

Enterprise modernization for IBM System z: **Creating Web services with EGL Wizards**

Skill Level: Intermediate

[Reginaldo Barosa](#) Executive IT Specialist – IBM Boston

October, 07 2009

This tutorial shows you how to build and test a simple Web service using IBM® Rational® Business Developer, which supports service-oriented architecture (SOA). SOA is a method of organizing applications in modular pieces (called services and clients). The services provide logic to the clients in the form of functions, similar to the way that Enterprise Generation Language (EGL) libraries make functions available to programs.

Before you start

Learn what to expect from this tutorial, and how to get the most out of it.

Be sure that you have started the [System z Sandbox](#) and that [IBM® Rational® Developer for System z](#) is running under the Windows in the system z Sandbox.

About this series

Walk through this scenario and others online as part of the Assets entry point and the Skills entry point of the [Enterprise Modernization Sandbox for IBM® System z®](#).

About this tutorial

This tutorial shows you how to build and test a simple Java<trade/> Platform, Enterprise Edition (JEE) Web service using IBM® Rational® Business Developer to be deployed into an IBM® WebSphere® Application Server.

Objectives

These are the steps that you will perform in this tutorial:

1. Create an EGL Web project that will hold the Web services.
2. Create the service: Here, you will create a new EGL Service file.
3. Set up the service as a Web service, and generate the Java code.
4. Test the Web Service using the generated WSDL.

Prerequisites

You need to be familiar with basic Web development concepts.

System requirements

The System z Sandbox

Getting started

This tutorial shows you how to build and test a simple Web service using IBM® Rational® Business Developer, which supports service-oriented architecture (SOA).

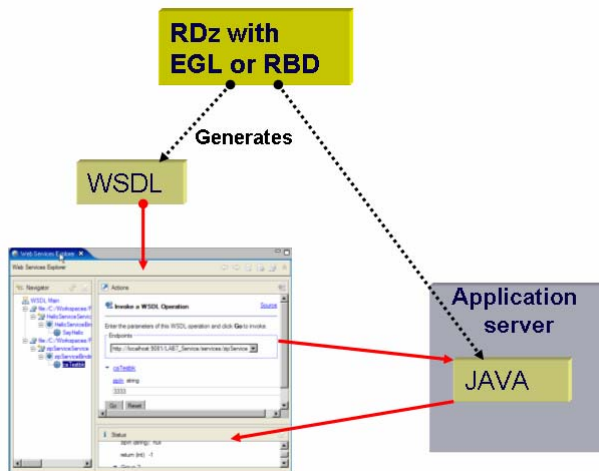
Create Web services with EGL

SOA is a method of organizing applications in modular pieces (called services and clients). The services provide logic to the clients in the form of functions, similar to the way that Enterprise Generation Language (EGL) libraries make functions available to programs.

Tools and architecture in these examples

This tutorial uses Rational Developer for System z with EGL, but it could be done using Rational Business Developer alone. Bear in mind that, in SOA, the services are stateless; they do not "remember" interactions with a particular client. In this way, each time the service is called, it is as though that service is being used for the first time. Services are also able to provide their functions to a wide variety of applications through the WSDL standard, promoting flexibility and code reuse, as shown in Figure 1.

Figure 1. Rational Business Developer services in SOA



What you will learn from this tutorial

By completing the exercises in this tutorial, you will build an EGL Web service. At the end of this section, you should be able to test the Web services that you have created and deployed to IBM® WebSphere® Application Server.

Build the Web service

You will create a simple Web service that uses EGL. The service accepts two fields that perform an add operation and return the results. In other words, you will do a simple calculation that will demonstrate how easy it is to create and deploy Web services using EGL. In your company, the business logic will be more complex than this one.

These are the steps that you will perform in this section:

1. Create an EGL project that will hold the Web services.
2. Create the service: Here, you will create a new EGL Service file.
3. Set up the service as a Web service, and generate the Java code.
4. Deploy the Java™ code and test it using the WSDL-generated code.

Section 1 - Create an EGL project for Web services

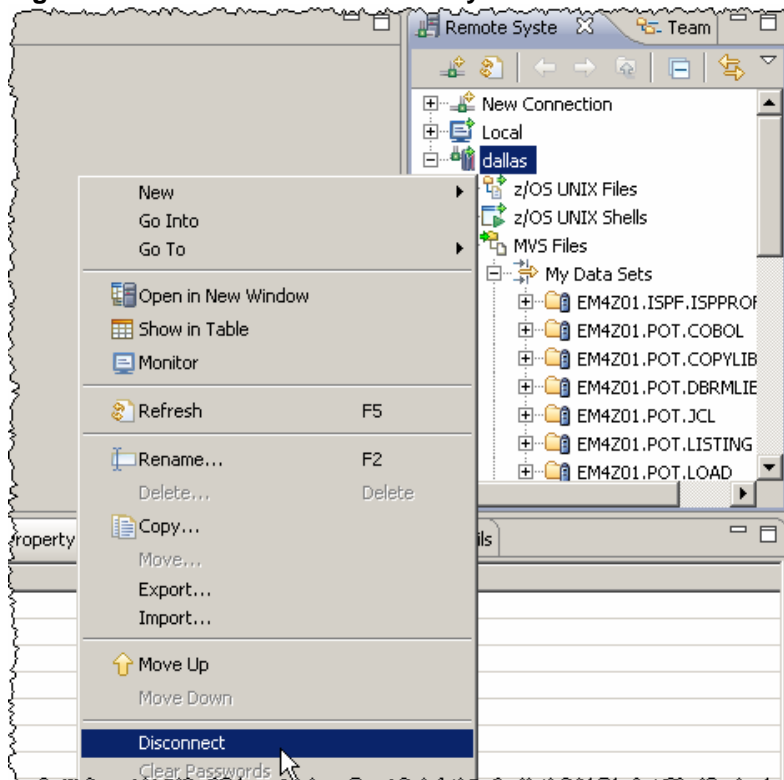
This example assumes that Rational Developer for System z with EGL is already started. Note that you are operating under VMWARE, which may slow response time.

