

Access IBM Workplace Collaboration Services using Web services

Skill Level: Intermediate

[Rohit Sahasrabudhe \(sahas@us.ibm.com\)](mailto:sahas@us.ibm.com)
Architect
EMC

Nov 2005

IBM Workplace Collaboration Services (IWCS) provides collaborative services that can be used in your applications. These collaborative services are available to you via Application Programming Interfaces (APIs) and Service Provider Interfaces (SPIs). A majority of these services are also available via Web Services. In this tutorial you will explore the steps required to access these collaborative services using Rational Application Developer to build your client applications. Two examples show you how to access two separate services exposed by IWCS server.

Section 1. Before you start

IBM® Workplace™ products provide the front-end to IBM's service oriented architecture (SOA) strategy. The IBM Workplace Collaboration Services API Toolkit allows developers to extend the IBM Workplace Collaboration Services platform with new business components, portlets, and other components using the J2EE programming model. The toolkit provides public APIs (Application Programming Interfaces) and SPIs (Service Provider Interfaces) that you can use in your applications. In this tutorial, you explore how to access these extensions via Web Services.

About this tutorial

In this tutorial you explore the Web services made available with IBM Workplace Collaboration Services API Toolkit. First, take a look at a simple Web service

sample, included with the toolkit. Then, take a look at another sample based on the Application Web Service.

Objectives

At the completion of this tutorial, you will know how to use Rational® Application Developer to access Web services exposed by the IBM Workplace Collaboration Services Server. Going through this tutorial won't make you an expert in all the Web services provided, but it will give you some background. You can then apply this knowledge with the details in the toolkit documentation to take advantage of these Web services in your applications. The toolkit comes with good documentation that includes both overview and in-depth descriptions. In this tutorial you will also learn how to use Rational Application Developer to minimize your coding efforts by using the Web Services Wizards to generate most of the code required to access these Web services.

Intended audience

This tutorial is written for developers who want to understand how to use Web services to simplify accessing the extension point of the IBM Workplace Collaboration Services. Users should have a basic background in Web services and the Java™ Technology.

It is also recommended that you look through the toolkit documentation to see the different types of extensions that are available through this toolkit. A general understanding of the toolkit will help you gain the most out of this tutorial.

Software requirements

To complete the steps in this tutorial, you will need to have the following software installed:

- IBM Workplace Collaboration Services V2.5.1 Server
- The IBM Workplace Collaboration Services [API Toolkit](#). Install the toolkit over your IBM Workplace Collaboration Services v2.5.1 Server. Follow the installation steps provided in the toolkit documentation. To obtain the IBM Workplace Collaboration Services API Toolkit, click the link above to display a listing of the IBM Workplace API toolkits available for download. Click "Tool: IBM Workplace Collaboration Services API Toolkit 2.5.1 Dev Files and Documentation for Windows and Linux" and follow the instructions provided to download the toolkit zip file.

- Rational Application Developer 6.0.0.1 with fixpacks [001](#), [002](#), and [003](#) with WebSphere Appserver 5.0 Test Environment.
A trial version of [IBM Rational Application Developer](#) is available to download.

You also need to download and unzip the sample code, [WebServicesClient.zip](#), included with this tutorial. This file contains java code needed later in the tutorial. Remember the location where you unzipped the files so you can access them later.

Section 2. Creating a simple Web services sample

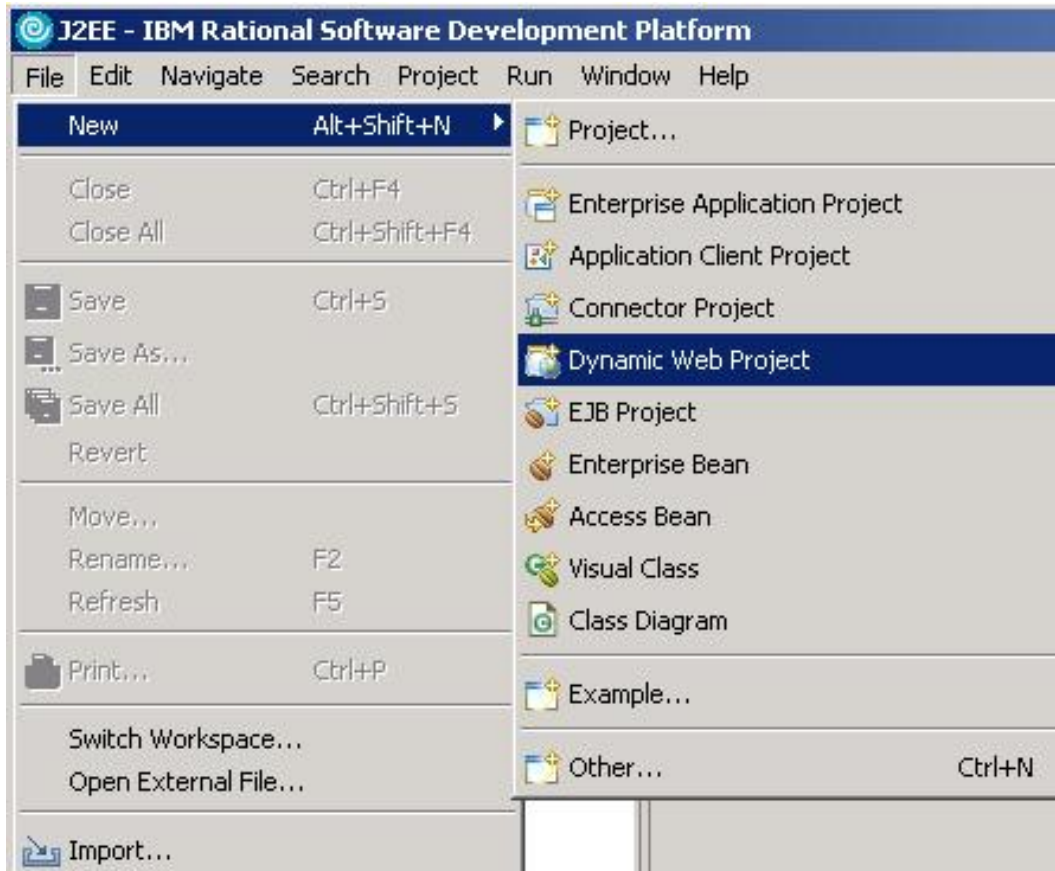
In your first example, create a Web service client proxy to execute methods exposed by the WebConference Web Service. Start by creating a workspace within Rational Application Developer (RAD) to hold your "Web service client" artifacts. Next, generate the client proxy and setup the required user credentials to access the Web service. At the end of this example, test your client to verify that the methods are executed correctly by accessing them on your own IWCS Server.

Create a Dynamic Web project

In this part, use Rational Application Developer's Web Service Client Wizard to connect to a WSDL (Web Services Definition Language) file. This file is used to describe the input and output parameters and methods provided by a particular Web service.

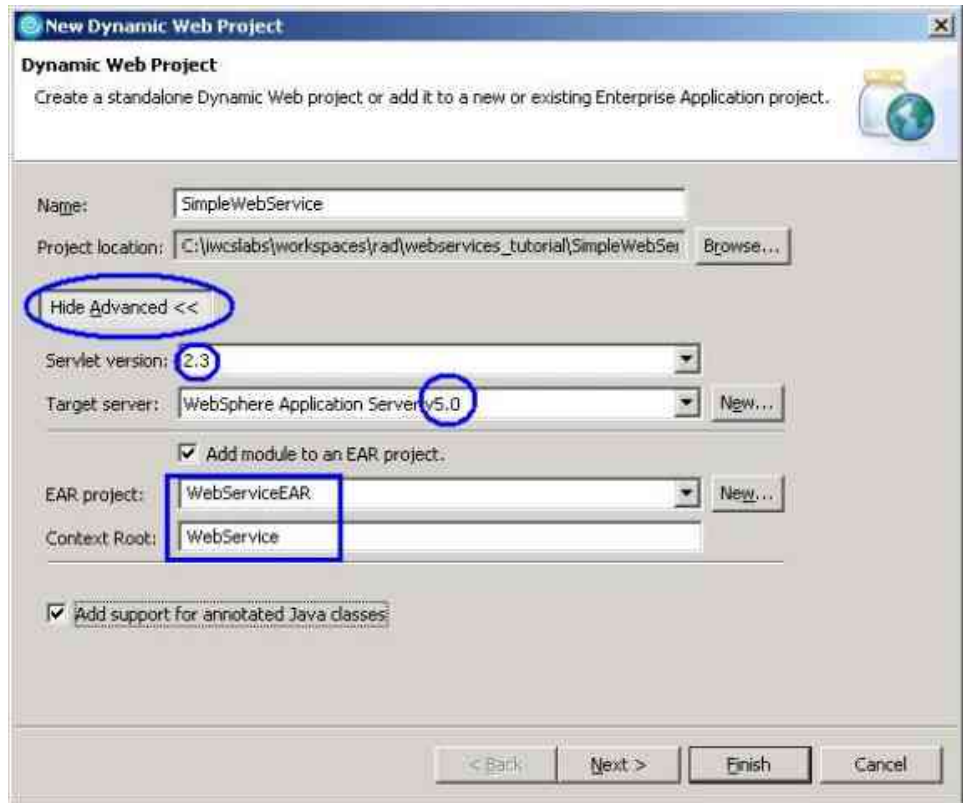
First, create a project to hold your Web services client.

1. Start Rational Application Developer.
 1. From the Windows Start menu select **Start Programs > IBM Rational > IBM Rational Application Developer V6.0 > Rational Application Developer**.
 2. A window prompts you to select a location for your Workspace. Use the default location.
 3. Click **OK**.
2. Click **File > New > Dynamic Web Project**.
Figure 01. Create a Dynamic Web Project



3. In the Name: field, enter SimpleWebService.
4. Click the Show Advanced >> icon.
 1. Enter 2.3 for Servlet version.
 2. Select **WebSphere Application Server v5.0** for Target Server.
Note: You have to select the Target Server to be WebSphere Application Server v 5.0, as IBM Workplace Collaboration Services 2.5.1 is running on top of WebSphere Application Server 5.0.2.6.
 3. Select **WebServiceEAR** for the EAR project.
 4. Enter `WebService` for Context Root.

Figure 02. New Dynamic Web Project Properties



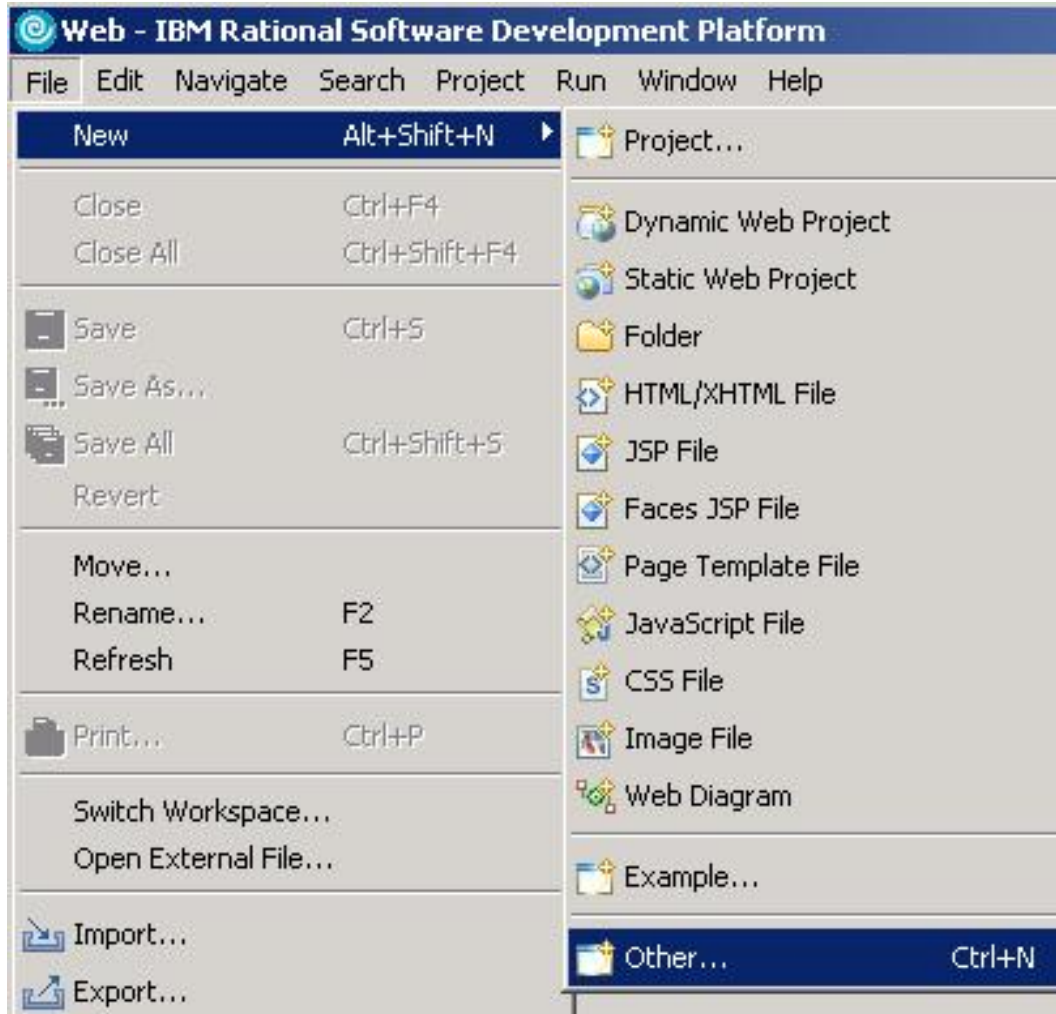
5. Click **Finish** to create the Dynamic Web Project.

