



Web Services, EBS R11 and R12 with SOA

Presenter: Sarah Sinclair and Gareth Roberts

This is a technical overview of how to get a web service up and running in EBS R11 with no SOA install.

There will be lots of ideas for those who need to cater for Web Services in Oracle but aren't on R12 and don't want a whole SOA install.

Gareth Roberts from Virtuate will also contrast R11 Web Services with a summary of Web Services in R12 utilizing the Release 12.1 Integrated SOA Gateway product.

What is Service Enablement?

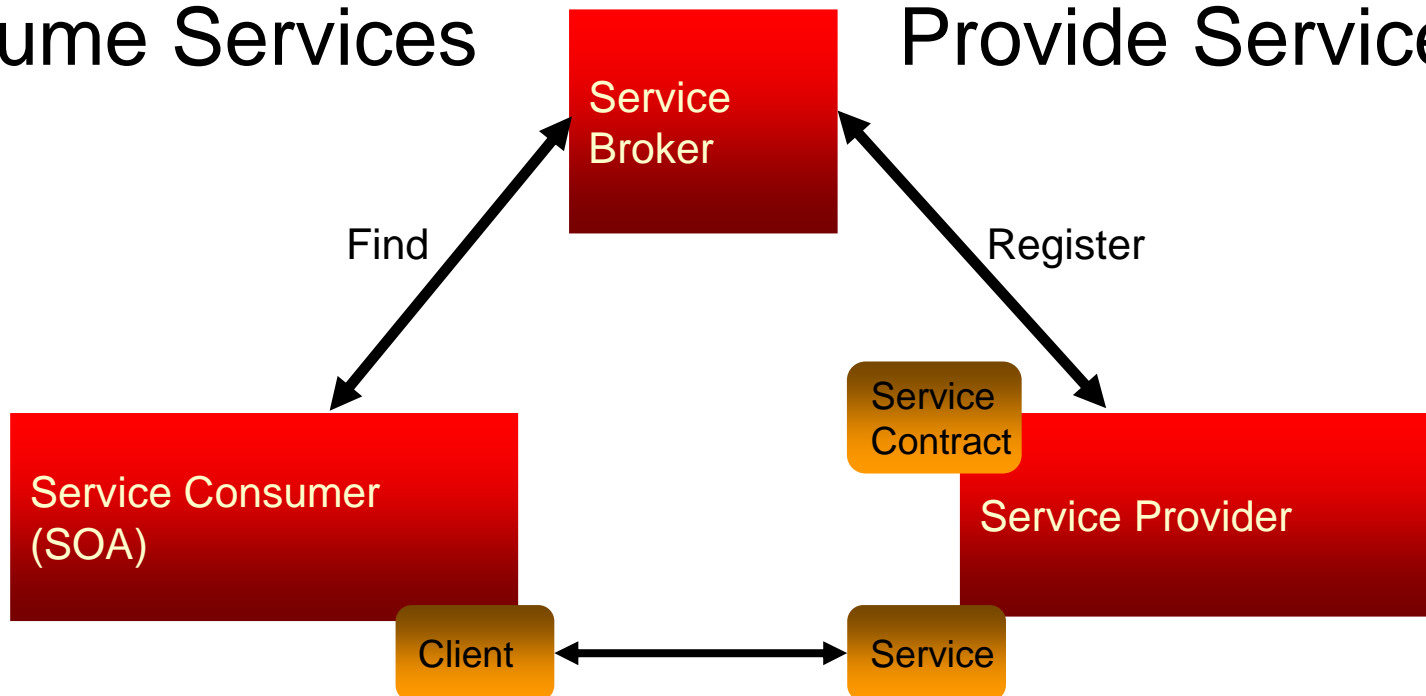
- Abstracted Re-Usable Interfaces
- Standards Based Web Services
- Documented Services & API

