HATS Toolkit 7.1 with the IBM WebFacing Tool

Create an Application Bridge between WebFacing and an EGL application.

Hands-On Tutorial

Table of contents

Introduction:..3
System requirements:...4
Estimated time required:...4
Prerequisites:...4
Hands-on:..5
Hands-on 1: Setting up the WebFacing and EGL projects................6
Step 1: Importing the WebFacing and EGL projects. 6
Hands-on 2: Creating a connection to your i5/OS, restoring the save file and setting up the WebFacing connection. 7
Step 1: Restoring the save file. 7
Step 2: Specify the i5/OS host name for the WebFacing project. 9
Hands-on 3: Setting up the database connection for the EGL project. ...10
Step 1: Creating the database connection 10
Hands-on 3 checkpoint: 12
Hands-on 4: Application Bridge Web Setting. 13
Step 1: Adding Application Bridge Web Setting........................13
Hands-on 4 checkpoint: 15
Hands-on 5: Adding display elements to the EGL Web page. 16
Step 1: Adding the display elements. 16
Step 2: Displaying the data on the Web page..........................17
Hands-on 6: Setting up the EGL project for the Application Bridge Scenario 1.....19
Step 1: Adding the EGL code..19
Step 2: Binding the returnToWebFacing function to the Web page.......23
Hands-on 7: Testing the Application Bridge Scenario 1. 23
Step 1: Running the Application. 23
Hands-on 7 checkpoint: 25
Hands-on 8: Setting up the Application Bridge Scenario 2. 26
Step 1: Adding the EGL code..26
Hands-on 9: Testing the Application Bridge Scenario 2. 26
Step 1: Running the application. 26
Hands-on 9 checkpoint: 28
Tutorial checkpoint:...28
Troubleshooting tips: 28

Introduction:

The IBM WebFacing Tool in Host Access Transformation Services (HATS) for System i toolkit allows you to convert the user interface of your i5/OS business applications into a Web user interface. Your converted i5/OS DDS display file source members are deployed as a Web application that communicates with your original program logic when accessed from a browser. The development-time conversion of your DDS application allows you to further develop and customize your application interface and integrate it with Web technologies.

With HATS, you can create HATS projects to transform your critical business applications on the host into Web or rich client applications. The interoperability feature of the HATS toolkit allows you to easily integrate your WebFacing and HATS Web applications when running on the Web.

You may also want to further enhance your i5/OS applications by converting part of your application into a Web application such as EGL. In this case, you can use the **Application Bridge feature** of WebFacing, which allows you to pass data between Web applications such as EGL and HATS or WebFacing applications so that you can take your i5/OS applications even further into modern technology. This goes beyond just getting data from i5/OS in a Web application using database access or by calling a batch program. It allows you to link your new Web application to an interactive i5/OS application and share data.

This tutorial will demonstrate how to setup and use an Application Bridge between WebFacing and an EGL application. The concepts learned in this tutorial can also be applied to HATS applications and is not limited to EGL Web applications. That is you can use the Application Bridge between WebFacing or HATS and an EGL application or any other Web application.

System requirements:

To complete this tutorial you need to have the following tools and components installed:

- IBM Host Access Transformation Services (HATS) for System i V 7.1 product with the WebFacing tool installed and all software updates through the IBM Installation Manager.
- Rational Business Developer (RBD) V 7.1 and Enterprise Generation Language (EGL).
- Rational Developer for System i (RDi)
- i5/OS® V5R3 or greater is required.

Estimated time required: Around 90 minutes.

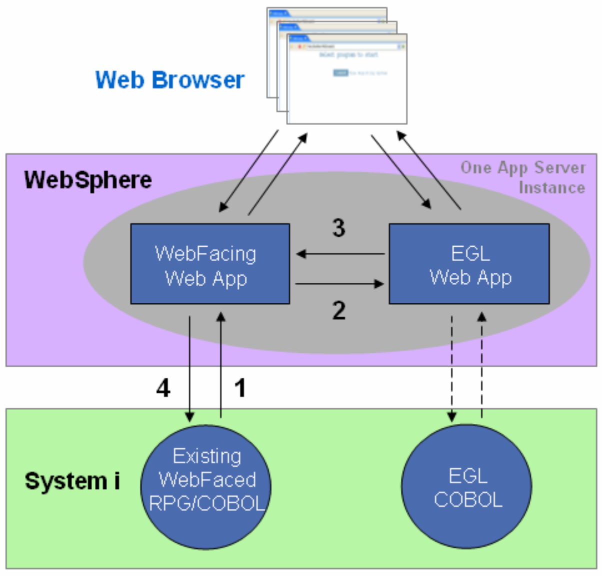
Prerequisites:

- Basic Microsoft® Windows operations
- Basic knowledge of how Web applications work.
- The WebFacing server must be running on your i5/OS. Verify that jobs QQFWFSVR and QQFVTSVR are running. This is done by using the WRKACTJOB JOB(QQF*) command in a 5250 session.
  If it is not started use the STRTCPSVR *webfacing command to start it. Make sure the user ID starting the WebFacing server has *USE authority for *PUBLIC user for object type *LIB and object type *OUTQ.

