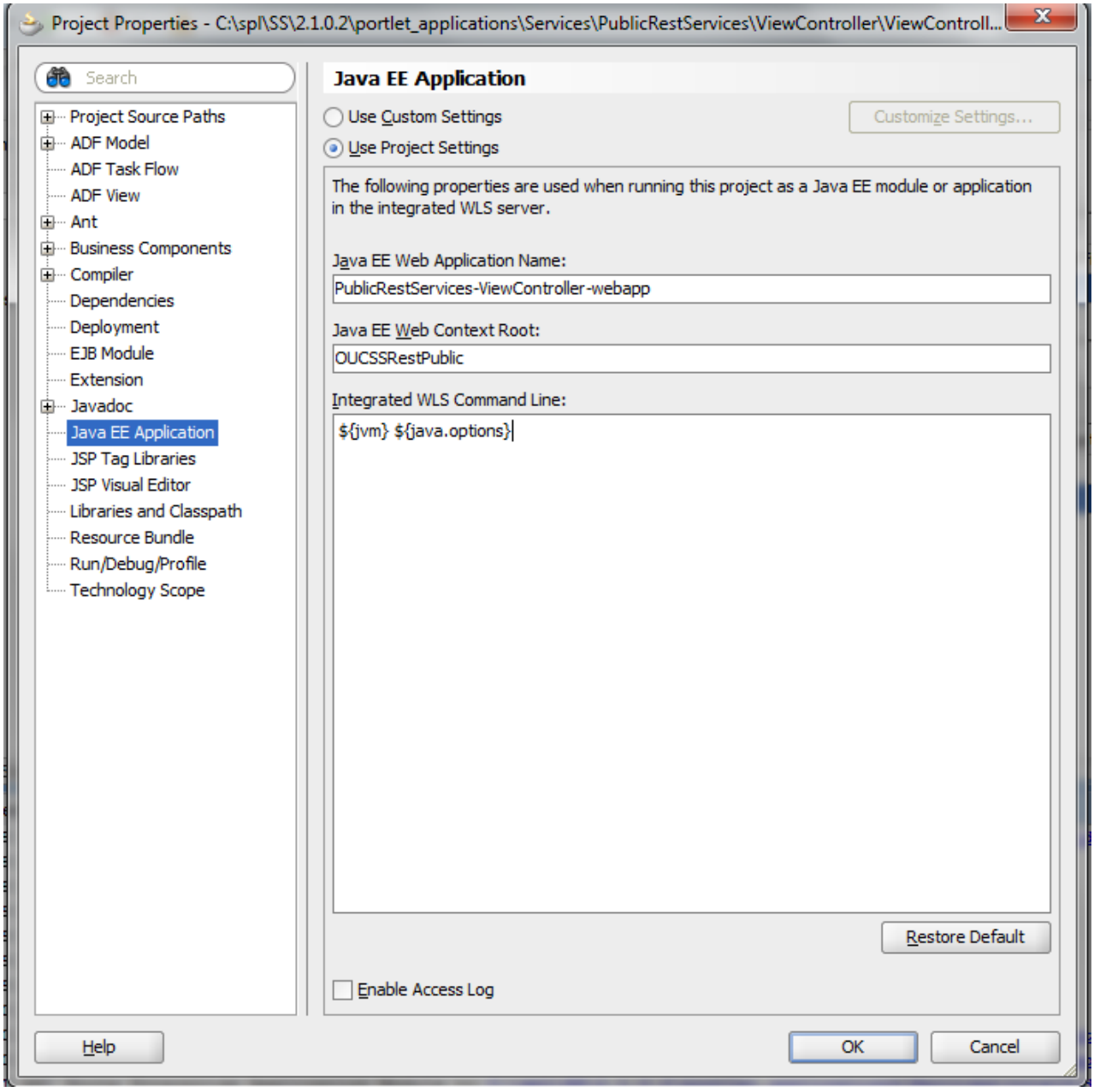
OUCSSPublicRegisterService.java

```
package view;

import com.oracle.ugbu.ss.rest.model.shared.internal.service.OUCSSAbstractRestService;

import javax.ws.rs.Path;

@Path("OUCSSRegisterService")
public class OUCSSPublicRegisterService extends OUCSSAbstractRestService
{
  public OUCSSPublicRegisterService(String serviceName)
    {

      super("OUCSSRegisterService");
    }

  public OUCSSPublicRegisterService()
    {

      this("OUCSSRegisterService");
    }
}
```