Ability to

Consume Services

Capability to

Provide Services





Case Study

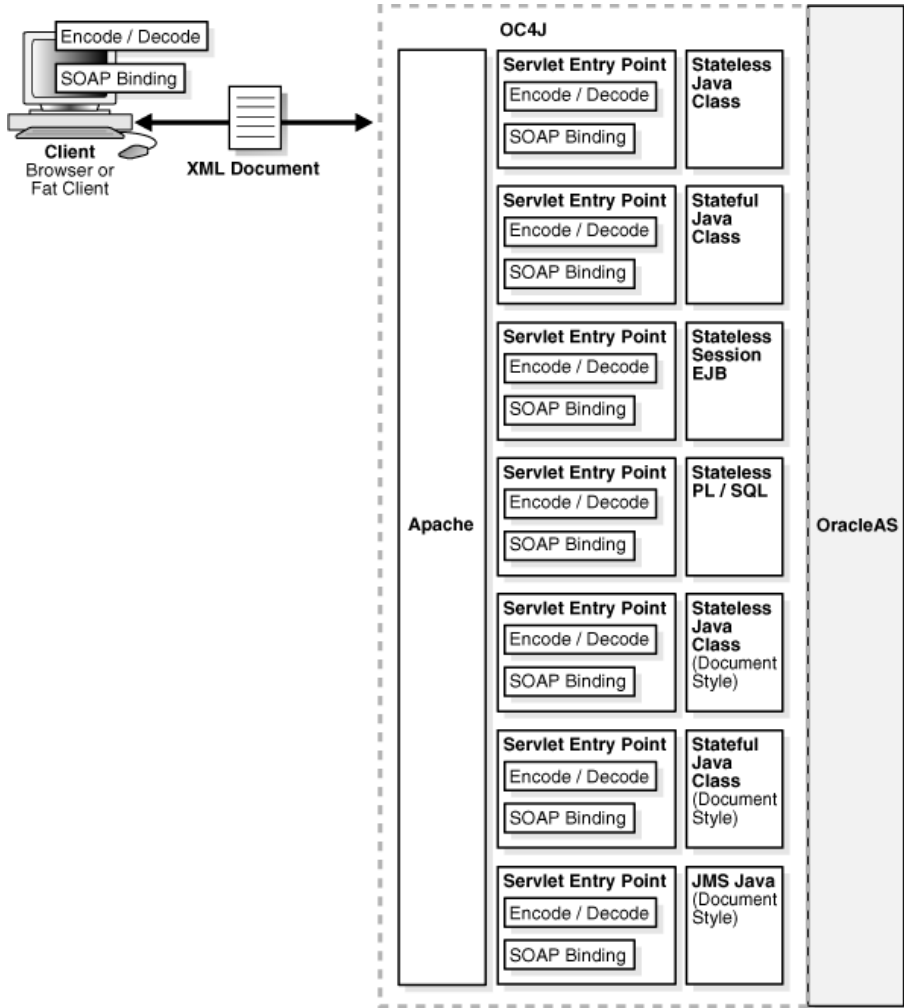
- External system is sending data by invoking our webservice – passing the inbound data and expecting a response.
- External system will call a generated webservice endpoint.
- Externally provided WSDL file
- (Web Service Definition Language)
- Top down vs Bottom up.



Why OC4J ?

- Old version of OAS running
- Multiple patches required
- Desire to isolate EBS from solution (do not want any patches to affect current EBS 11i version)
- Desire to have full control over web service processing.

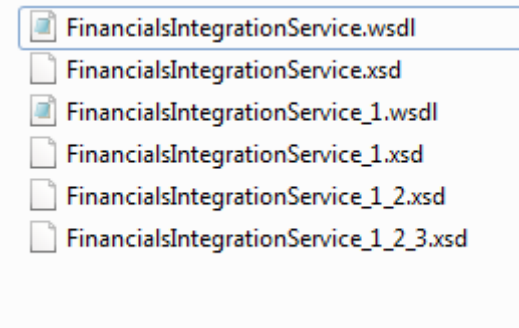
OAS / OC4J architecture



WSDL File

- Web Services Definition Language
- Operations

```
<wsdl:operation name="FindFinancialProject">  
<wsdl:input  
  wsaw:Action="http://www.datacom.co.nz/IRIS/Financials  
  /IFinancialsIntegrationService/FindFinancialProject"  
  message="tns:IFinancialsIntegrationService_FindFinanc  
  ialProject_InputMessage"/>  
</wsdl:operation>
```