Hands-on:
In this tutorial you will learn how to build an Application Bridge between WebFacing and an EGL application. You can then use this bridge to communicate or pass data between the two applications.

There are two scenarios applicable to this bridge:

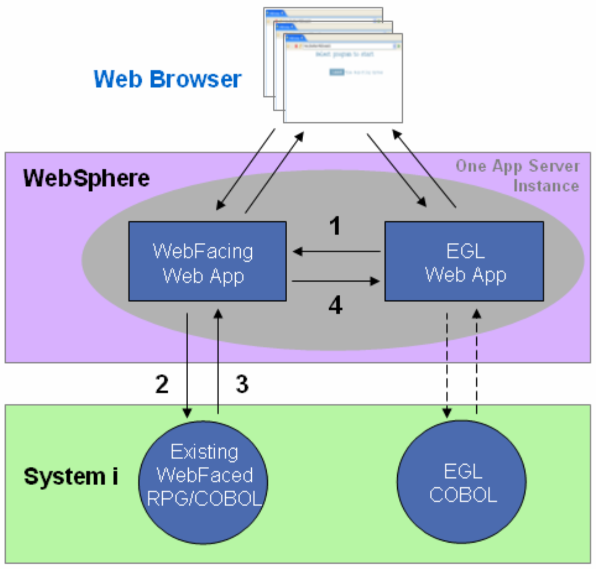
- Scenario 1: WebFacing calls EGL application with data and EGL returns control to WebFacing with data



When WebFacing Web application is running:

1. The host application on System i writes a new DDS record with data
2. WebFacing sends data using request attributes on a forward call to EGL
3. EGL returns control to WebFacing with data updates
4. WebFacing returns data back to the host application on System i

- Scenario 2: EGL calls WebFacing application and WebFacing returns control to EGL with data



When EGL Web application is running:

1. EGL calls WebFacing Web application through programmatic invocation
2. WebFacing invokes the host application on System i
3. The host application on System i writes a new DDS record with FRCDTA keyword specified and then terminates the application
4. WebFacing returns data and control back to EGL using forward

In this tutorial for:

- Scenario 1: WebFacing will pass some customer details to the EGL application. The EGL application will then do a search in the database for the customer and return some additional customer details to the WebFacing application.

- Scenario 2: This is as simple as invoking a WebFacing job using programmatic invocation. WebFacing will then return data and control back to the EGL application using a linkage record.

WebFacing sends data and transfers control to the EGL application when the System i program does a write on a WebFaced linkage record.

You will learn how to create a linkage record, the WebFacing interface for the Application Bridge. You will then run the projects that you have set up and will be able to see the Application Bridge flow.

NOTE: If you face difficulty in any part of the tutorial refer to the Troubleshooting section at the end of the tutorial. Some common problems and their solutions have been documented there.
If you want to look up more information on EGL or WebFacing, refer to the Help Contents or the Tutorials Gallery in the product.
The EGL part of this tutorial builds upon the tutorial "Build a JSF search page with EGL". You can find more information for it in the Tutorials Gallery.

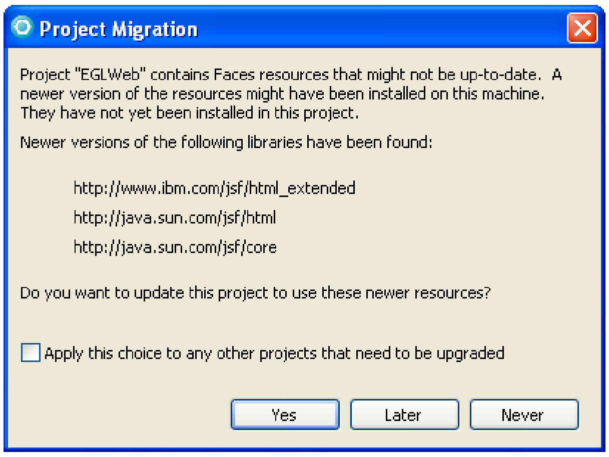
Hands-on 1: Setting up the WebFacing and EGL projects.

In this Hands-on you will set up the WebFacing and EGL projects necessary to run the tutorial.

Step 1: Importing the WebFacing and EGL projects.