Create Web service client proxy

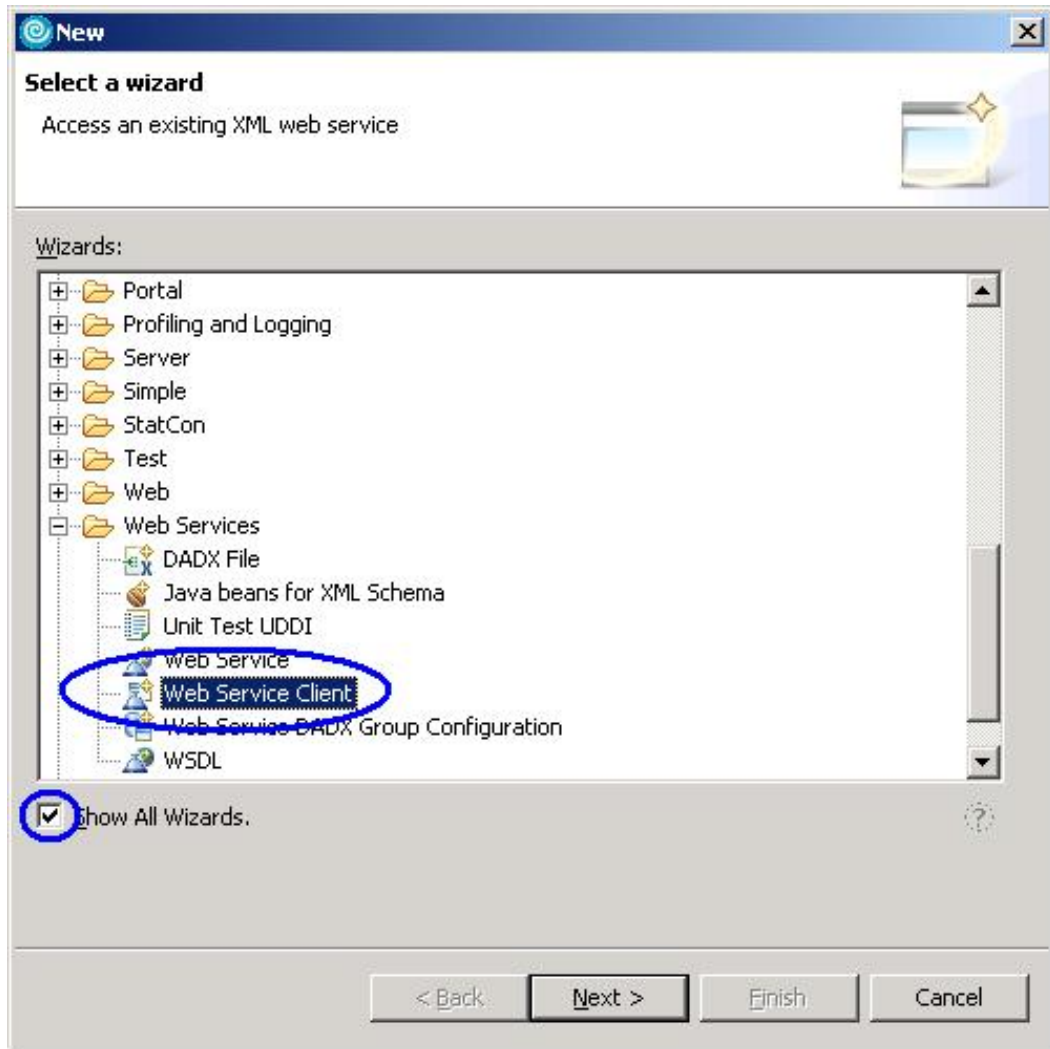
When prompted, select **Yes** to switch to the Web Perspective.

Now, create your Web service client proxy.

1. Click **File > New > Other**.
Figure 03. Create a new Web service client proxy



2. Select the check box to Show All Wizards.
3. Scroll down and expand **Web Services** and select **Web Services Client**.
Figure 04. Select the Web Service Client option

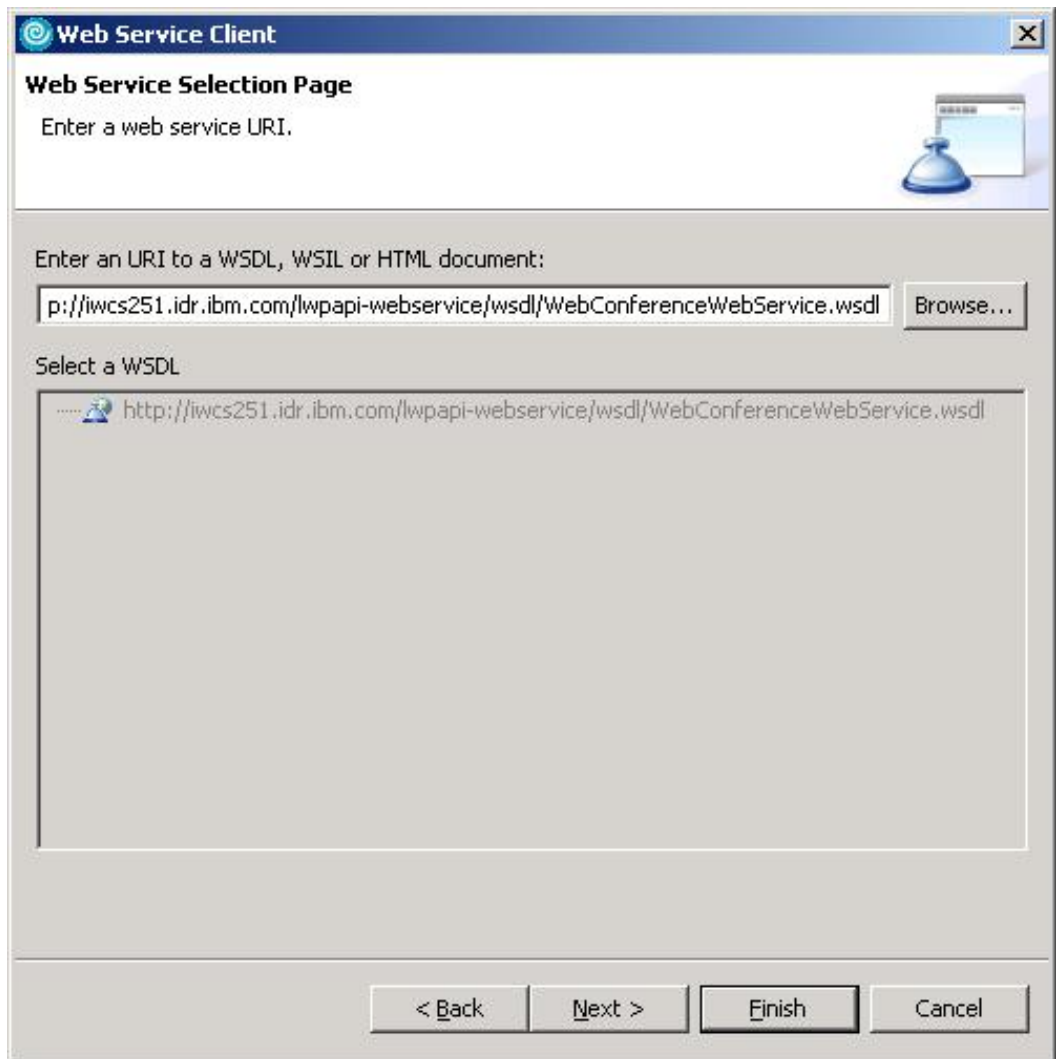


4. Click **Next**.
5. If prompted to enable Web Services Development capability, select **OK**.
6. Accept the defaults on the Web Services window and click **Next**.
7. Enter the following URL:

```
http://FullyQualified.Domain.Name/lwpapi-webservice/wsd/ConferenceWebService.wsdl
```

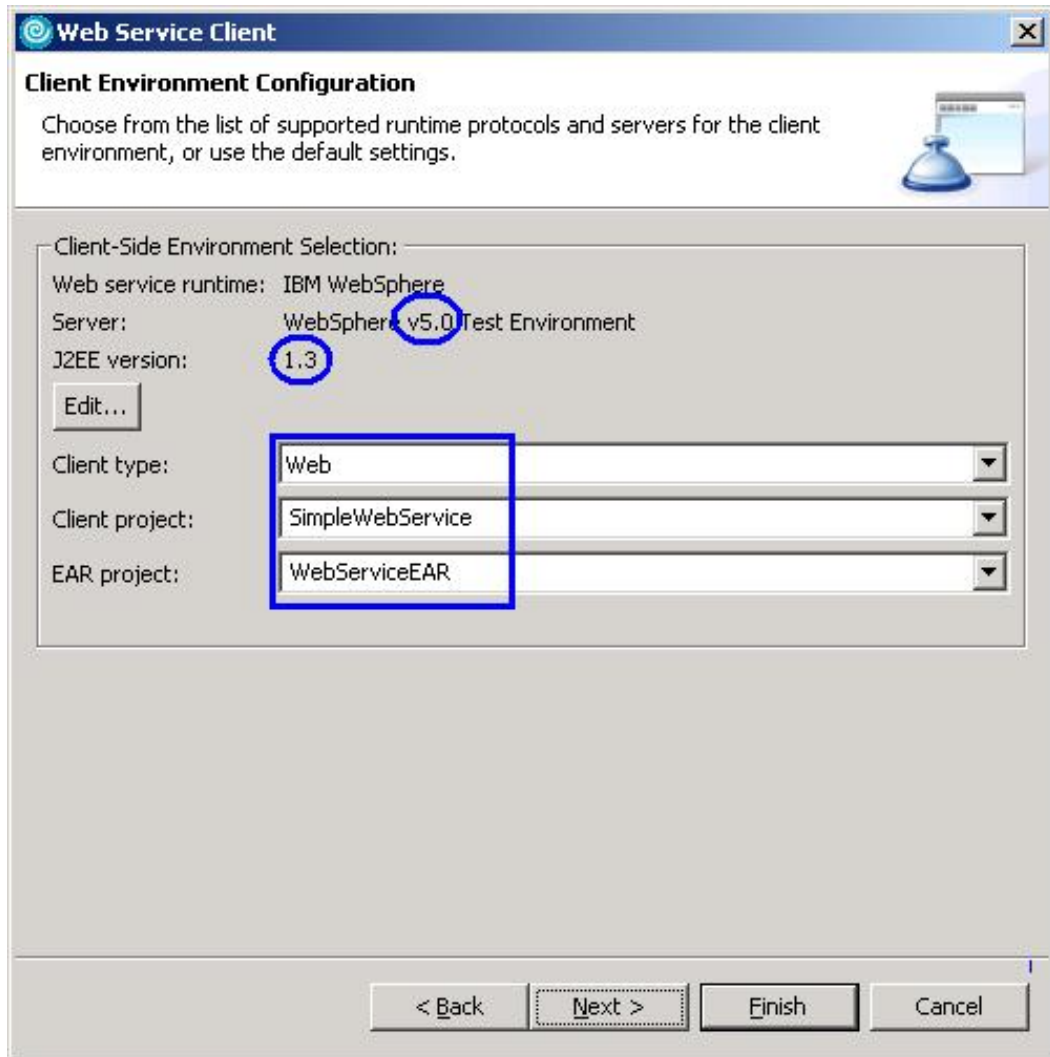
NOTE: FullyQualified.Domain.Name needs to be replaced by your environment. For this tutorial, the author used "iwcs251.idr.ibm.com". The URI is case sensitive, note the WSDL file name above.