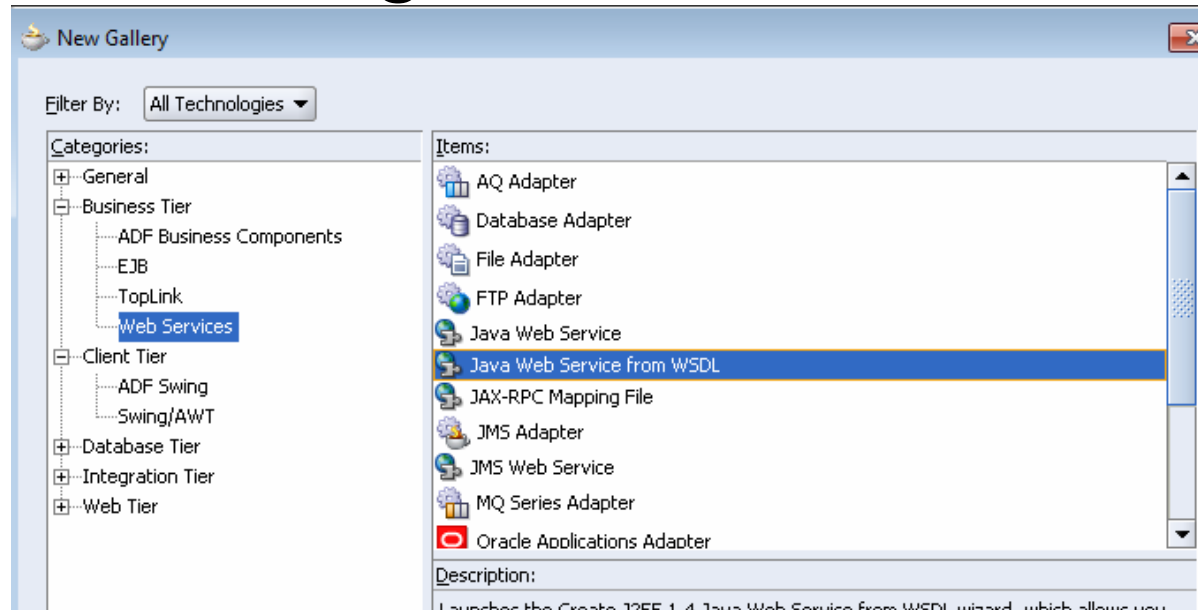


Create the Webservice

- Start up Jdeveloper (10.1.3)
- Create a new Application and Project
- Copy the wsdl and related files to the Project top level directory
- Choose File->New
- Web Services – > Java Web Service from WSDL