Disconnect from z/OS System

When The VMWARE session is started you will be connected to z/OS system and you will see the z/OS Projects perspective..

1. Because in this tutorial you will not use z/OS, you can disconnect to the z/OS system. Right click on dallas and select disconnect as shown in Figure 2.

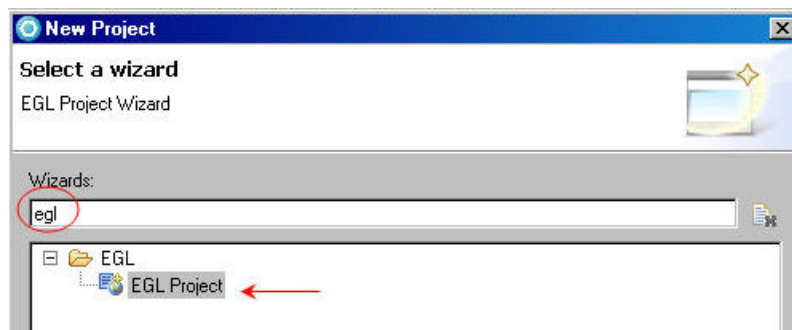
Figure 2. Disconnection from z/OS system



Creating the new project

1. To create a new EGL Web project in the Workbench, click **File > New > Other** (or use Ctrl + N).
2. Type `egl` in the Wizards field, select **EGL Project**, and click **Next**, as shown in Figure 3.

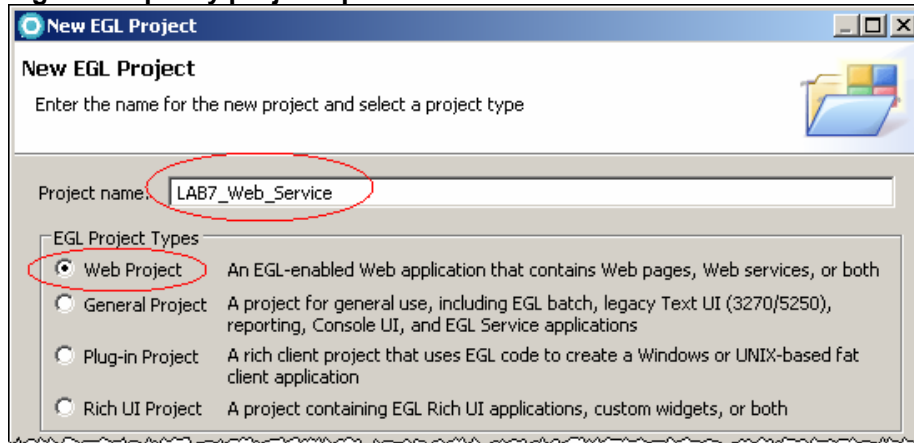
Figure 3. Select a wizard



Creating Web services with EGL Wizards

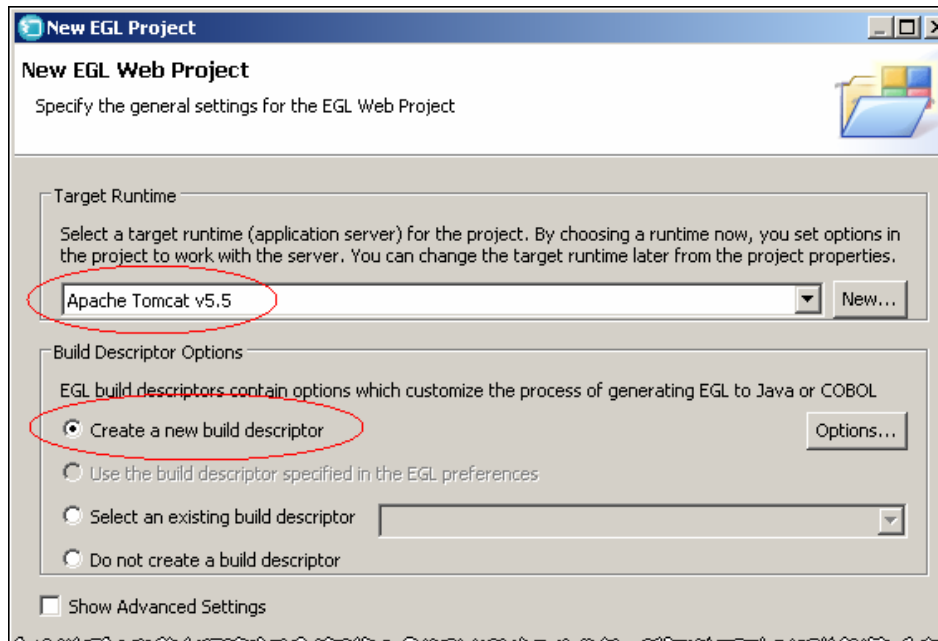
3. Name the project **LAB7_Web_Service**, select **Web project**, and click **Next**, as shown in Figure 4. Remember that VMWARE will slow you down, be patient until this operation give you another dialog.