1. Download and save the **WFEGLTutorial.zip** file to your local file system.
2. Open HATS Toolkit 7.1 using a new workspace.
   On the desktop task bar click **Start > Programs > IBM Software Development Platform > IBM Rational HATS 7.1 > HATS Toolkit 7.1**
3. Import the projects from the project interchange file. Choose **File > Import,** and then choose **Other > Project Interchange.**
4. Click **Next**.
5. Browse to the location where you stored the project interchange

6. Select **All** projects and click **Finish**.
7. If you get this window choose **Later**.


Project Migration

Project "EGLWeb" contains Faces resources that might not be up-to-date. A newer version of the resources might have been installed on this machine. They have not yet been installed in this project.

Newer versions of the following libraries have been found:

http://www.ibm.com/jsf/html_extended
http://java.sun.com/jsf/html
http://java.sun.com/jsf/core

Do you want to update this project to use these newer resources?

☐ Apply this choice to any other projects that need to be upgraded

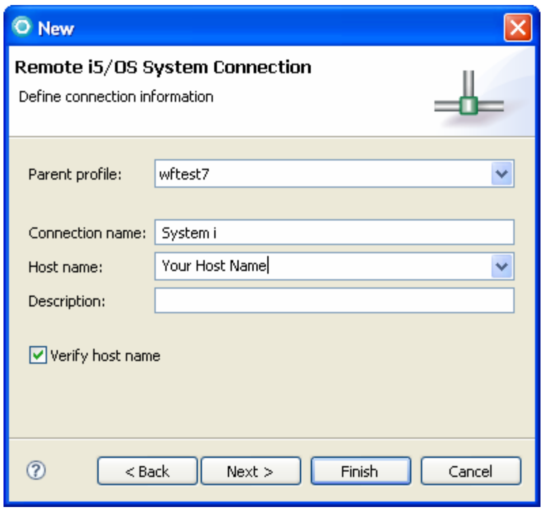
NOTE: There will be some errors in the EGL project. The errors will go away when you are done setting up the project.

Hands-on 2: Creating a connection to your i5/OS, restoring the save file and setting up the WebFacing connection.

Step 1: Restoring the save file.

In this Hands-on you will create a connection to your i5/OS and restore the save file that contains the display file source and the program source.

1. In the **Navigator** view, expand the **WebFacingWeb** project.
2. Right click the **WFEGLTUT.savf** save file and select **Restore on i5/OS**.
3. In the “Specify the i5/OS server to restore to:” section click on the **New** button to create a new connection.
4. Click **Next** on the Name personal profile page.
5. On the Remote i5/OS connection page, first enter the Host Name of your i5/OS.
6. Then enter the connection name as **System i**.
7. The New connection window looks like this:



Parent profile: wftest7

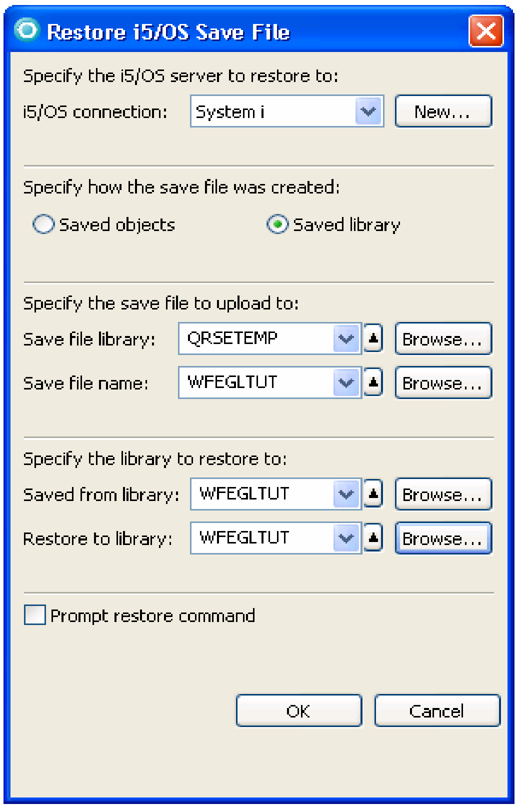
Connection name: System i

Host name: Your Host Name

Description:

[✓] Verify host name

8. Click **Finish**.
9. In the **Saved from library** enter **WFEGLTUT**.
10. In the **Restore to library** enter **WFEGLTUT**.
11. The Restore i5/OS Save File looks like this:



Specify the i5/OS server to restore to:

i5/OS connection: System i

Specify how the save file was created:

Saved objects [✓ Saved library]

Specify the save file to upload to:

Save file library: QRSETEMP

Save file name: WFEGLTUT

Specify the library to restore to:

Saved from library: WFEGLTUT

Restore to library: WFEGLTUT

[ ] Prompt restore command

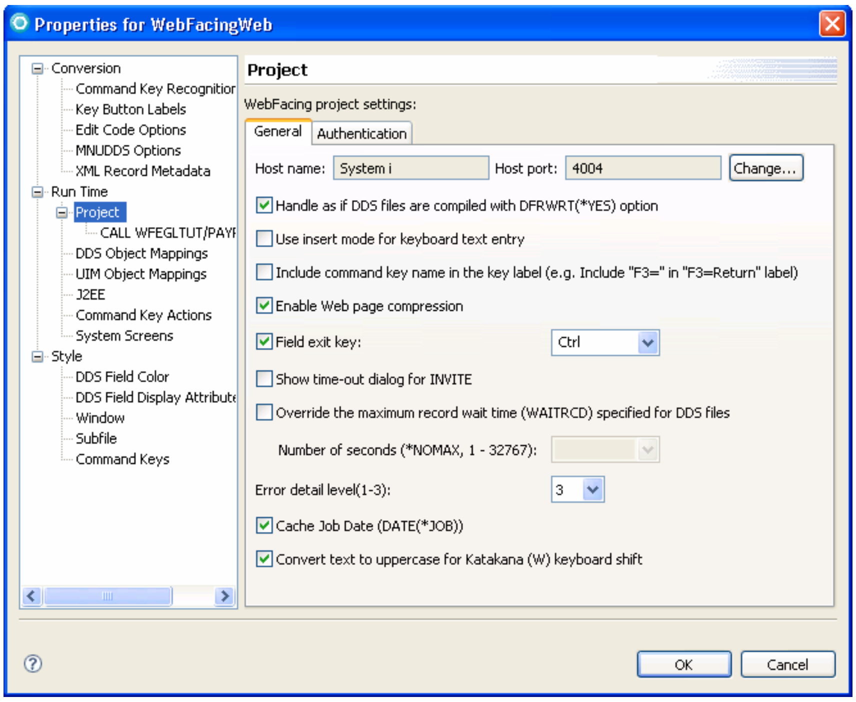
12. Click **OK** to restore the library.
13. Enter user ID and password to the i5/OS when prompted.
14. You may be prompted QRSETEMP does not exist. Create it? Click **Yes.**
15. You should see an information window to indicate that the library is restored properly on the i5/OS.
16. **Important Note:** You need to change the **initial library list** associated with the

user profile used for the WebFacing application to include library **WFEGLTUT**.

Step 2: Specify the i5/OS host name for the WebFacing project.

The WebFacing project needs the i5/OS credentials to be able to communicate with it. Follow the steps below to specify your i5/OS host name for your WebFacing project.

1. Open the **WebFacing** perspective by selecting, **Window > Open Perspective > Other… > WebFacing**.
2. In the WebFacing Projects Explorer view right click on the **WebFacingWeb** project and select **Properties**.
3. On the Project properties page under Run Time->Project, click on **Change…** (on the right beside **Host port**).



Properties for WebFacingWeb

- Conversion
  - Command Key Recognitior
  - Key Button Labels
  - Edit Code Options
  - MNUDDS Options
  - XML Record Metadata
- Run Time
  - Project
    - CALL WFEGLTUT/PAYF
  - DDS Object Mappings
  - UIM Object Mappings
  - J2EE
  - Command Key Actions
  - System Screens
- Style
  - DDS Field Color
  - DDS Field Display Attribute
  - Window
  - Subfile
  - Command Keys

Project

WebFacing project settings:

General | Authentication

Host name: System i   Host port: 4004   Change...

- ✓ Handle as if DDS files are compiled with DFRWRT(*YES) option
- ☐ Use insert mode for keyboard text entry
- ☐ Include command key name in the key label (e.g. Include "F3=" in "F3=Return" label)
- ✓ Enable Web page compression
- ✓ Field exit key: Ctrl
- ☐ Show time-out dialog for INVITE
- ☐ Override the maximum record wait time (WAITRCD) specified for DDS files

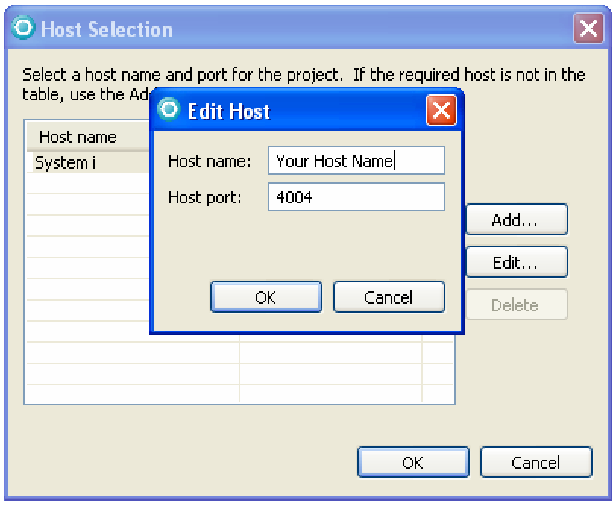
Number of seconds (*NOMAX, 1 - 32767):