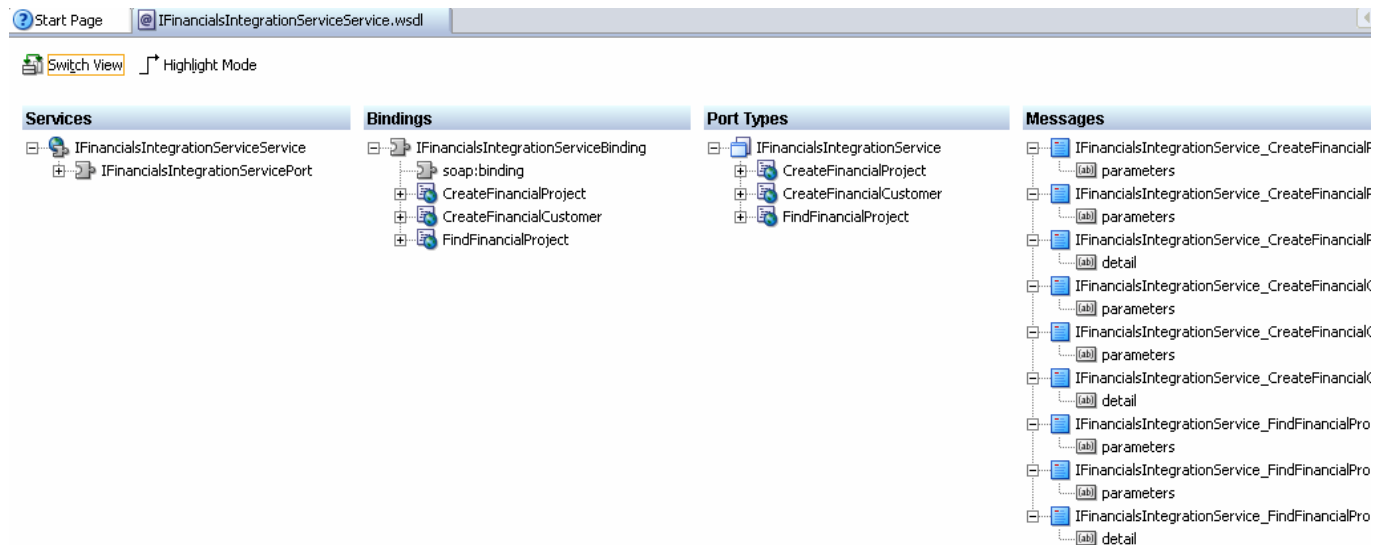
Create the Webservice

- Go through the wizard



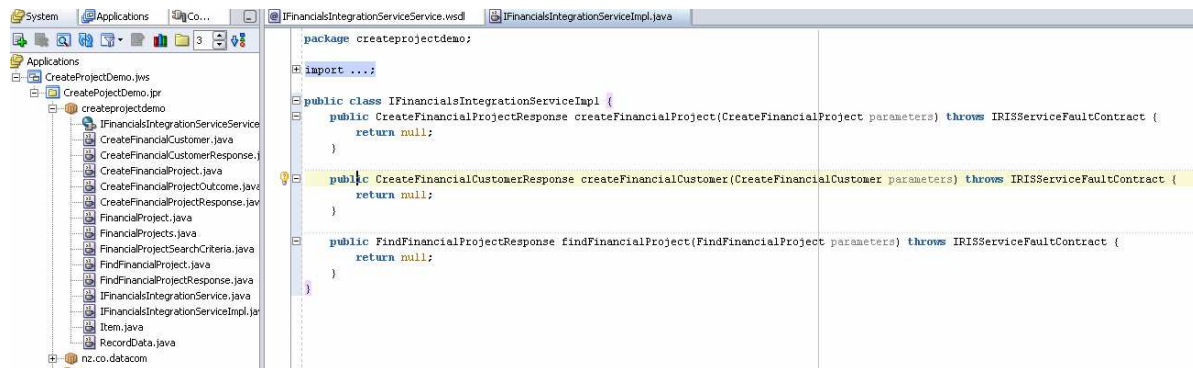
Create the Webservice

- Resulting web service skeleton is created:
- Visual view of the WSDL file



Create the Webservice

- Resulting web service skeleton is created:
- Java skeleton code created
- One method per operation from the WSDL file has been created



The screenshot shows an IDE window with a project structure on the left and Java code on the right. The project structure includes a package named 'createprojectdemo' containing several Java files, including 'IFinancialsIntegrationServiceService' and 'IFinancialsIntegrationServiceImpl.java'. The code in the right pane shows the skeleton for the 'IFinancialsIntegrationServiceImpl' class, with three methods: 'createFinancialProject', 'createFinancialCustomer', and 'findFinancialProject'. Each method is currently implemented with 'return null;' and throws 'IRISServiceFaultContract'.

```
package createprojectdemo;

import ...;

public class IFinancialsIntegrationServiceImpl {

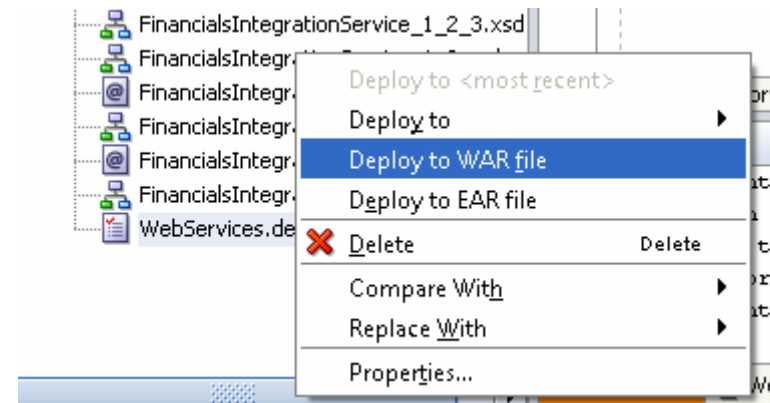
    public CreateFinancialProjectResponse createFinancialProject(CreateFinancialProject parameters) throws IRISServiceFaultContract {
        return null;
    }

    public CreateFinancialCustomerResponse createFinancialCustomer(CreateFinancialCustomer parameters) throws IRISServiceFaultContract {
        return null;
    }

    public FindFinancialProjectResponse findFinancialProject(FindFinancialProject parameters) throws IRISServiceFaultContract {
        return null;
    }
}
```