Figure 4. Specify project options



4. Use the drop down and select **Apache Tomcat v5.5** as *Target Runtime*. Be sure that **Create a new build descriptor** is selected and click **Finish** as shown in Figure 5.

Figure 5. Specify more project options



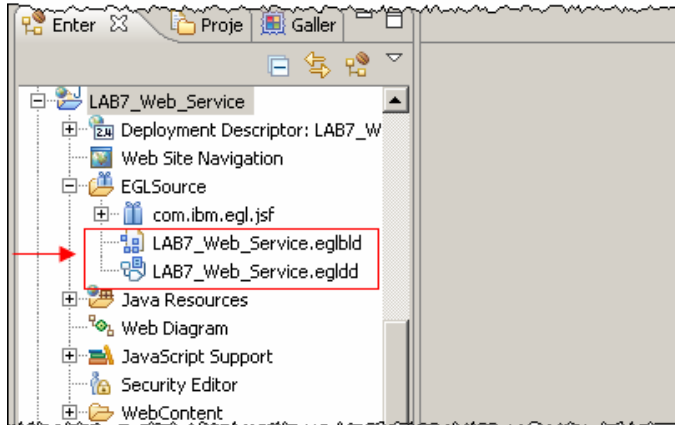
5 If the “*Confirm Perspective Switch*” dialog may appear, and asks you to switch to the Java EE perspective click **Yes**. We could use Java EE perspective to play with EGL Web services.

6 Close any opened window using **Ctrl + Shift + F4**.

Creating Web services with EGL Wizards

7 Expand the folder in the *Project Explorer* called **LAB7_Web_Service** and **EGLSource**. Note that an EGL build descriptor (*.eglbd) as well the EGL deployment descriptor (*.egldd) has been created with the names *LAB7_Web_Service.eglbd* and *LAB7_Web_Service.egldd*. Later you will modify the defaults created. (Figure 6)

Figure 6. Project created

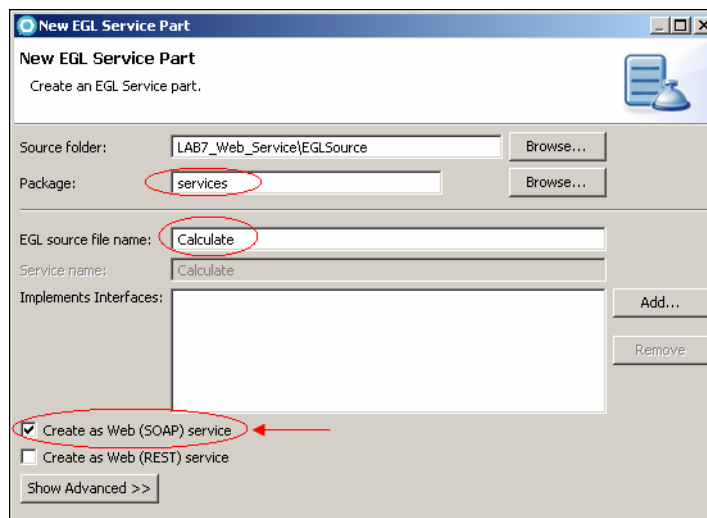


Section 2 - Create the service business logic


Perform these steps to create the service business logic:

1. Right-click the **EGLSource** folder and select **New > Service**.
2. In the Package field, type **services**.
3. In the EGL source file name field, type **Calculate**.
4. Select **Create as web (SOAP) service** and then click **Finish**. Be patient may take a while here.

Figure 7. Creating a new SOAP Web service

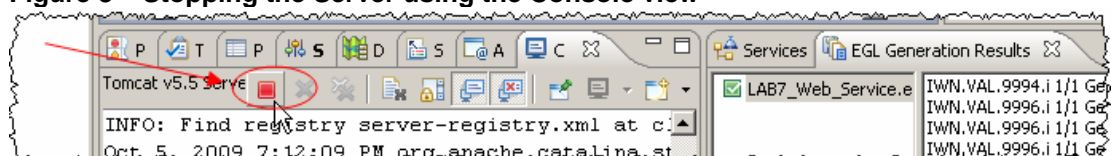


5. The code will be deployed to Tomcat, just be patient and wait until Tomcat is started.

When is started you now need to **stop it** using the **Console** view and clicking the red icon.  Using the *Console* view just click in the Red button to stop this server.

You still need to add the code before deploying it.. ☺ (Figure 8)

Figure 8 – Stopping the Server using the Console view



Creating Web services with EGL Wizards

The EGL code will be created and the editor will open.