Figure 05. Web Service URI



8. Click **Next** .
Note: If you get an error, double check the URI.
9. On the Client Environment Configuration window, make sure the following fields are pre-selected:

Figure 06. Client Environment Configuration

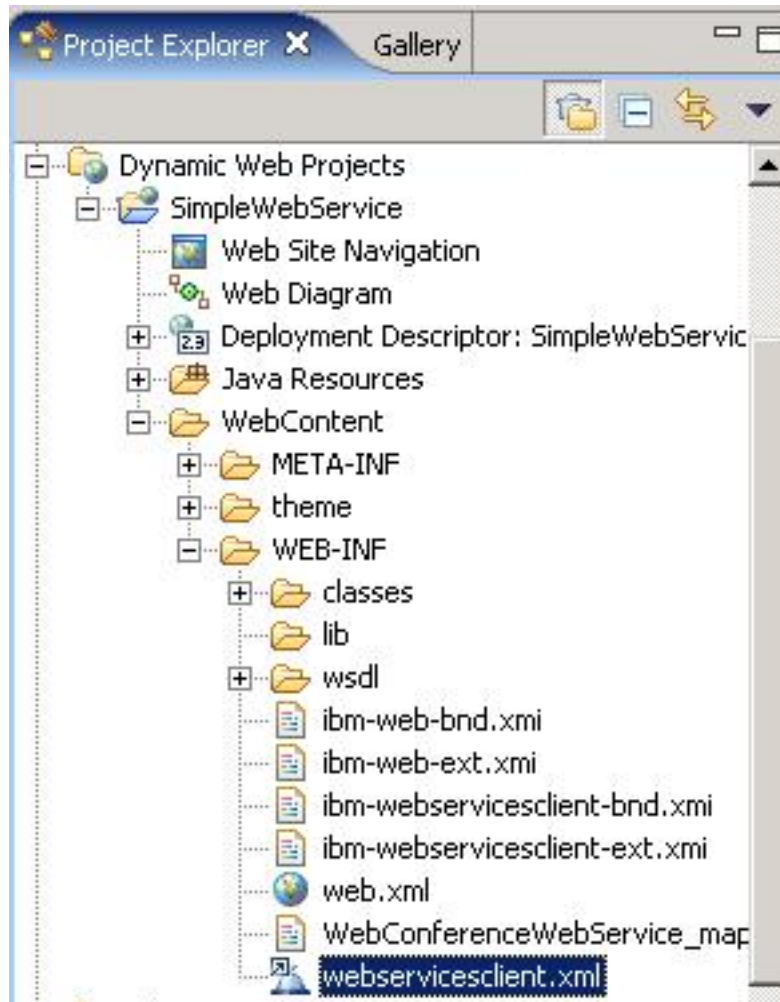


10. Click **Next**.
11. Click **Finish** on the Web Service Proxy Page.

Create Web service client proxy

Next, setup the user credential information for the simple Web service client.

1. Using the Project Explorer, locate **webserviceclient.xml**.
2. Expand to **Dynamic Web Projects > SimpleWebService > WebContent > WEB-INF > webservicessclient.xml**.
Figure 07 Open webservicessclient.xml file

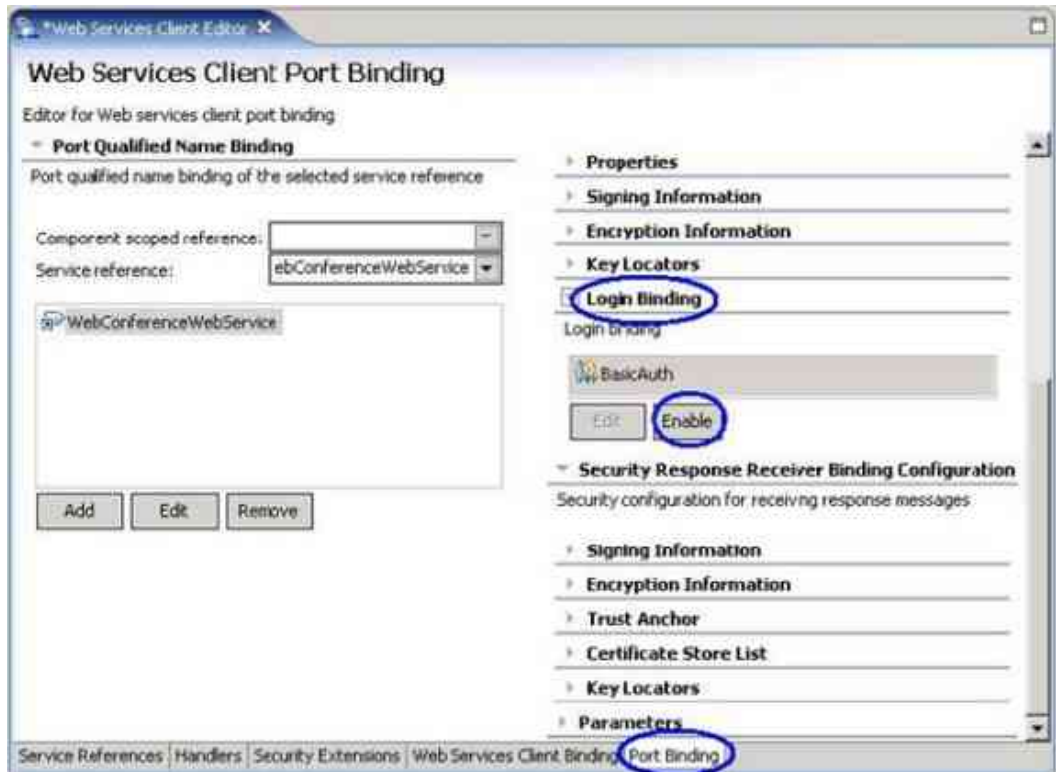


3. Double-click **webservicescient.xml**.
 4. Click the **Security Extensions** bottom tab.
 5. Expand the **Login Config** section.
 6. Select **BasicAuth** for Authentication method.
- Figure 08. Security Extensions, select Login Config**



7. Click the **Port Binding** bottom tab.
8. Expand the **Login Binding**.
9. Click **Enable**.

Figure 09. Enable Login Binding



10. Select **com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler** for Callback handler.
 11. Enter the IWCS Administrator's ID and password.
Example: wpsadmin / wpsadmin
 12. Click **OK**.
- Figure 10. Select the callback handler**

Login binding dialog

Authentication method: BasicAuth

Token value type:

URI:

Local name:

Callback handler: com.ibm.wsspi.wssecurity.auth.callback.NonPromp

Basic authentication:

User ID: wpsadmin

Password: *****

Property:

Name	Value

Add Remove

OK Cancel

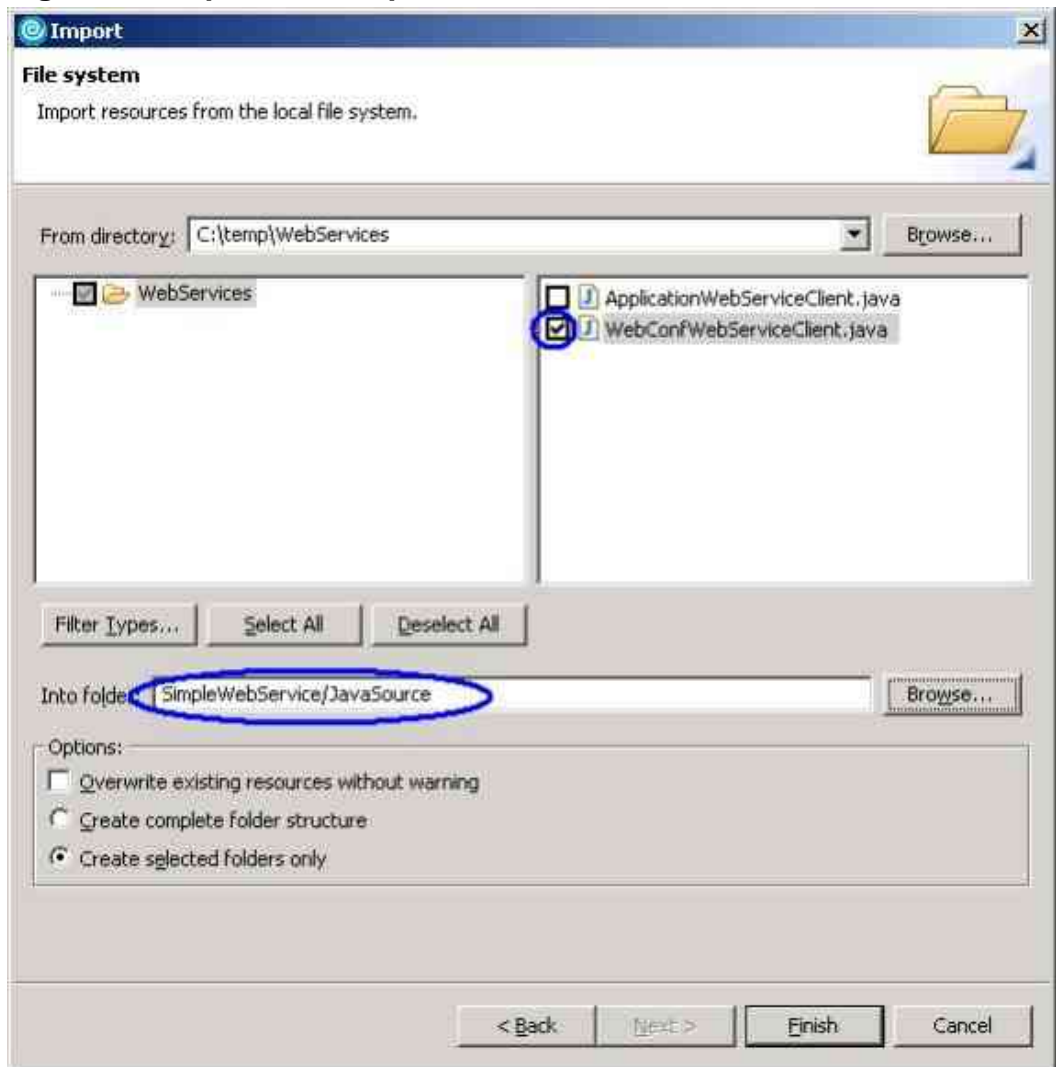
13. Save the **webservicesclient.xml** file by pressing CTRL + S.

Exploring a Java Web service client

To import in a simple Java client for your Web service:

1. Select **File > Import**.
2. Select **File System** as an import source and click **Next**.

3. Click **Browse** to locate the From directory.
 4. Navigate to the WebServices folder where you unzipped the WebServicesClient.zip file and click **OK**.
 5. Select **WebConfWebServiceClient.java**.
 6. Click **Browse** to locate the Into folder.
 7. Expand **SimpleWebService**, select the JavaSource folder and click **OK**.
- Figure 11. Import the simple Java client**



8. Click **Finish**.
9. Expand to **Dynamic Web Projects > SimpleWebService > Java**

Resources > Java Source > (default package) > WebConfWebServiceClient.java.

10. Double-click **WebConfWebServiceClient.java**.
11. Make sure that the string **endpoint_server** is set to:

```
FullyQualified.Domain.Name
```

Note: For the tutorial, the author is using `iwcs251.idr.ibm.com`.