Deploy the Webservice

- You can setup to deploy direct to server if you wish
- Right click on WebServices.deploy – Deploy to WAR file





Deploy the Webservice

- In J2EE application modules are packaged as EAR, JAR and WAR based on their functionality
- JAR: EJB modules which contains enterprise java beans class files and EJB deployment descriptor are packed as JAR files with .jar extension
- WAR: Web modules which contains Servlet class files, JSP Files , supporting files, GIF and HTML files are packaged as JAR file with .war(web archive) extension
- EAR: All above files(.jar and .war) are packaged as JAR file with .ear (enterprise archive) extension and deployed into Application Server.

OC4J service

- List of Applications and Options to Deploy, Redeploy and Undeploy

ORACLE Enterprise Manager 10g
Application Server Control

OC4J: home Page Re

[Home](#) [Applications](#) [Web Services](#) [Performance](#) [Administration](#)

This page shows the J2EE applications and application components (EJB Modules, WAR Modules, Resource Adapter Modules) deployed to this OC4J instance.

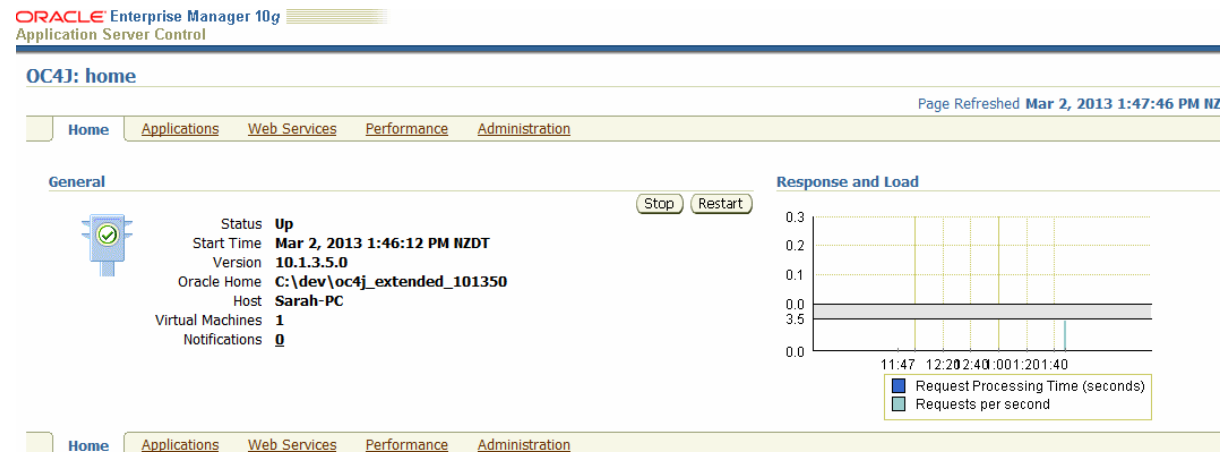
View

[Select All](#) | [Select None](#) | [Expand All](#) | [Collapse All](#)

Select	Name	Status	Start Time	Active Requests	Request Processing Time (seconds)	Active EJB Methods	Application Defined MBeans
<input type="checkbox"/>	▼ All Applications						
<input type="checkbox"/>	ascontrol	↑	Mar 2, 2013 1:46:14 PM NZDT	1	0.10	0	
<input type="checkbox"/>	▼ default	↑	Mar 2, 2013 1:46:14 PM NZDT	0	0.00	0	

OC4J service

- oc4j –start
- http://localhost:8888/





OC4J service

- Choose Deploy
- Need the context root:
- Context Root: A name that gets mapped to the document root of a Web application

OC4J service

- Choose Deploy
- Need the context root:
- Context Root: A name that gets mapped to the document root of a Web application