6. Using content assist (Ctrl + Space) add the code in Listing 1 to the editor. (If you prefer, you can import or copy the code from **C:\EGL_POTLAB7\Calculate.egl**).

Listing 1. Code for creating EGL Calculate service

```
package services;

// service
service Calculate

function addFields(field1 int in, field2 int in)
    returns (int) result int;
    result = field1 + field2;
    return (result);
end

end
```

7 Save and close the file (**Ctrl + Shift +F4**)

8 Click **Yes** to save the changes

What you have done so far

You have created an EGL Web Project and defined an EGL Web Service with only one operation (add two fields). Now that you have written the code for this service, you can make it available to other applications as a Web service or as EGL services.

Note: In this tutorial, you will create only a SOAP Web service. Making the service available in this way involves creating service binding information, which tells other applications where to find the service and what functions are available. The service publishes this information as a WSDL file. As an alternative to SOAP Web services, EGL client applications can access EGL service applications as EGL services. This method offers better performance than Web services, but it can be used only between two EGL applications.

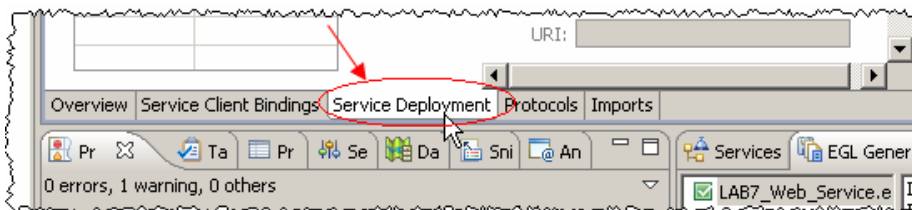
Notice that the Java generation has taken place by default. You should not have generation errors.

Section 3 - Set up the service as a SOAP Web service

In this step, you create an EGL deployment descriptor and add binding information to it. You will also configure the project's build descriptor to use the deployment descriptor.

1. Double-click **LAB7_Web_Service.eglidd** to open the EGL service deployment descriptor and then click the **Service Deployment** tab, which is located at the base of the editor (see Figure 9).

Figure 9. Service Deployment tab



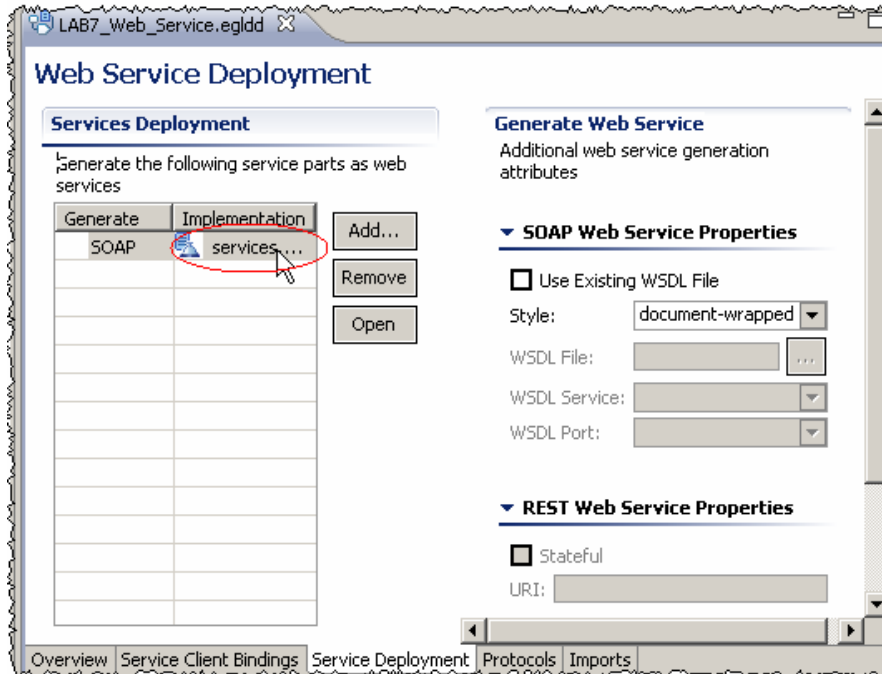
Creating Web services with EGL Wizards

2. Move the mouse to **services...** under **Implementation**.

Notice that this service is shown in the list on the Service Deployment list. You do not need to add any other information on this tab for now. The deployment descriptor is showing the Web service binding information. No changes are required here.

(Figure 10)

Figure 10. Web Service Deployment definitions



Using the deployment descriptor with services

When you write a service in EGL, you must also create a deployment descriptor to accompany it. One default EGL service deployment descriptor was created for you.

The deployment descriptor file contains both the information that describe how your service part calls external applications and the information that describes how your EGL service is wrapped with a Web service and exposed to external applications. When used with Java generation, the generating EGL deployment descriptor files result in a runtime binding file (which uses the naming convention of ddname-bind.xml). When used with COBOL generation, a binding program is generated, instead.

3. Double-click **LAB7_Web_Service.egl**, which is located in the project's EGLSource folder. The build descriptor file contains build descriptor options that describe how EGL will generate your project into the output language.