12. The following lines of code find the WebConferenceWebService using the JNDI Lookup. You need a handle to this service to invoke methods on it.

```
Context ctx = new InitialContext();
String webService = "java:comp/env/service/WebConferenceWebService";
WebConferenceWebService service = (WebConferenceWebService)
    ctx.lookup(webService);
```

13. Once you have the WebService handle, you can create your proxy object that can be used to invoke methods.

```
WebConferenceWebService_SEI proxy =
    service.getWebConferenceWebService(new URL(endpoint));
```

14. In this simple example, you are using the Web Conferencing Web service, to retrieve the Build and Release information; plus, the boolean - `isAvailable`.

```
boolean b = proxy.isAvailable();
out.write("isAvailable= " + b + "<br>");

WorkplaceVersionElement wve = proxy.getWorkplaceServerVersion();
out.write("build= " + wve.getBuild() + "<br>");
out.write("release= " + wve.getRelease() + "<br>");
```

This Web service client code doesn't utilize the "Web Conferencing" service. You are using this particular Web Conference WSDL file because there is no generic WSDL. If you would like to gain additional information on what this particular and other Web services offer, refer to the API documentation.

Testing your simple Web service client

Now that you have a good understanding of how to create the proxy code and have

explored a simple Java implementation of a Web service client, test it.

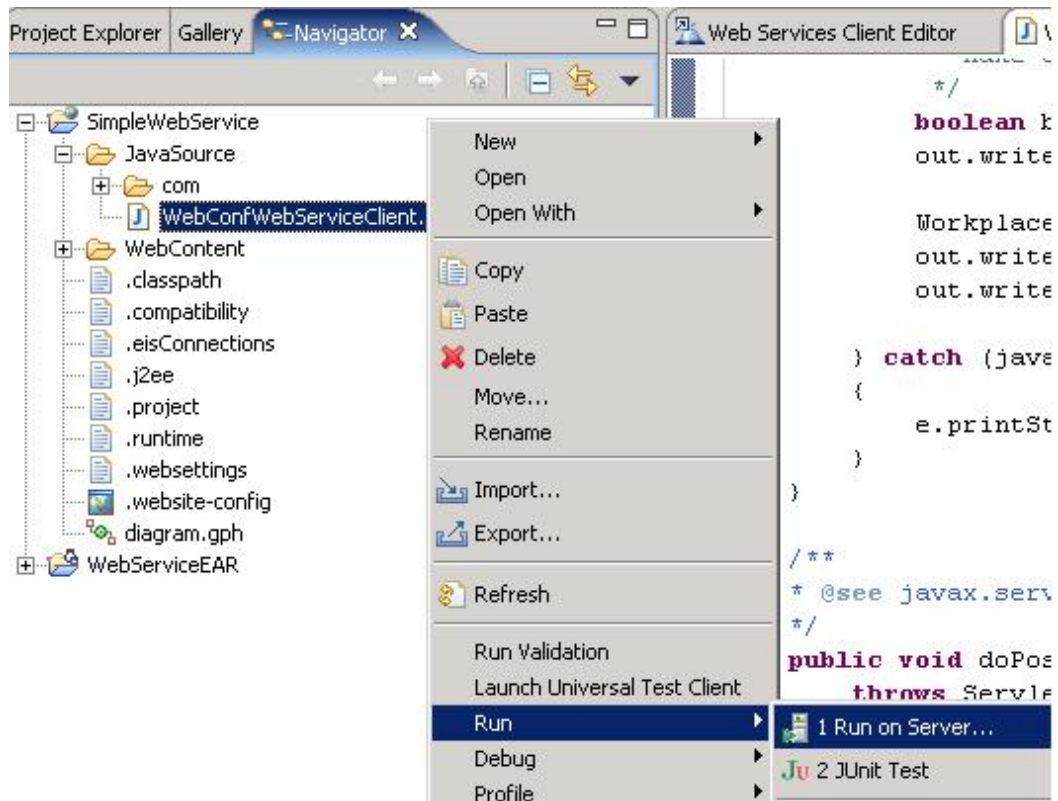
1. Make sure that the IWCS Server you plan to test against is running.
2. Now make sure that none of the Test Environments within Rational Application Developer are running. Click the Servers tab.
3. The **WebSphere v5.0 Test Environment @ localhost** should be running. If it is, stop it by right-clicking on the Started status and select **Stop**.
4. All the Test Environments should be in the Stopped state.

Figure 12. Test Environment status

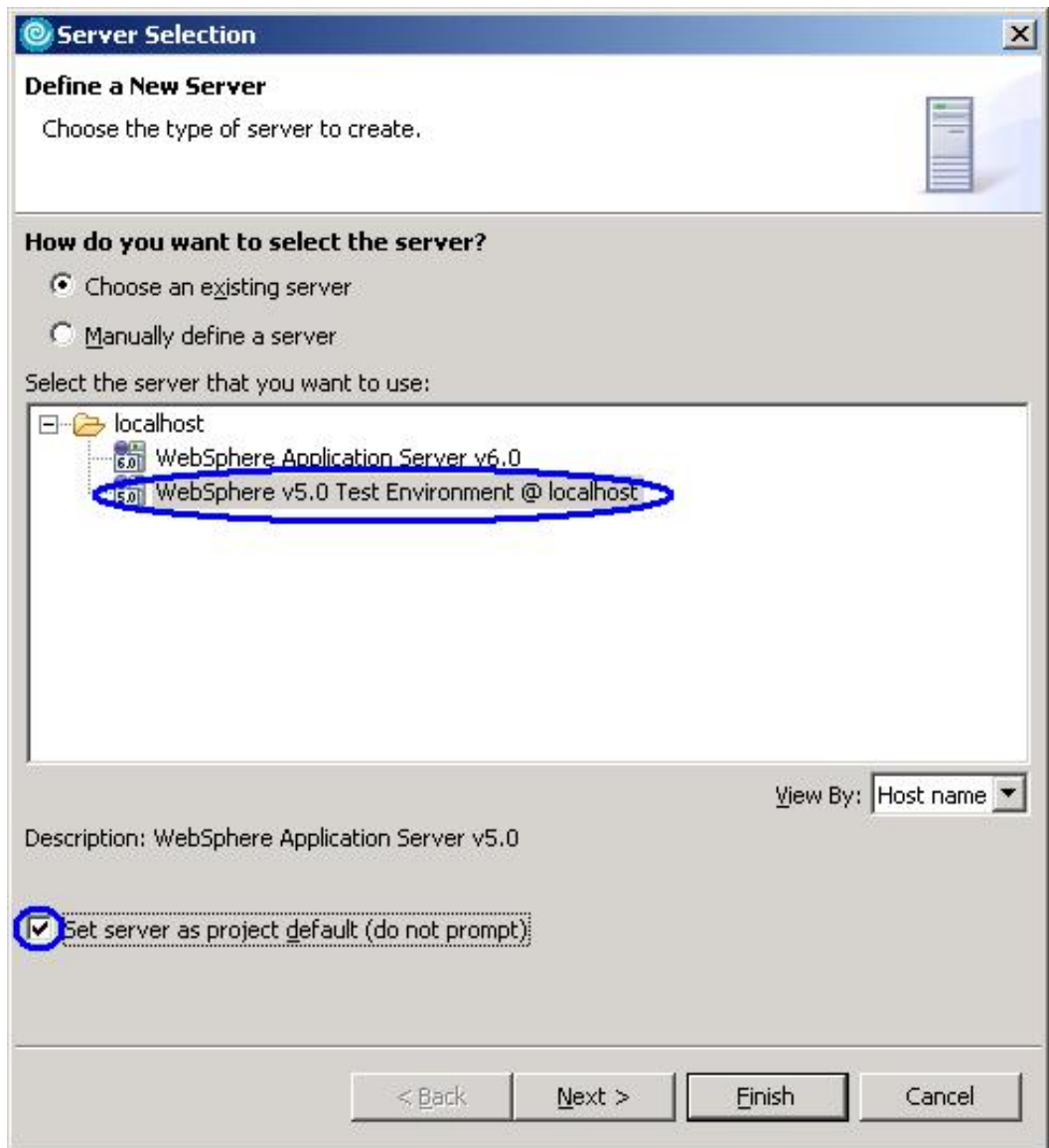


5. Click **Window > Show View > Navigator**. This adds the Navigator view to your workspace.
6. Expand to **SimpleWebService > Java Resources > WebConfWebServiceClient.java**.
7. Right-click **WebConfWebServiceClient.java** and select **Run > Run on Server**.

Figure 13. Run the Web service client



8. On the Server selections window, select the **WebSphere v5.0 Test Environment @ localhost**.
9. Check **Set server as project default (do not prompt)**. **Figure 14. Select the Test Environment**



10. Click **Finish**.
The Test Environment starts and displays the output in a Web Browser launched within Rational Application Developer.

Figure 15. Simple Web service Result



11. Now, stop the Server so you can move on to the next example. Click the **Server** bottom tab.
12. Right-click **Started WebSphere v5.0 Test Environment** and select **Stop**.
13. Wait until the server status changes to Stopped.

Summary

In this example, you built a simple Web service client to talk to the Web Conferencing Web service. You invoked the following three methods: `isAvailable`, `build` and the `release` information. As you can see in the results above, the methods were executed on the server and their responses are displayed to the client.

Section 3. Creating an application Web service

In your second example, create another Web service client proxy. The steps required to setup the initial client proxy and user credentials are the same. Except, in this example you will be connecting to the application Web service interface. This service, allows you to get access to the infrastructure level services, more specifically to the applications deployed on the IWCS Server.

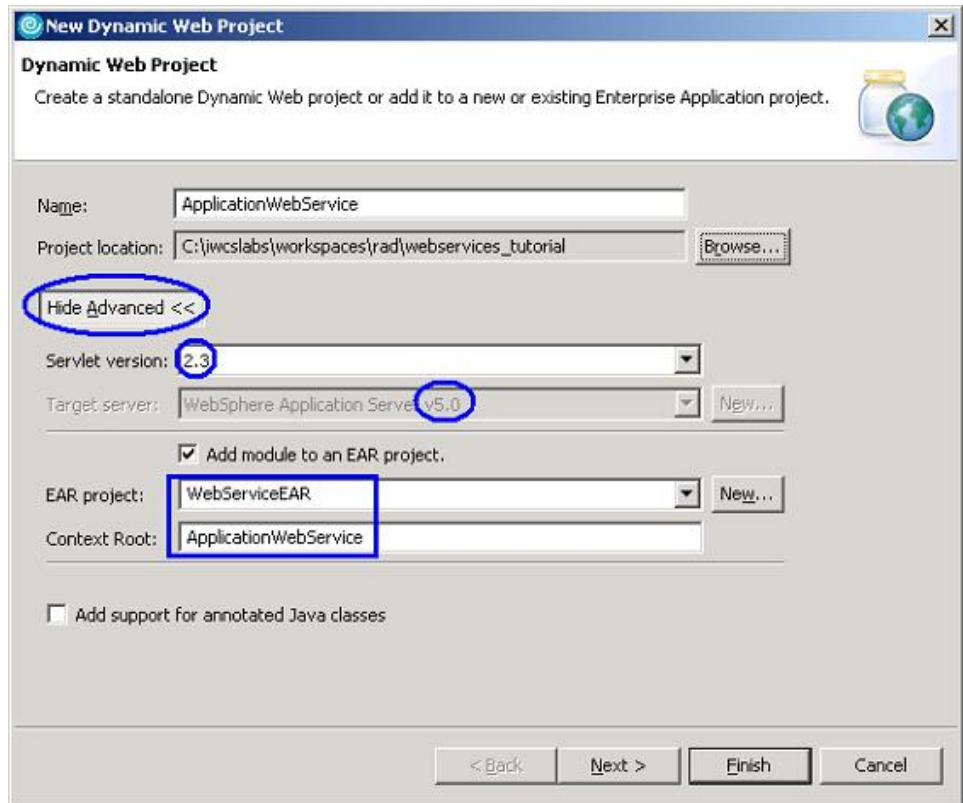
Create a dynamic Web project

In this part, use Rational Application Developer's Web Service Client Wizard to connect to a WSDL (Web Services Definition Language) file. This file is used to describe the input and output parameters and methods provided by a particular Web service.

First you need to create a project that holds your Web services client.