The screenshot shows the 'Application Attributes' tab in the OC4J deployment console. At the top, there are three tabs: 'Select Archive', 'Application Attributes', and 'Deployment Settings'. Below the tabs, the 'Deploy: Application Attributes' section is active. It displays the following information:

- Archive Type: **Web Module (WAR file)**
- Archive Location: **WebServices.war**
- Deployment Plan: **Creating a new plan**

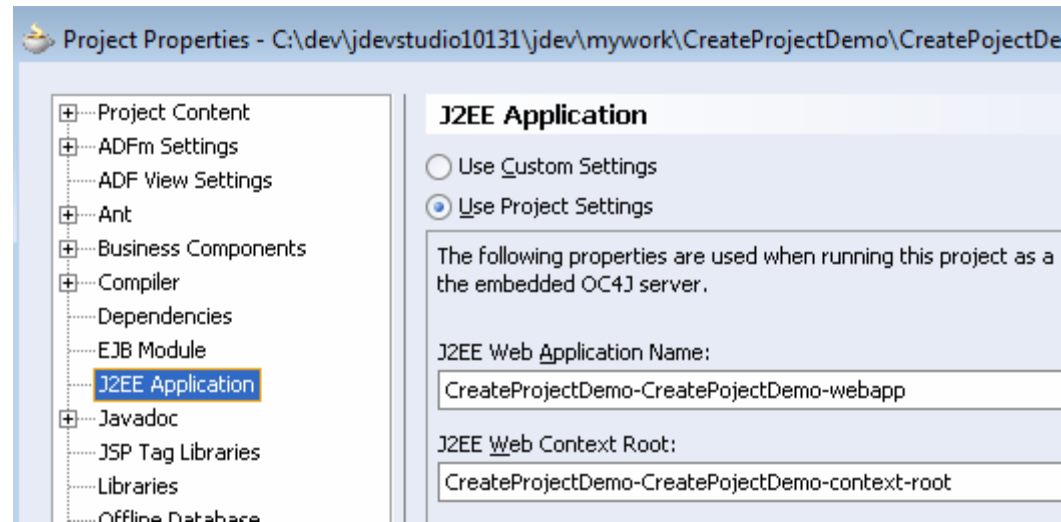
Below this information, there are several configuration fields:

- * Application Name:
- Parent Application:
- Bind Web Module to Site:
- Context Root: A table with two columns: 'Web Module' and 'Context Root'.

Web Module	Context Root
WebServices.war	ProjectDemo-context-root

OC4J service

- Context Root - Jdeveloper: Right click on Project Properties



Web Service Deployment

- The exciting bit...

Confirmation

The Application "CreateProjectDemo" has been successfully deployed.

[Return](#)

Progress Messages

```
[2/03/2013 2:24:28 PM] Copy the archive to C:\dev\oc4j_extended_101350\j2ee\home\applications\CreateProjectDemo.ear
[2/03/2013 2:24:28 PM] Initialize C:\dev\oc4j_extended_101350\j2ee\home\applications\CreateProjectDemo.ear begins...
[2/03/2013 2:24:28 PM] Unpacking CreateProjectDemo.ear
[2/03/2013 2:24:28 PM] Done unpacking CreateProjectDemo.ear
[2/03/2013 2:24:28 PM] Unpacking WebServices.war
[2/03/2013 2:24:28 PM] Done unpacking WebServices.war
[2/03/2013 2:24:28 PM] Initialize C:\dev\oc4j_extended_101350\j2ee\home\applications\CreateProjectDemo.ear ends...
[2/03/2013 2:24:28 PM] Starting application : CreateProjectDemo
[2/03/2013 2:24:28 PM] Initializing ClassLoader(s)
[2/03/2013 2:24:28 PM] Initializing EJB container
[2/03/2013 2:24:28 PM] Loading connector(s)
[2/03/2013 2:24:28 PM] Starting up resource adapters
[2/03/2013 2:24:28 PM] Initializing EJB sessions
[2/03/2013 2:24:28 PM] Committing ClassLoader(s)
[2/03/2013 2:24:28 PM] Initialize WebServices begins...
[2/03/2013 2:24:28 PM] Initialize WebServices ends...
[2/03/2013 2:24:28 PM] Started application : CreateProjectDemo
[2/03/2013 2:24:28 PM] Binding web application(s) to site default-web-site begins...
[2/03/2013 2:24:28 PM] Binding WebServices web-module for application CreateProjectDemo to site default-web-site under context root CreateProjectDemo-CreateProjectDemo-context-root
[2/03/2013 2:24:33 PM] Initializing Servlet: oracle.j2ee.ws.server.JAXRPCServlet for web application WebServices
[2/03/2013 2:24:33 PM] Binding web application(s) to site default-web-site ends...
[2/03/2013 2:24:33 PM] Application Deployer for CreateProjectDemo COMPLETES. Operation time: 5306 msec
```