Error detail level(1-3): 3

- ✓ Cache Job Date (DATE(*JOB))
- ✓ Convert text to uppercase for Katakana (W) keyboard shift

OK   Cancel

4. In the Host Selection window click on **Edit**.
5. In the Host name text box enter the host name of your i5/OS.



Host Selection

Select a host name and port for the project. If the required host is not in the table, use the Ad

Host name
System i

Add...   Edit...   Delete

OK   Cancel

Edit Host

Host name: Your Host Name

Host port: 4004

OK   Cancel

6. Click **OK** to close the Edit Host window.
7. Again click **OK** to close the Host Selection window.
8. Again click **OK** to close the WebFacing Project properties page.

Hands-on 3: Setting up the database connection for the EGL project.

In this Hands-on, you will setup a database connection for a database that has already been imported into the EGL project for you.
This connection will enable your EGL application to connect to the database both when you design the application (the design-time connection) and when you run the application on the server (the run-time connection).

Step 1: Creating the database connection

1. In the Navigator view, right-click the **EGLWeb** project and then click **Properties**.
2. In the Properties window, click **EGL Runtime Data Source**.
3. On the EGL Runtime Data Source page, select **Load values from a data tools connection** and then click New. The **New Connection** window opens.
4. In the New Connection window, under Select a database manager, expand **Derby** and click **10.1**.
5. In the **Database location field**, select the following folder: *workspace-location*/EGLWeb/WebContent/EGLDerbyR7, where *workspace-location* is the location of your workspace. You do not need to change the User ID or Password fields.
6. In the **Class location field**, enter the path to the file derby.jar.
   For your convenience the derby.jar has been place in your EGL project, so the path to it is *workspace-location*\EGLWeb\derby.jar
   There are other several ways you can find this file:
   - If you have installed IBM® WebSphere Application Server, version 6.1, you can use the version of Derby that is included with the server. Look for derby.jar in the following folder:
     install_location/runtimes/base_v61/derby/lib
   - If you have installed database tools along with your product, you may be able to find the file in the following location:
     shared_resources/plugins/com.ibm.datatools.db2.cloudscape.driver_version/driver
     shared_resources
       The shared resources directory for your product, such as C:\Program Files\IBM\SDP70Shared on a Windows system or /opt/IBM/SDP70Shared on a Linux system. If you installed and

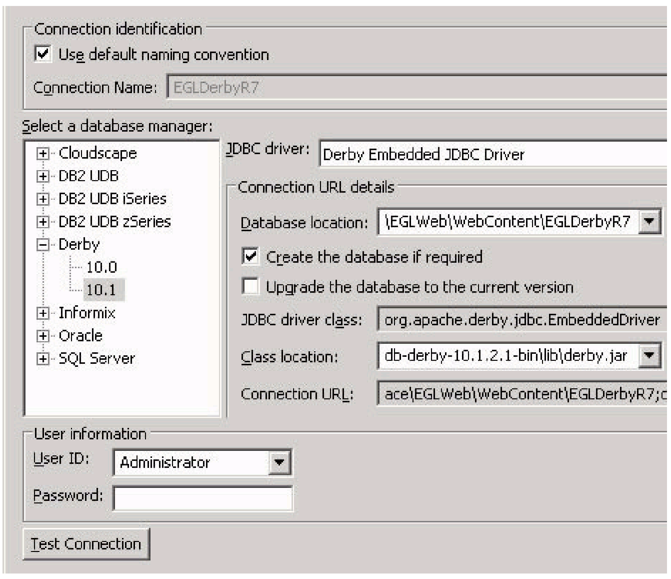
kept a previous version of an IBM product containing EGL before installing your current product, you may need to specify the shared resources directory that was set up in the earlier install.

version

The installed version of the plug-in, including three numbers separated by periods, a string separator, and the date and time that the plug-in was built; for example, 7.0.0.RFB_20070120_1300. If more than one is present, use the one with the most recent version number, unless you have a reason to use an older version.

- If you cannot find the file in the previous two locations, try searching for the derby.jar file in your product installation directory. Different installations may have the file in different locations.
- If you cannot find the file on your computer, you can download the file directly from the Derby Web site: http://db.apache.org/derby/. You will need to download the most recently released version of Derby and extract the derby.jar file to a place on your computer.