1. Start Rational Application Developer, if it's not running:
 1. From the Windows Start menu select **Start > Programs > IBM Rational > IBM Rational Application Developer V6.0 > Rational Application Developer**.
 2. A window prompts you to select a location for your Workspace. Use the default location.
 3. Click **OK**.
2. Click **File > New > Dynamic Web Project**.
3. Enter `ApplicationWebService` for Name.
4. Click the Show Advanced >> icon.
 1. Select `2.3` for Servlet version.
 2. Select `WebSphere Application Server v5.0` for the Target Server.
Note: You have to select the Target Server to be WebSphere Application Server v 5.0, because IBM Workplace Collaboration Services 2.5.1 is running on top of WebSphere Application Server 5.0.2.6.
 3. Select `WebServiceEAR` for the EAR project.
Note: As soon as you select `WebServiceEAR` as the EAR project, the WebSphere Application Server V5.0 selection will be grayed out. This is because you had selected the v5.0 release in the previous section where you originally built the `WebServiceEAR` project.
 4. Select `ApplicationWebService` for the Context Root.

Figure 16. New Application Web Service

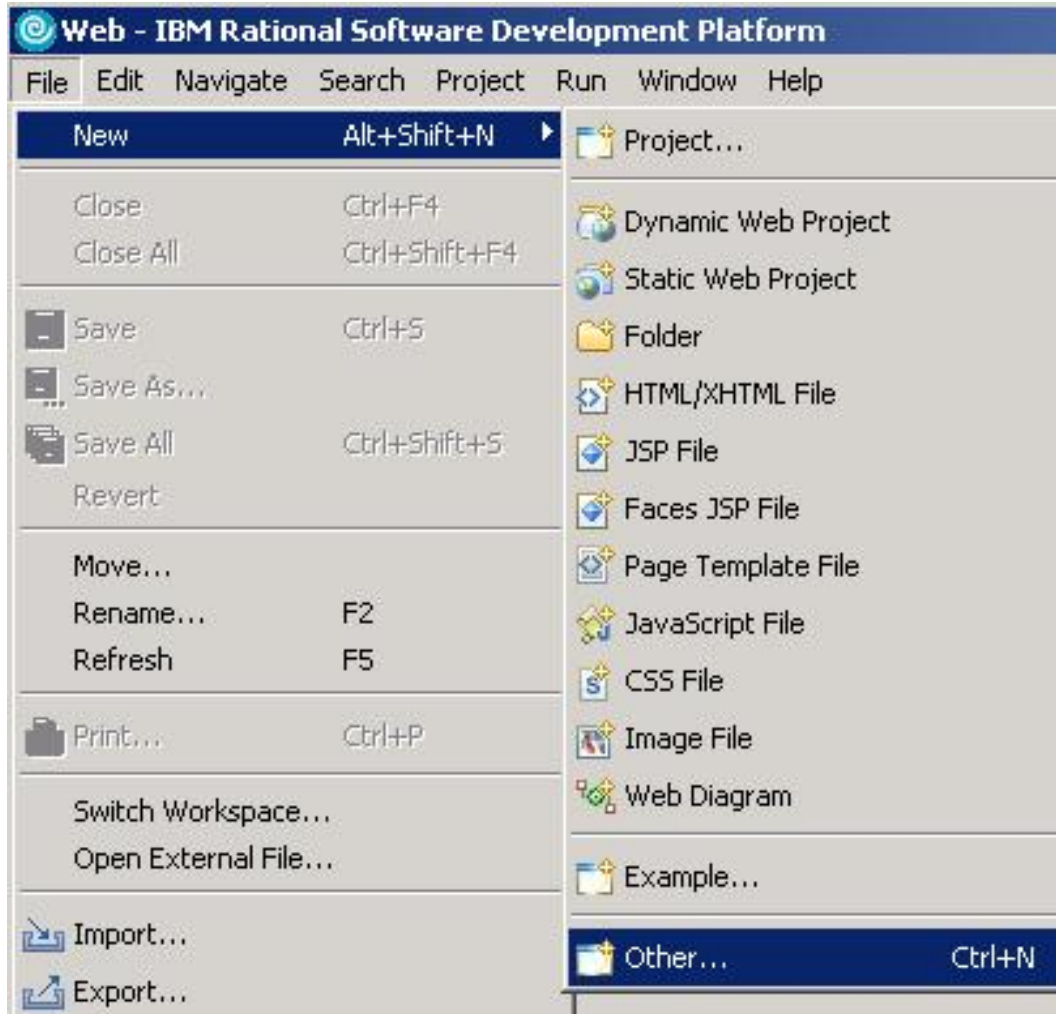


5. Click **Finish**.
6. If prompted to Add the child project "ApplicationWebService" to the EAR project, click **OK**.
7. If prompted to switch to the Web Perspective, click **Yes**.

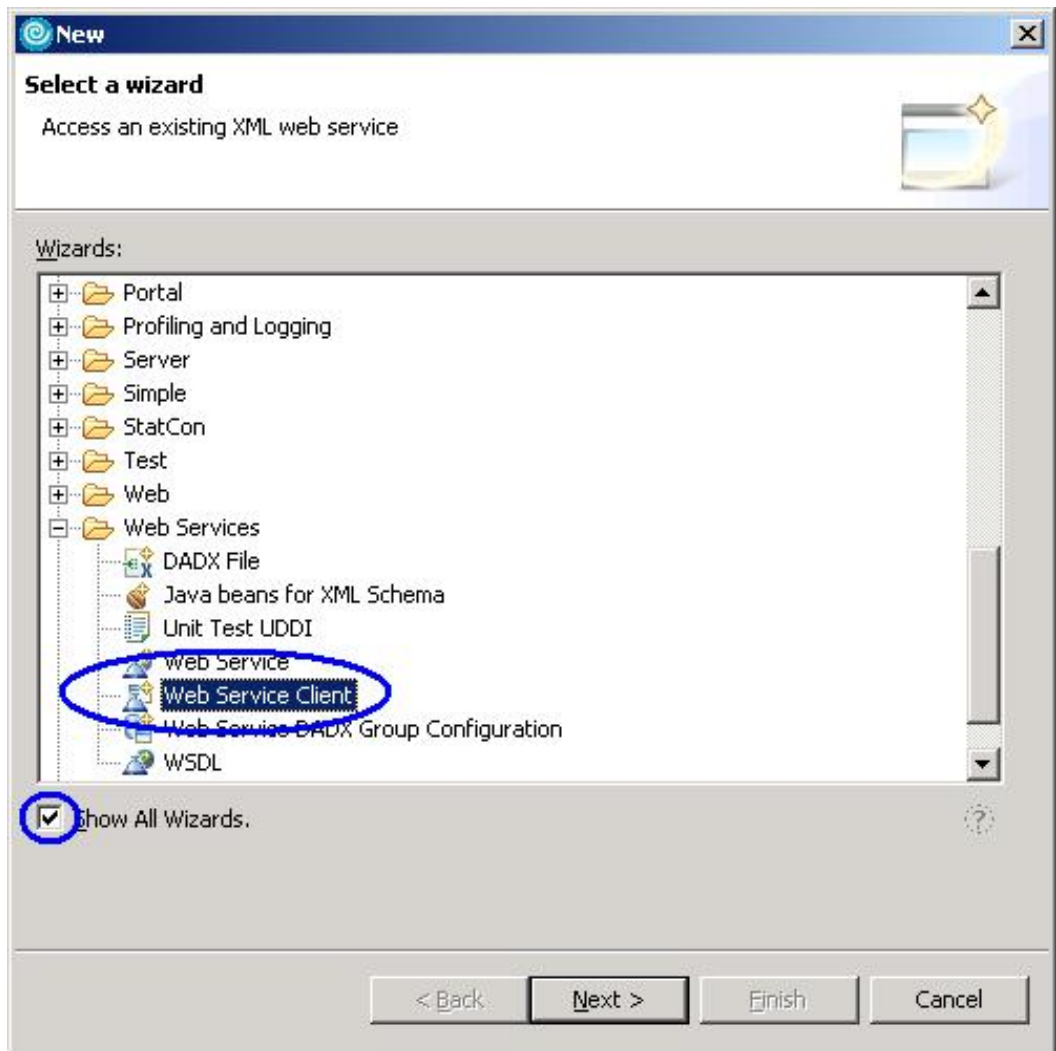
Create Web service client proxy

To create your Web service client proxy:

1. Click **File > New > Other**.
Figure 17. Create a new Web service client proxy



2. Select the Show All Wizards check box.
3. Scroll down and expand **Web Services** and select **Web Services Client**.
Figure 18. Select the Web Service Client option

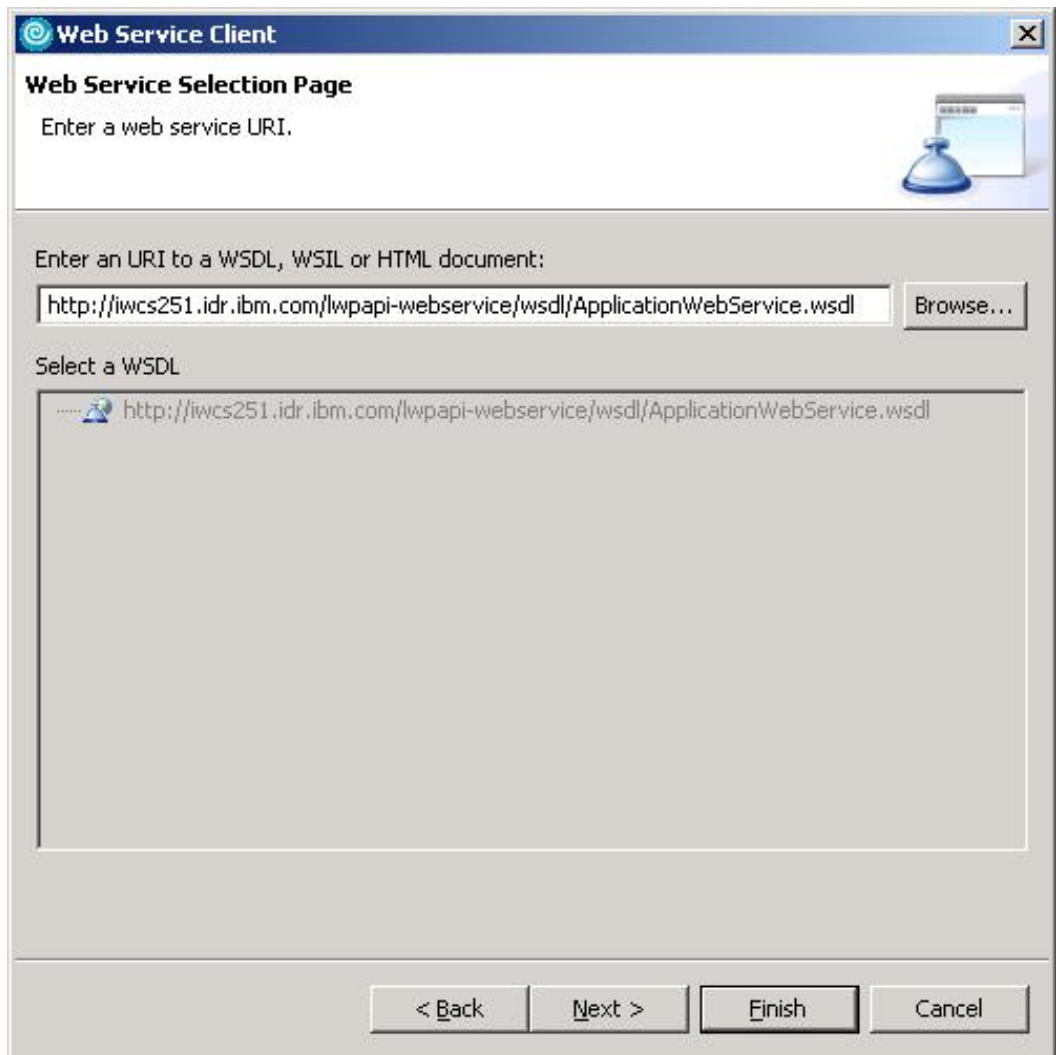


4. If prompted to enable Web Services Development capability, select **OK**.
5. Accept the defaults on Web Services window and click **Next**.
6. Enter the following URL for `..URI` to a WSDL, WSIL or HTML document:

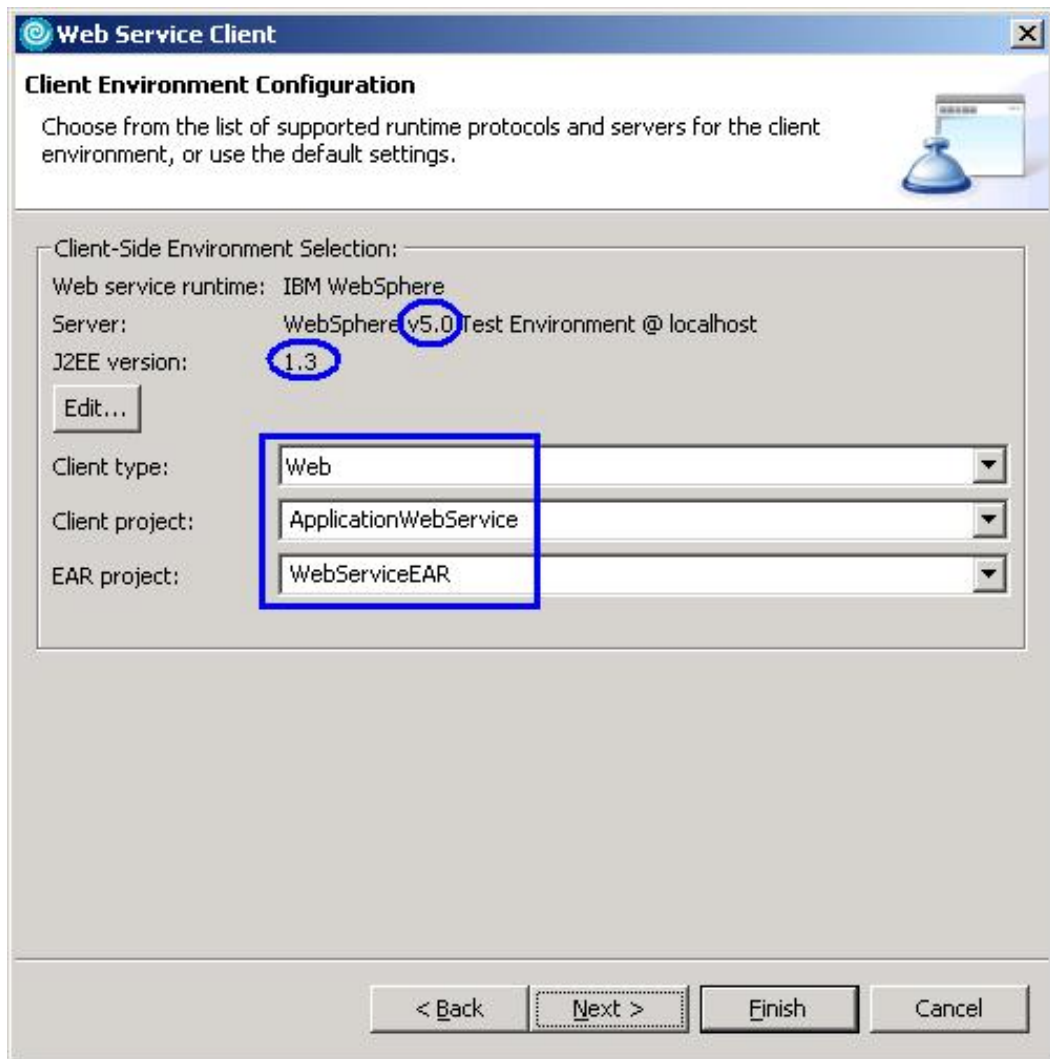
```
http://FullyQualified.Domain.Name/lwpapi-webservice/wsd/ApplicationWebService.wsdl
```

NOTE: `FullyQualified.Domain.Name` needs to be replaced by your environment. For this tutorial, the author used `iwcs251.idr.ibm.com`. The URI is case sensitive, note the `wsd` file name above.

Figure 19. Application Web Service URI

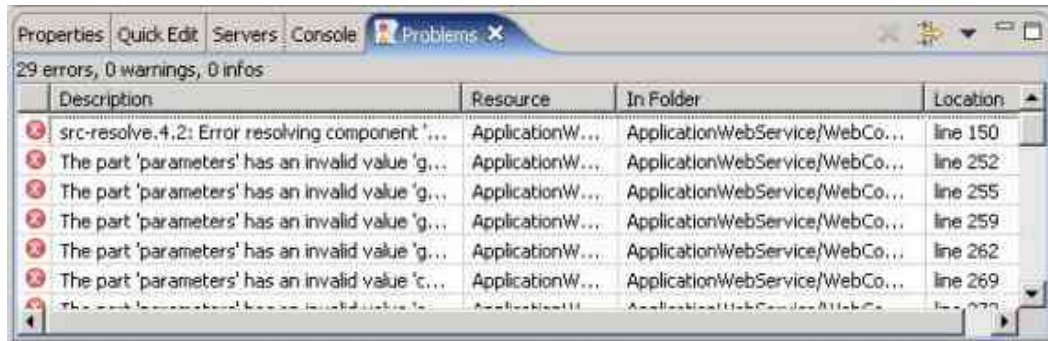


7. Click **Next** .
Note: If you get an error, double check the URI.
 8. On Client Environment Configuration, the following fields should be pre-selected. Verify and make any changes as required.
- Figure 20. Client Environment Configuration**



9. Click **Next**.
10. Click **Finish** on the Web Service Proxy Page.
11. If you check the Problems tab, notice a lot of errors reported for the ApplicationWebService.wsdl file.

Figure 21. Application Web Service Problems

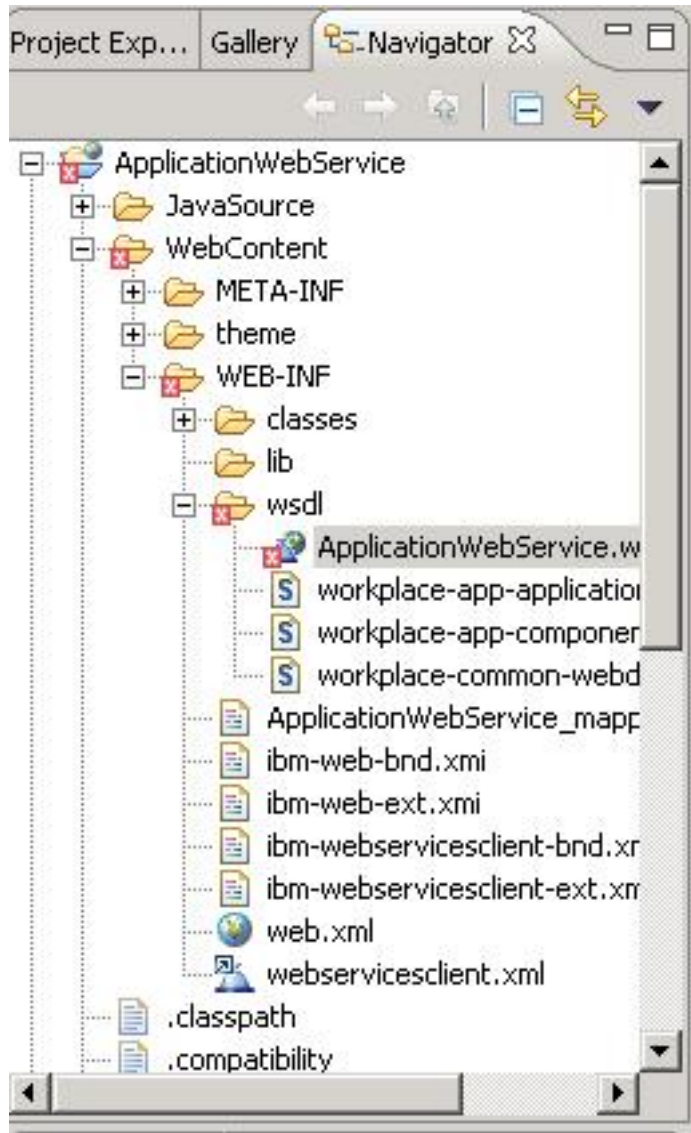


Fix errors

Let's fix the errors.

1. Using the Navigator, locate **ApplicationWebService.wsdl**.
2. Expand to: **ApplicationWebService > WebContent > WEB-INF > wsdl > ApplicationWebService.wsdl**.

Figure 22. Application Web Service WSDL file



3. Double-click **ApplicationWebService.wsdl**.
4. Immediately below the already existing import statements:

```
<xsd:import namespace="http://data.common.api.workplace.ibm.com/2004/12"
  schemaLocation="workplace-common-webdata.xsd"/>
<xsd:import namespace="http://data.application.app.api.workplace.ibm.com/2004/12"
  schemaLocation="workplace-app-application-webdata.xsd"/>
```

add the following code:

```
<xsd:import namespace="http://data.component.app.api.workplace.ibm.com/2004/12"
  schemaLocation="workplace-app-component-webdata.xsd"/>
```

Figure 23. Adding the XSD Import to the Application Web Service

WSDL file

```

ApplicationWebService.wsdl
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:app-application-data="http://data.application.app.api.workplace.ibm.com/2004/12"
  <wsdl:types>
  <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:impl="http://serv:
  <xsd:import namespace="http://data.common.api.workplace.ibm.com/2004/12" schemaLocation="http://data.common.api.workplace.ibm.com/2004/12/common-app-api-workplace-ibm-com-2004-12.xsd" />
  <xsd:import namespace="http://data.application.app.api.workplace.ibm.com/2004/12" schemaLocation="http://data.application.app.api.workplace.ibm.com/2004/12/application-app-api-workplace-ibm-com-2004-12.xsd" />
  <xsd:import namespace="http://data.component.app.api.workplace.ibm.com/2004/12" schemaLocation="http://data.component.app.api.workplace.ibm.com/2004/12/component-app-api-workplace-ibm-com-2004-12.xsd" />
  <!-- *** Common Request and Response Types *** -->
  <!-- WorkplaceWebService.getCurrentUser() -->
  <element name="getCurrentUser">
  <complexType>
  <sequence/>
  </complexType>
  </element>
  <element name="getCurrentUserResponse">
  <complexType>
  <sequence>
  <element name="MemberElement" type="common-data:MemberElement"/>
  </sequence>
  </complexType>
  </element>
  </wsdl:types>
  </wsdl:definitions>

```

5. Save the ApplicationWebService.wsdl file by pressing CTRL + S.

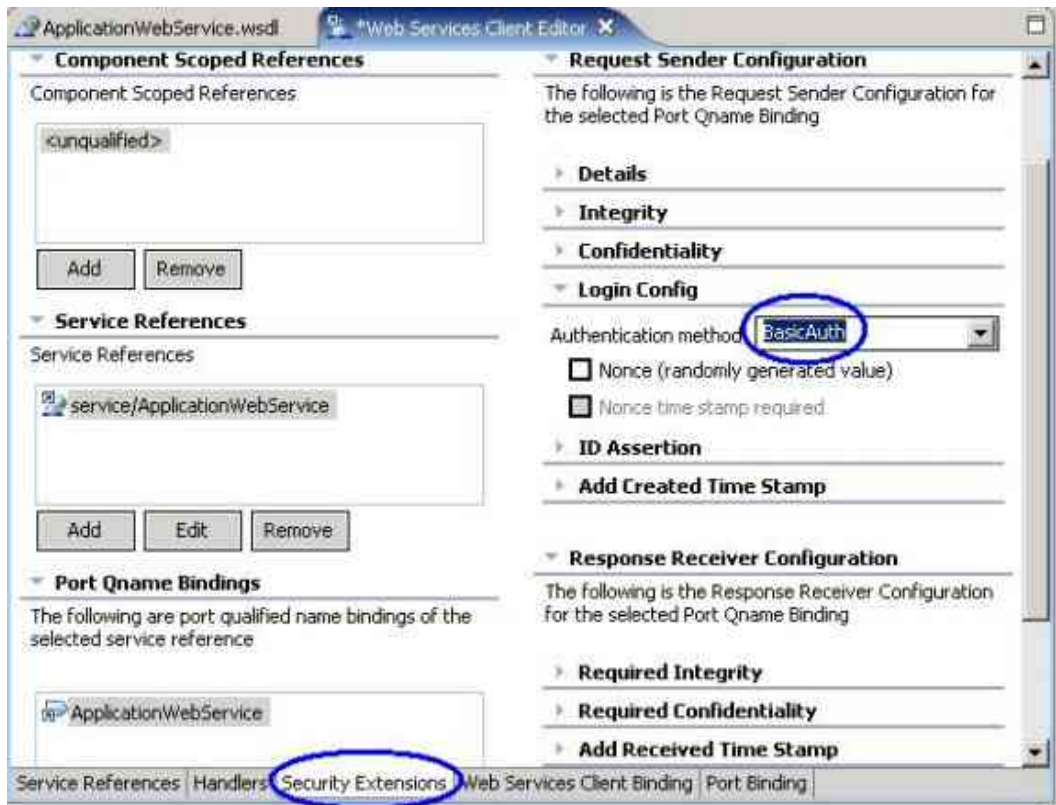
All the errors should be gone.