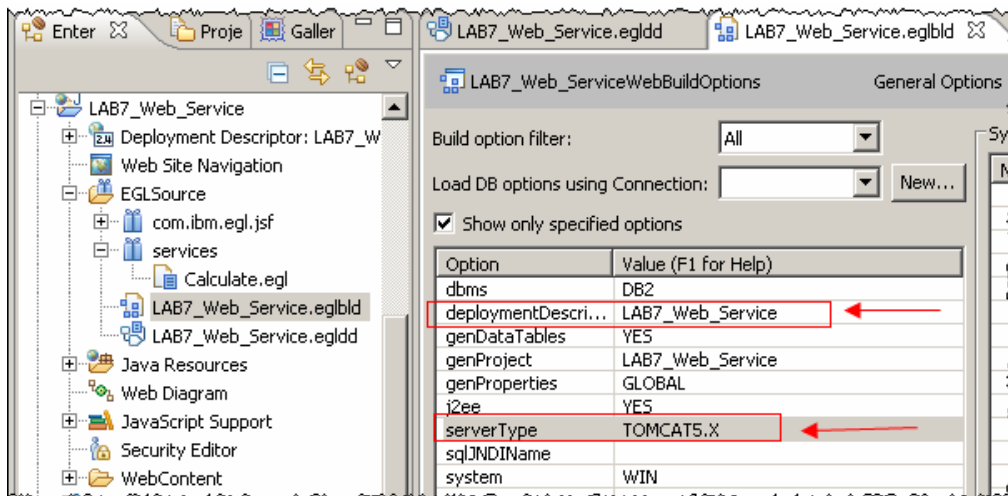
Notice that the deploymentDescriptor is pointing to **LAB7_Web_Service.egldd**, and that **serverType** is TOMCAT5.X (

Creating Web services with EGL Wizards

Figure 11). You can implement this service in other supported servers, such as WebSphere Version 6.X, IBM® CICS® Version 3.X, , Apache Tomcat Version 5.X and Apache Tomcat Version 6.X

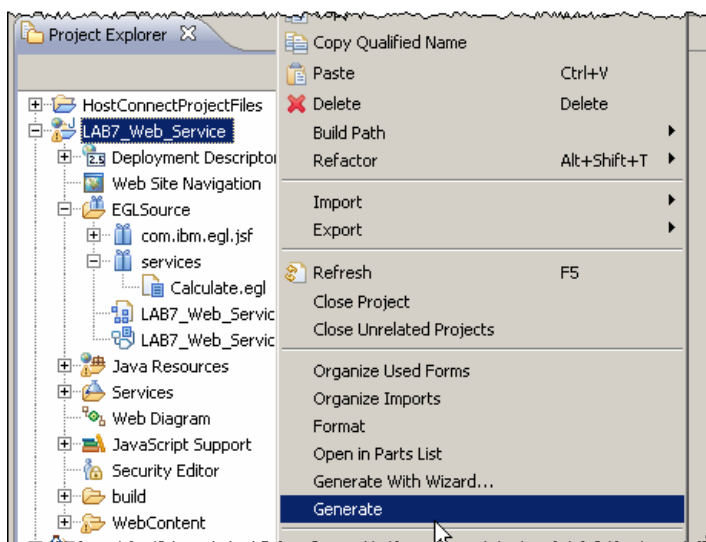
Creating Web services with EGL Wizards

Figure 11. EGL Deployment Descriptor



4. Close all opened editors by using **Ctrl + Shift + F4**. No changes are required.
5. Again, no changes are required here, so click **No** if you are asked if you want to save any change.
6. Right-click on the project, **Lab7_Web_Service**, and select **Generate** (Figure 12). This will generate Java code.

Figure 12. Generate Java code

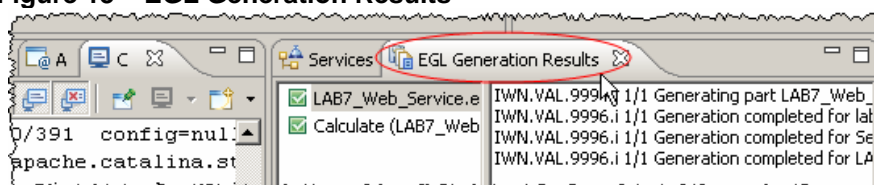


The code is automatically deployed to Tomcat and the server is started .
See the message similar to:
"INFO: Server startup in 6178 ms"

You are now ready to test the code.

7. If you click on tab **EGL Generation Results** you should have no errors as shown in Figure 13

Figure 13 – EGL Generation Results



What you have done so far

You have created an EGL Web project and defined an EGL Web service.

You also verified the EGL deployment descriptor to create Web services and the EGL build descriptor, and generated the EGL, thus creating Java code. You might now test the Web services that was already deployed.