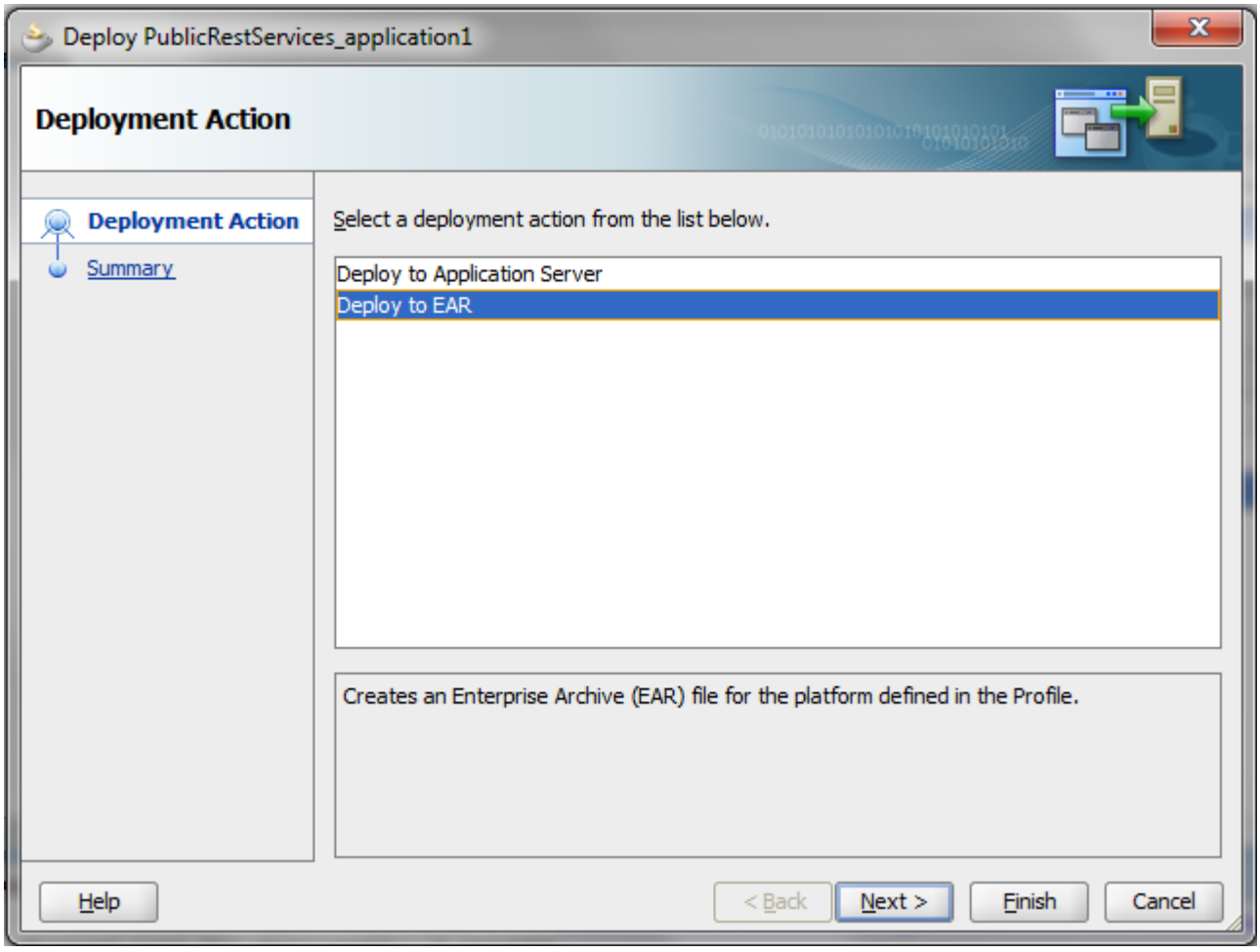
OUCSSJacksonJsonProvider.java

```
package view;

import javax.ws.rs.ext.ContextResolver;

import com.fasterxml.jackson.annotation.JsonInclude.Include;
import com.fasterxml.jackson.databind.MapperFeature;
import com.fasterxml.jackson.databind.ObjectMapper;


public class OUCSSJacksonJsonProvider
  implements ContextResolver<ObjectMapper>
{

  private static final ObjectMapper MAPPER = new ObjectMapper();

  static
  {
    MAPPER.setSerializationInclusion(Include.NON_EMPTY);
    MAPPER.disable(MapperFeature.USE_GETTERS_AS_SETTERS);
  }

  public OUCSSJacksonJsonProvider()
  {
    super();
  }

  @Override
  public ObjectMapper getContext(Class<?> type)
  {
    return MAPPER;
  }
}
```

9 Right-click the ViewController project, then choose Project properties. Set the Web Context Root of the application to an appropriate name (e.g, OUCSSRestPublic) as used in the sample application. This context will be used to access the public REST service.


Project Properties - C:\spl\SS\2.1.0.2\portlet_applications\Services\PublicRestServices\ViewController\ViewControll...
Search
Project Source Paths
ADF Model
ADF Task Flow
ADF View
Ant
Business Components
Compiler
Dependencies
Deployment
EJB Module
Extension
Javadoc
Java EE Application
JSP Tag Libraries
JSP Visual Editor
Libraries and Classpath
Resource Bundle
Run/Debug/Profile
Technology Scope
Java EE Application
Use Custom Settings
Customize Settings...
Use Project Settings
The following properties are used when running this project as a Java EE module or application in the integrated WLS server.
Java EE Web Application Name:
PublicRestServices-ViewController-webapp
Java EE Web Context Root:
OUCSSRestPublic
Integrated WLS Command Line:
${jvm} ${java.options}
Restore Default
Enable Access Log
Help
OK
Cancel

10 Deploy the Application to an EAR file from JDeveloper. Deploy this EAR file to the WebLogic server using the server console through `Deployments -> Console`, as described in **Deploying the Web Application to a Server**.



11 OUCSSRegisterService can be accessed using the URL `http://host:port/OUCSSRestPublic/public/OUCSSRegisterService`.