[Return](#)

Web Service Testing

ORACLE Enterprise Manager 10g
Application Server Control

OC4J: home

[Home](#) [Applications](#) [Web Services](#) [Performance](#) [Administration](#)

Application All

[Test Service](#)

Select	Port Name <small>▲</small>	Web Service	Application	Application Status	Port Enabled	Start Time
<input checked="" type="radio"/>	IFinancialsIntegrationServicePort	IFinancialsIntegrationServiceService	CreateProjectDemo	↑	✓	Mar 2, 2013 2:24:33 PM NZDT

Web Service testing

- Enter data through the form or raw XML.
- Click Invoke (not shown)

IFinancialIntegrationServiceService endpoint

For a formal definition, please review the [Service Description](#).

Download the JavaScript Stub (BETA) for [IFinancialIntegrationServicePort](#) and see its [documentation](#).

IFinancialIntegrationServicePort

Operation : CreateFinancialProject HTML Form XML Source

Reliable Messaging CreateFinancialProject
 CreateFinancialCustomer
 FindFinancialProject

WS-Security Include In Header

parameters

RecordData Include In Message

FINCustomerCode	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message
FINProjectCode	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message
IrisID	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message
ObjectType	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message
OfficerResponsible	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message
Subclassification1	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message
Subclassification2	<input type="text"/>	xsd:string	<input checked="" type="checkbox"/> Include In Message

Web Service testing

- Results
- The not so exciting part as it has done nothing.

Test Result

View: [Formatted XML](#) | [Raw XML](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://www.datacom.co.nz/IRIS/" xmlns:ns1="http://www.datacom.co.nz/IRIS/Financials/"
xmlns:ns2="http://www.datacom.co.nz/IRIS/Contacts/"><env:Body><ns1:CreateFinancialProjectResponse xsi:nil="1"/></env:Body></env:Envelope>
```



Web Service code

- Combination Java and PL/SQL
- Add some code to our Impl.java file.
- Could do some jdbc calls to Oracle to pass the data to pl/sql for further processing.
- Could use java to manipulate the incoming data.



Web Service code

- Send a response back
- Our current Java stub created from the wizard

```
public CreateFinancialProjectResponse  
createFinancialProject(CreateFinancialProject parameters) throws  
IRISServiceFaultContract {  
    return null;  
}
```



Web Service code

- Reponse code added

```
public CreateFinancialProjectResponse  
createFinancialProject(CreateFinancialProject parameter    rs) throws  
IRISServiceFaultContract {
```

```
    CreateFinancialProjectResponse response ;  
    response = new CreateFinancialProjectResponse(); //initialise it
```

```
    CreateFinancialProjectOutcome outcome;  
    outcome = new CreateFinancialProjectOutcome(); //intialise it
```

```
    //set the values  
    outcome.setErrorMessage("ALL WAS GREAT");  
    outcome.setSuccess(true);
```

```
    response.setCreateFinancialProjectResult(outcome);  
    return response;
```

```
}
```


Web Service code

- Deploy and test it

```
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns0="http://www.datacom.co.nz/IRIS/"
  xmlns:ns1="http://www.datacom.co.nz/IRIS/Financials/"
  xmlns:ns2="http://www.datacom.co.nz/IRIS/Contacts/">
  <env:Body>
    <ns1:CreateFinancialProjectResponse>
      <ns1:CreateFinancialProjectResult>
        <ns1:ErrorMessage>ALL WAS GREAT</ns1:ErrorMessage>
        <ns1:Success>>true</ns1:Success>
      </ns1:CreateFinancialProjectResult>
    </ns1:CreateFinancialProjectResponse>
  </env:Body>
</env:Envelope>
```