Section 4 - Test the Web Service using the generated WSDL

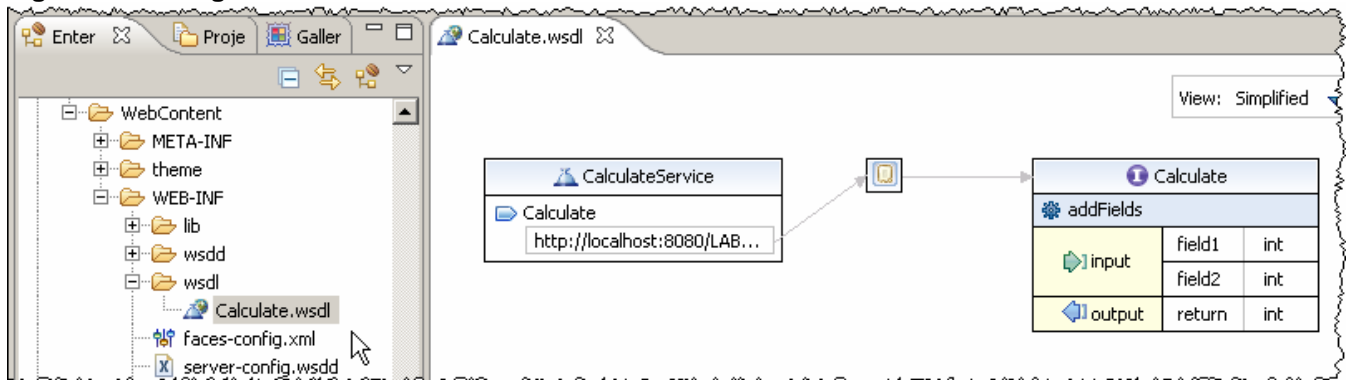
The Generated WSDL file

WSDL files communicate information about services to clients, describing the functions provided in the service and specifying the location of the service. In this section, you generate a WSDL file from the service.

To see the WSDL:

1. Expand **WebContent**, **WEB-INF wsd** and double click on **Calculate.wSDL** to edit it (Figure 14). Notice the input message (field1 and field2) and the output message (return).

Figure 14 – The generated WSDL



2. Close all opened files (**Ctrl + Shift + F4**).

The WSDL file describes the service so that clients can connect to it at run time, and the deployment descriptor file allows EGL to make the service available at run time.

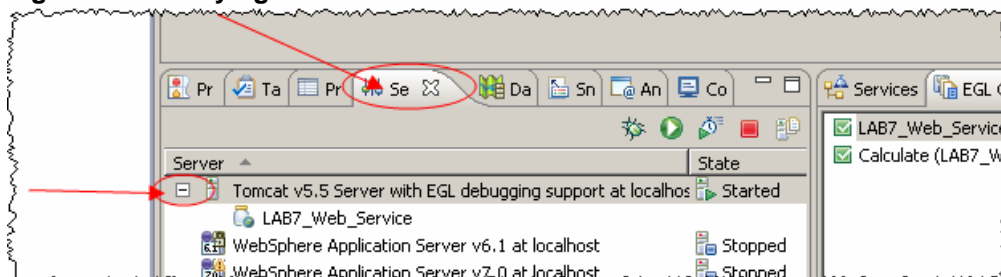
Test the Web Service using the generated WSDL

3. First let's make sure that the Tomcat Application Server is running and the Java code is already deployed.

Using the **Servers** view (click on **Servers** tab), expand the plus sign (+) under Tomcat and verify that the project **LAB7_Web_Service** is already there and that the server state is **Started**.

If not already started, just select it and click on the green arrow to start it. (Figure 15).

Figure 15 – Verifying that Tomcat Server is started

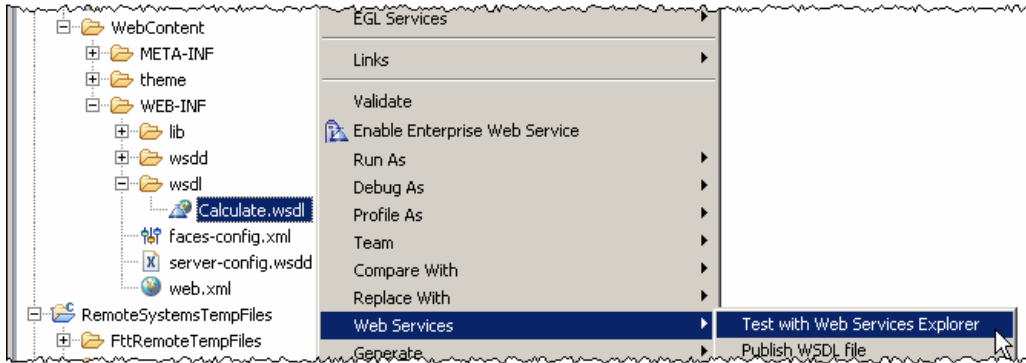


Creating Web services with EGL Wizards

Rational Business Developer offers a nice interactive Web Services test facility called the Web Services Explorer! You can use this tool to test your Web Service functionality effectively, before embedding calls to it from your service client!