The New Connection window looks like this, with your own workspace and location information in the Database location and Class location fields:



Connection identification

☑ Use default naming convention

Connection Name: EGLDerbyR7

Select a database manager:

- Cloudscape
- DB2 UDB
- DB2 UDB iSeries
- DB2 UDB zSeries
- Derby
  - 10.0
  - 10.1
- Informix
- Oracle
- SQL Server

JDBC driver: Derby Embedded JDBC Driver

Connection URL details

Database location: \EGLWeb\WebContent\EGLDerbyR7

☑ Create the database if required

☐ Upgrade the database to the current version

JDBC driver class: org.apache.derby.jdbc.EmbeddedDriver

Class location: db-derby-10.1.2.1-bin\lib\derby.jar

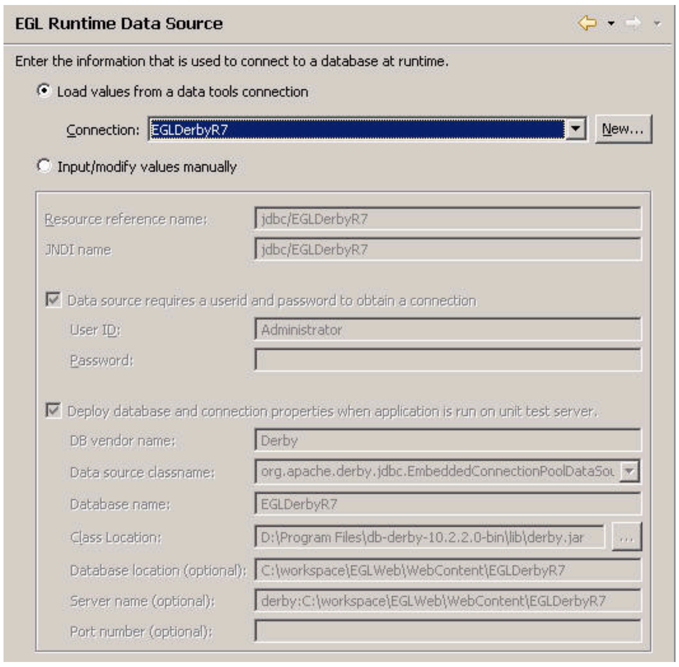
Connection URL: ace\EGLWeb\WebContent\EGLDerbyR7;c

User information

User ID: Administrator

Password:

7. Click **Finish**. The new connection is created and the necessary information for the connection is filled into the fields below:


EGL Runtime Data Source

Enter the information that is used to connect to a database at runtime.

(●) Load values from a data tools connection

Connection: EGLDerbyR7 [New...]

( ) Input/modify values manually

Resource reference name: jdbc/EGLDerbyR7

JNDI name: jdbc/EGLDerbyR7

☑ Data source requires a userid and password to obtain a connection

User ID: Administrator

Password:

☑ Deploy database and connection properties when application is run on unit test server.

DB vendor name: Derby

Data source classname: org.apache.derby.jdbc.EmbeddedConnectionPoolDataSou

Database name: EGLDerbyR7

Class Location: D:\Program Files\db-derby-10.2.2.0-bin\lib\derby.jar

Database location (optional): C:\workspace\EGLWeb\WebContent\EGLDerbyR7

Server name (optional): derby:C:\workspace\EGLWeb\WebContent\EGLDerbyR7

Port number (optional):

Most of this connection information comes from the information that you entered in the New Connection window. In addition, EGL has given this connection a *JNDI name*, which is an identifier for the connection. By default, the JNDI name is jdbc/EGLDerbyR7, based on the name of the database. The application will use this name to access the database connection at run time.

8. Click **OK**.
9. You may see a window asking if you want to update the information in the project's build descriptor options based on this connection. If you see this window, click **Yes**.

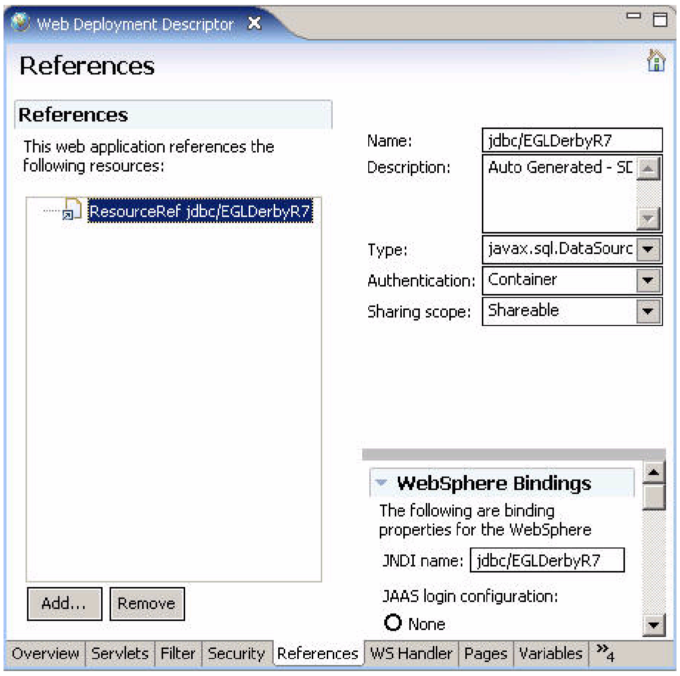
Hands-on 3 checkpoint:

In this Hands-on, you set up a database connection for the project. When you used the EGL Runtime Data Source page of the project's Properties window, you first created a design-time connection to the database using the workbench's data tools. EGL used the information in this design-time connection to create a matching connection to be used at run time for the WebSphere Application Server. In this case, the changes EGL made to your projects include:

- EGL set the values of certain database-related build descriptor options, as explained earlier in the Hands-on.
- EGL created a *JNDI name* to use as a name for the connection. By default, the JNDI name created for your project is jdbc/EGLDerbyR7, based on the name of the database.
- EGL added a *data source* to the EAR project's deployment descriptor. This data source associates the JNDI name with the database itself. Now, other

projects acting as modules within this EAR project can access the database through the JNDI name.

- EGL added a resource reference to that JNDI name in the EGLWeb project's Web deployment descriptor. Now the EGLWeb project can use the data source defined in the EAR project, using the JNDI name.


Web Deployment Descriptor
References
References
This web application references the following resources:
ResourceRef jdbc/EGLDerbyR7
Add...
Remove
WebSphere Bindings
The following are binding properties for the WebSphere
Overview
Servlets
Filter
Security
References
WS Handler
Pages
Variables

Name: jdbc/EGLDerbyR7
Description: Auto Generated - SC
Type: javax.sql.DataSourc
Authentication: Container
Sharing scope: Shareable
JNDI name: jdbc/EGLDerbyR7
JAAS login configuration:
None

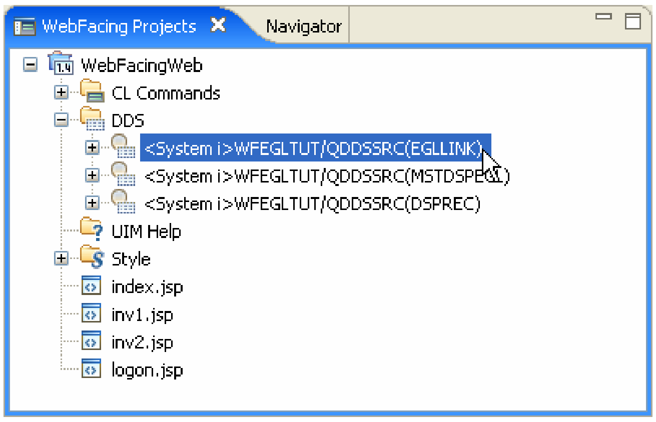
Note: If the projects are deployed to another machine the database connection will need to be setup again.

Hands-on 4: Application Bridge Web Setting.

Step 1: Adding Application Bridge Web Setting.

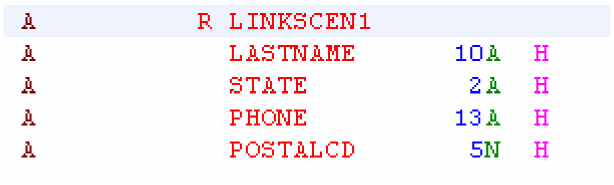
In this Hands-on you will setup the interface needed in a WebFacing project to enable an application bridge between WebFacing and an EGL application.

1. Under the DDS folder in the WebFacing Projects view, open the **EGLLINK** display file. By double clicking on it.


WebFacing Projects
Navigator
WebFacingWeb
CL Commands
DDS
<System i>WFEGLTUT/QDDSSRC(EGLLINK)
<System i>WFEGLTUT/QDDSSRC(MSTDSPEC)
<System i>WFEGLTUT/QDDSSRC(DSPREC)
UIM Help
Style
index.jsp
inv1.jsp
inv2.jsp
logon.jsp

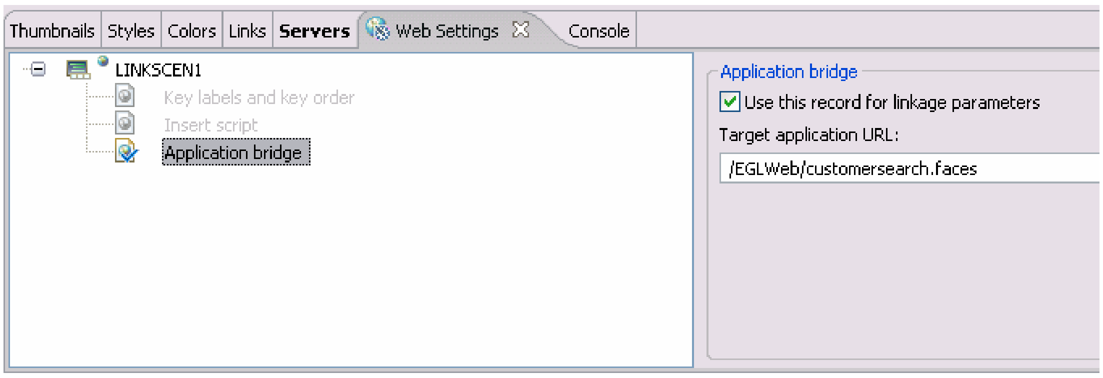
2. You may need to enter the user ID and password for you System i profile.