Set up the user credentials

Now, let's set up the user credential information for your application Web service client.

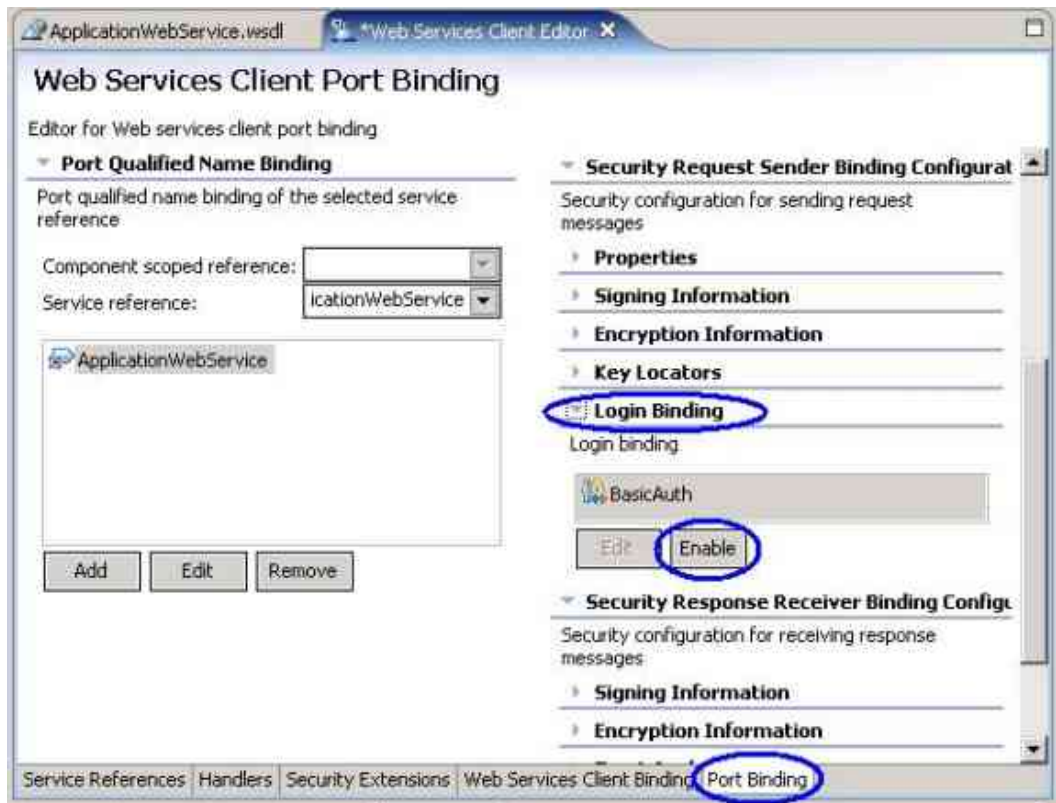
1. Using the Navigator, locate **webserviceclient.xml**.
2. Expand to **ApplicationWebService > WebContent > WEB-INF > webservicessclient.xml**.
3. Double-click on **webservicessclient.xml**.
4. Click the **Security Extensions** bottom tab.
5. Expand the **Login Config**.
6. Select **BasicAuth** for Authentication method.

Figure 24. Select Basic Auth for ApplicationWebService



7. Click the **Port Binding** bottom tab.
8. Expand the **Login Binding**.
9. Click **Enable**.

Figure 25. Enable BasicAuth



10. Select **com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler** for Callback handler.
 11. Enter `wpsadmin` for the User ID and `wpsadmin` for the Password.
 12. Click **OK**.
- Figure 26. Update the Callback handler**

Login binding dialog

Authentication method: BasicAuth

Token value type:

URI:

Local name:

Callback handler: com.ibm.wsspi.wssecurity.auth.callback.NonPromp

Basic authentication:

User ID: wpsadmin

Password: *****

Property:

Name	Value

Add Remove

OK Cancel

13. Save the webservicessclient.xml file by pressing CTRL + S.

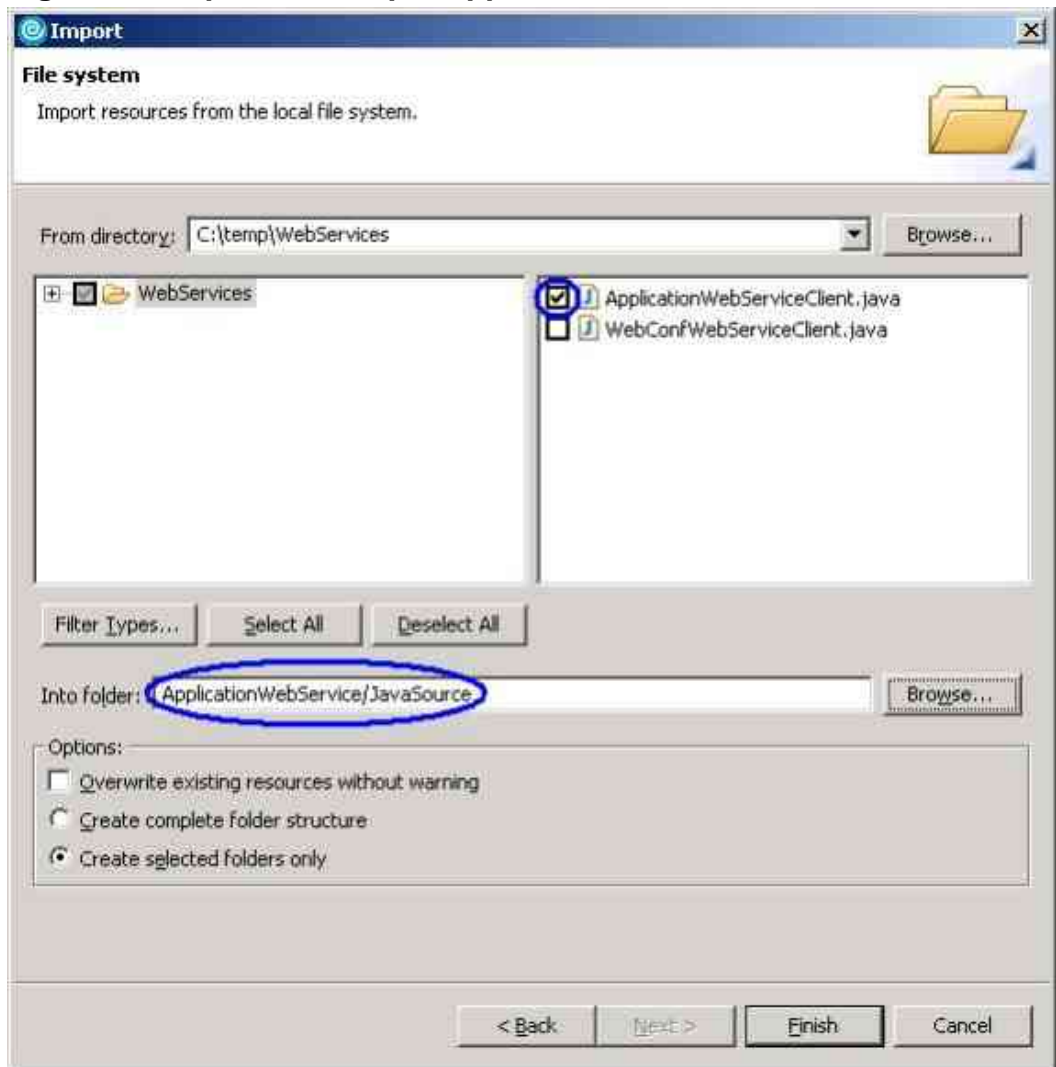
Exploring the application Web service client

Import in a simple Java client for your Web service:

1. Select **File > Import**.
2. Select **File System** as the import source and click **Next**.

3. Click Browse to locate the From directory.
4. Navigate to the **WebServices** folder where you unzipped the **WebServicesClient.zip** and click **OK**.
5. Select **ApplicationWebServiceClient.java**.
6. Click Browse to locate the Into folder.
7. Expand **ApplicationWebService**, select the JavaSource folder and click **OK**.

Figure 27. Import the simple Application Web Service Java client



8. Click **Finish**.

- Using the Navigator, expand to **ApplicationWebService > Java Resources > ApplicationWebServiceClient.java**.
- Double-click **ApplicationWebServiceClient.java**.
- Make sure that the String **endpoint_server** is set to:

```
FullyQualified.Domain.Name
```

Note: For the tutorial, the author used `iwcs251.idr.ibm.com`.

- Following lines, find the ApplicationWebService using the JNDI Lookup. You need a handle to this service in order to invoke methods on it.

```
Context ctx = new InitialContext();
ApplicationWebService appWebService =
(ApplicationWebService) ctx.lookup(appWebServiceName);
```

- Once you have the WebService handle, you can create your proxy object that can be used to invoke methods.

```
ApplicationWebService_SEI appWebServiceProxy =
appWebService.getApplicationWebService(new URL(appWebServEndPoint));
```

- In the first example, you used the Web Conferencing Web service to retrieve the Build and Release information; plus, the boolean - `isAvailable`. In this example, you utilized the Application Infrastructure APIs to retrieve all the Applications and display their members. In get further details on the Web services available via the toolkit and to better understand the different types of APIs, refer to the API documentation.
- Lines 92 - 127 go through all the applications and display their members.

Testing your application Web service client

Now that you have a good understanding of how to create the proxy code and have explored the Java implementation of a Web service client, let's test it.

- Make sure that the IWCS Server you plan to test against is running.
- Make sure that none of the Test Environments within Rational Application

Developer are running. Click the Servers tab.

3. The **WebSphere v5.0 Test Environment @ localhost** should be running. If it is, stop it by right-clicking the Started status and select **Stop**.
4. All the Test Environments should be in the Stopped state.