4. Right click on **Calculate.wsdl**, and select **Web Services → Test with Web Services Explorer**. (Figure 16)

Figure 16 – Using the Web Services Explorer

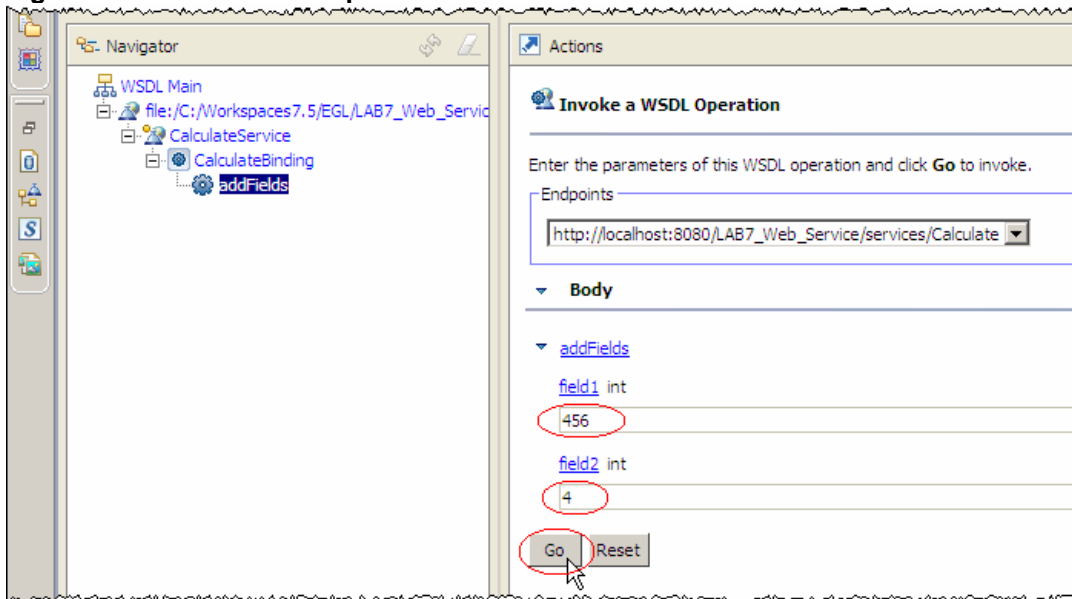


5. Double click the blue title (Web Services Explorer). to have this view maximized.

Expand **CalculateBinding** and click **addFields**, which is the service that you are going to test. Enter two fields to be added, (field 1 integer value of **456**, field 2 integer value of **4**, and then click the **Go** button.

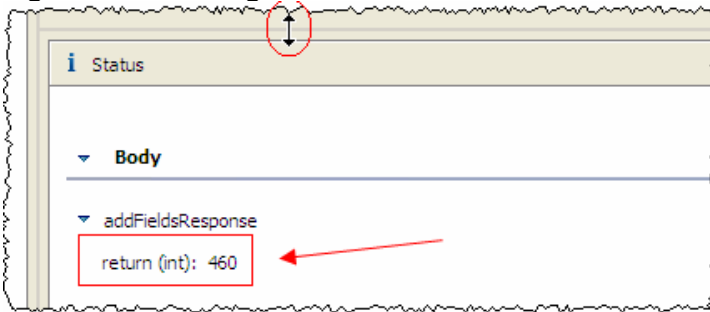
You will need to scroll down the Web Services Explorer view to see the GO button.

Figure 17 Web Services Explorer results.



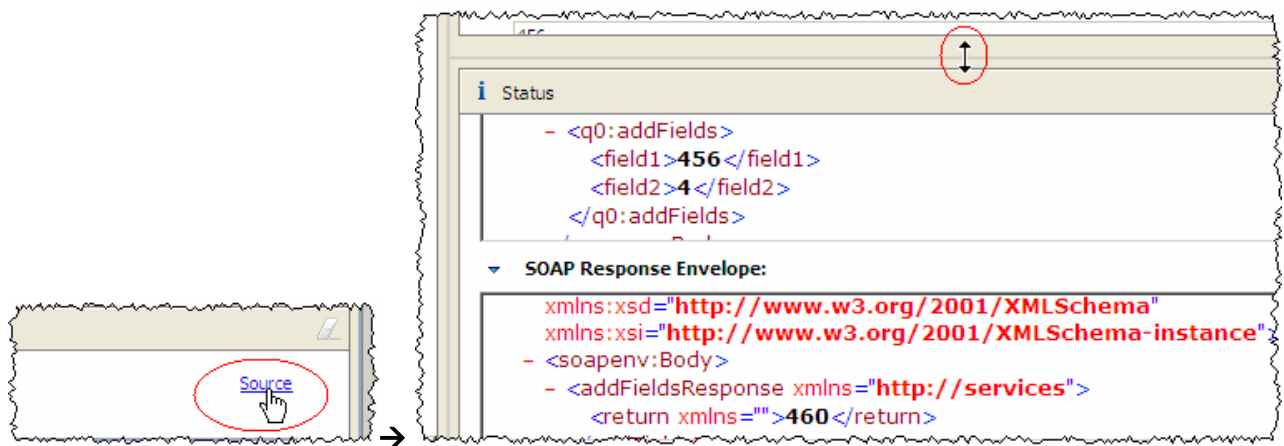
- Creating Web services with EGL Wizards
6. Resize the Status area to see the results.

Figure 18 – Resizing the view



7. Click on **Source** link and resize the window to see the SOAP envelope (Figure 19)

Figure 19 – SOAP Envelope



8. Close all opened files (**Ctrl + Shift+ F4**).

What have you done so far?

You have defined an EGL Web Service, verified the EGL Deployment descriptor to create Web Services, verified the EGL build descriptor and generated the Java Code

You generated the WSDL of the Services created and deployed the Java code created that implements the web service.

Using the WSDL generated you tested the service, running the Java deployed code inside of the Tomcat Application Server.

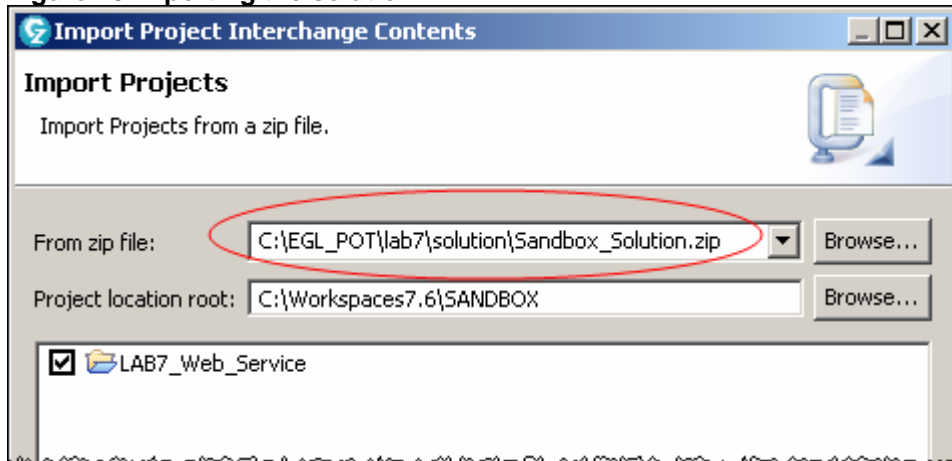
You also verified the input and output SOAP messages..

Solution

If you could not complete the tutorial, do not get frustrated. If you missed one step or selected a bad choice in any of the wizards, you would have problems.

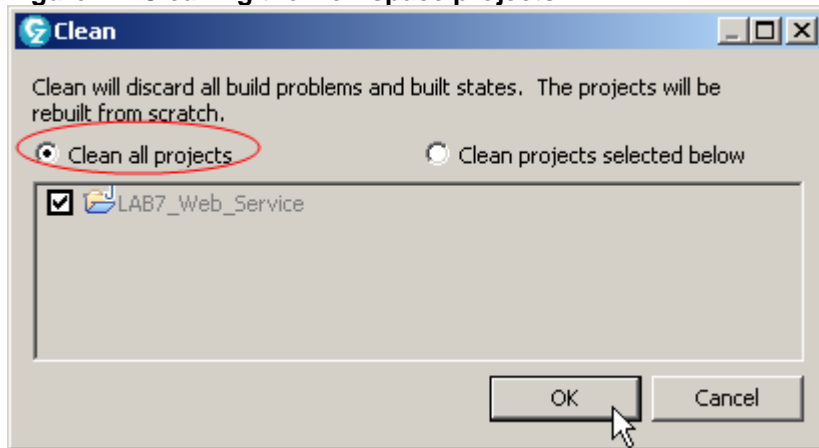
1. In that case, you can load the solution to your workspace by selecting **File > Import > Other > Project Interchange** and using the solution located at **C:\EGL_POT\lab7\solution\Sandbox_Solution.zip**.
2. Select **LAB7_Web_service** project as show in Figure 20 and click **Finish**.

Figure 20 Importing the solution.



2. Open the **Web perspective** and select **Project > Clean.. > Clean all projects** as seen in Figure 21)

Figure 21. Cleaning the workspace projects



4. Perform the tests as seen in Section 4.

Resources

Learn

Get more information about EGL visiting [EGL Café](#)

Find out more about IBM [Enterprise Modernization Solutions](#).

Learn more about [IBM Rational Developer for System z](#).

Watch a demo of [Rational Developer for System z](#).

Read [Unleash the power of mainframe assets into SOA](#).

Visit the [Rational page on developerWorks](#) to find technical resources and learn about best practices for the Rational Software Delivery Platform.

Subscribe to [The Rational Edge weekly newsletter](#).

Subscribe to the [IBM® developerWorks® newsletter](#), a weekly update on the best of developerWorks tutorials, articles, downloads, community activities, Web casts and events.

Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

Get an evaluation version of [Rational Developer for System z](#).

Download [trial versions of other IBM Rational software](#).

Download these [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Tivoli®, and WebSphere®.

Discuss

Join the [Architecture forum on developerWorks](#) to get connected with others and take advantage of their expertise and experience to get you tips that can help you as a developer or architect to use the principles of service-oriented architecture (SOA).

Check out [developerWorks blogs](#). and get involved in [the developerWorks community](#).

About the author



Reginaldo W. Barosa is an IBM Certified Application Development Specialist. He provides sales support, helping customers with enterprise transformation solutions and development tools, such as IBM WebSphere Developer for System z. Before joining IBM US, Reginaldo worked for 27 years in IBM Brazil. He has co-authored IBM Redbooks and has written books, as well as other articles for IBM developerWorks. He holds a degree in electrical engineering from Instituto Mauá de Tecnologia, São Paulo, Brazil.