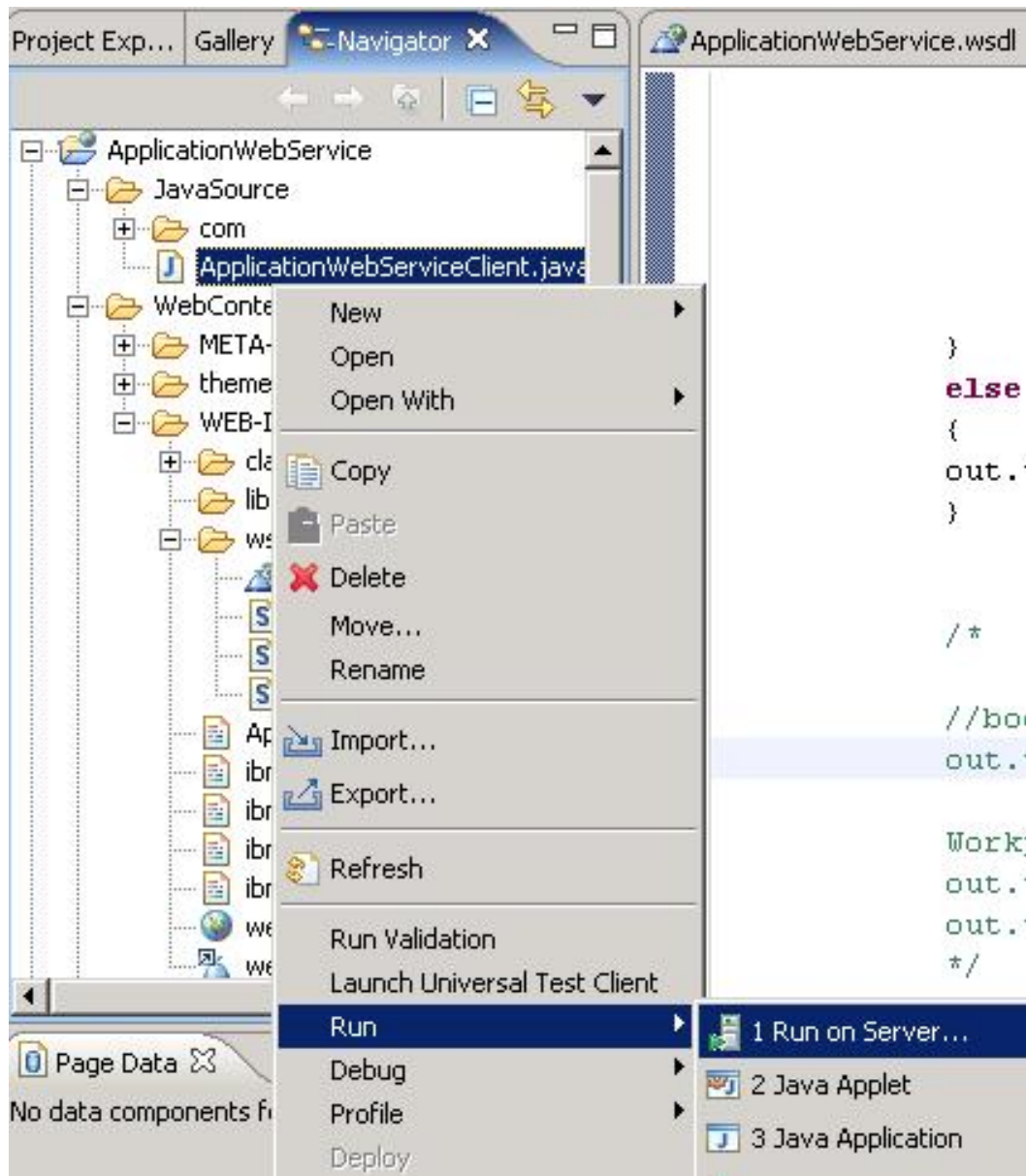
Figure 28. Test Environment status



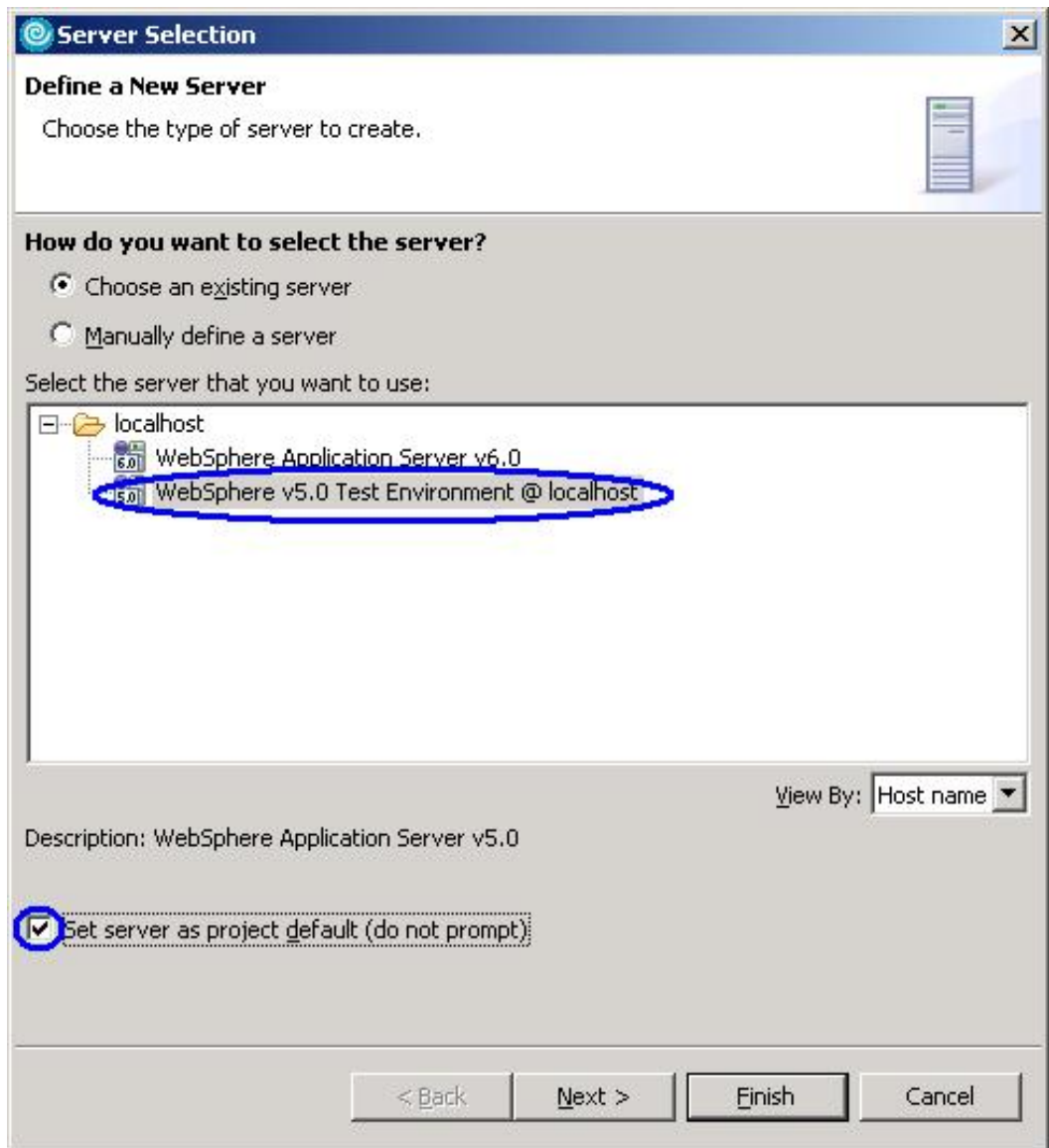
Server	Host name	Status
WebSphere Application Server v6.0	localhost	Stopped
WebSphere v5.0 Test Environment...	localhost	Stopped

5. Using the Navigator, Expand to **ApplicationWebService > Java Resources > ApplicationWebServiceClient.java**.
6. Right-click **ApplicationWebServiceClient.java** and select **Run > Run on Server**.

Figure 29. Run the Web service client

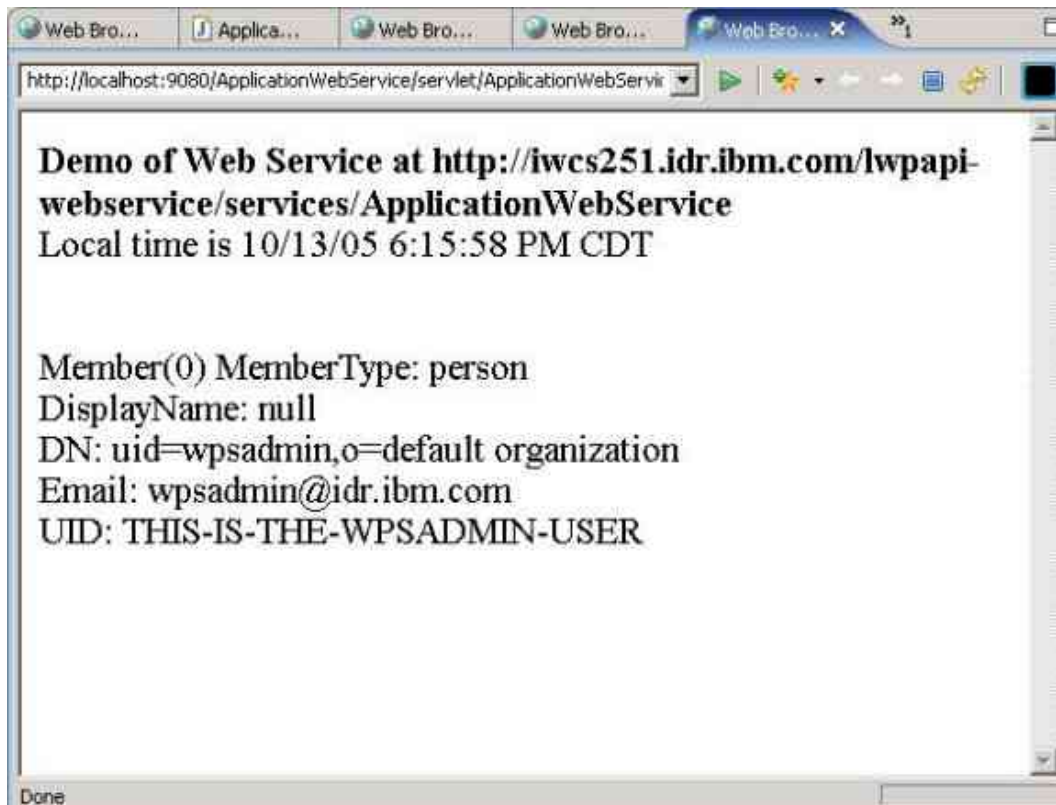


7. On the Server selections window, select **WebSphere v5.0 Test Environment @ localhost**.
8. Check **Set server as project default (do not prompt)**. **Figure 30. Select the Test Environment**



9. Click **Finish**.
The Test Environment starts and displays the output in a Web Browser launched within Rational Application Developer.

Figure 31. Application Web service result



Similar to your results from the first example, here you see the output of the methods you executed on the server. Depending on how many applications you have on your system, you see the list of users from those applications. This is a screen shot of a test server, with no applications. It appears this method always returns details about this one user as shown above. This must be from some internal application. You can test it on your own. It is also recommended to test these results after creating one or two applications on your own server.

Section 4. Summary

In this tutorial, you explored some of the Web services available through IWCS Server. The first example was to introduce you to Web services by testing a simple Web Service Client code. The second example, expanded on the first sample by using executing Application Infrastructure based APIs through Web services. At this time, you should have a good understanding of how to use Rational Application Developer to connect to the WSDL files provided by the IBM Workplace Collaboration Services Server. To continue exploring the Web services and other extension points available within the IBM Workplace Collaboration Services, it is

recommended to read the documentation provided with the toolkit. The documentation covers all the Web services and other extension points. Using the documentation, you can see and learn about the specific methods exposed by the individual Web services (via the WSDL files).

Downloads

Description	Name	Size	Download method
Web Services Client Sample Code.	WebServicesClient.zip	3KB	HTTP

[Information about download methods](#)

Resources

Learn

- Learn more about IBM Workplace Collaboration Services [API Toolkit](#).
- [IBM Workplace](#) on developerWorks contains many technical resources.
- The [technical library view](#) for IBM Workplace has related topics you might find helpful.
- [Exploring the application development options for IBM Lotus Workplace 2.0](#) is an article that explains the APIs and SPIs provided by the IWCS V2.0 Server.

Get products and technologies

- Build your next development project on Linux™ with [IBM trial software](#) available for download from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- See a list of all the [IBM Workplace](#) discussion forums.
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).
- Read or participate in a discussion forum for [Workplace Application Development tools](#), including Workplace Designer, Workplace Builder, toolkits, and more.
- Read or participate in a discussion forum for [Workplace Collaboration Services](#), including Messaging, Team Collaboration, Collaborative Learning, and Documents.

About the author

Rohit Sahasrabudhe



Rohit is an enablement architect for ISV and Developer Relations Technical Consulting in Austin, Texas. Rohit is an IBM Certified for e-business Solution Designer, a Solution Technologist, and a Red Hat Certified Engineer. He joined IBM as a Software Engineer, working on Lotus Domino Solutions. Now you will find him helping IBM Business Partners and Independent Software vendors gain knowledge in IBM products such as WebSphere Application Server and Portal Server including others. For the past two years, he has been leading the IBM Workplace Collaboration Services efforts for Business Partners. He

travels around the world educating, enabling and evangelizing. You can contact Rohit at sahas@us.ibm.com.