3. Currently the display file source is empty. You will be adding two new records for the two scenarios described for the Application Bridge in the introduction of this tutorial.
4. Create a new record in the EGLLINK display file with the record name **LINKSCEN1**. This record will be used for transferring control and data to EGL and returning data to WebFacing for scenario 1.
5. Under the DDS folder in the WebFacing Projects view, open the **DSPREC** display file by double clicking on it.
6. Copy the fields defined under the record **DSPREC** in this file with **Usage** (Column 38) **B** (Both) and paste them under record **LINKSCEN1** in the EGLLINK display file.
7. Change the **Usage** of all the fields under record LINKSCEN1 from **B** to **H** (Hidden).
8. The record **LINKSCEN1** looks like this:



```
A            R LINKSCEN1
A              LASTNAME      10A  H
A              STATE          2A  H
A              PHONE         13A  H
A              POSTALCD       5N  H
```

9. Now you will add a record level Web Setting to define that this record is to be used as a linkage record for the Application Bridge.
10. Click on the record **LINKSCEN1** in the editor, this will display the Web Settings that can be specified for this record in the **Web Settings** Tab found at the bottom of the workbench (Click on the Web Settings view at the bottom of the workbench if it is not in focus)
11. In the **Web Settings** tab choose **Application bridge** and **select** the **Use this record for the application bridge parameters**.
12. In the **Target application URL** enter the URL of your application in this format: /appContextRoot/appEntryPoint
    For the tutorial the context root of the EGL project is **EGLWeb** and the entry point in the EGL application for scenario 1 is **customersearch.faces**
    Thus the Target application URL would be
    /EGLWeb/customersearch.faces
    Note: The entry point for the EGL application is actually a customersearch**.jsp** Web page. As it is a JSF page you specify it as customersearch**.faces**
13. The Web Setting looks like this:



LINKSCEN1
- Key labels and key order
- Insert script
- Application bridge

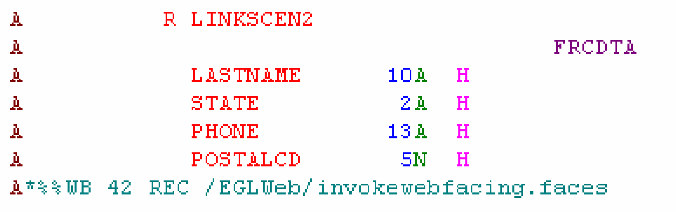
Application bridge

[✓] Use this record for linkage parameters

Target application URL:

/EGLWeb/customersearch.faces

14. Now you will add a Record for scenario 2. Create a record with the record name **LINKSCEN2**. For Scenario 2 FRCDTA keyword must be specified on the record in addition to the Application bridge WebSetting.
15. Copy all the hidden fields from record **LINKSCEN1** to **LINKSCEN2**.
16. Add the **FRCDTA** keyword on the record **LINKSCEN2**.
17. Add an Application bridge Web Setting for the **LINKSCEN2** record with the Target Application URL: **/EGLWeb/invokewebfacing.faces**.
18. The **LINKSCEN2** record looks like this:



```
A          R LINKSCEN2
A                                            FRCDTA
A            LASTNAME       10A   H
A            STATE           2A   H
A            PHONE          13A   H
A            POSTALCD        5N   H
A*%%WB 42 REC /EGLWeb/invokewebfacing.faces
```

19. **NOTE**: The updated display file source has already been compiled for i5/OS and was present in the save file that you had restored. The display file source has also been converted in WebFacing. So you do not have to compile or convert it for the Tutorial. The sole purpose of the above checkpoint is to show you how to create a linkage record.

Hands-on 4 checkpoint:

You can specify an Application bridge Web Setting on any record with all hidden fields. You can then use this record for scenario 1, that is, when you do a WRITE of this linkage record, WebFacing will forward control and data to an EGL application. The WRITE should be followed by a READ on this record, so that the System i application can get the updated data. When the EGL application returns control to WebFacing, the data returned by EGL will be copied to the linkage record by WebFacing.

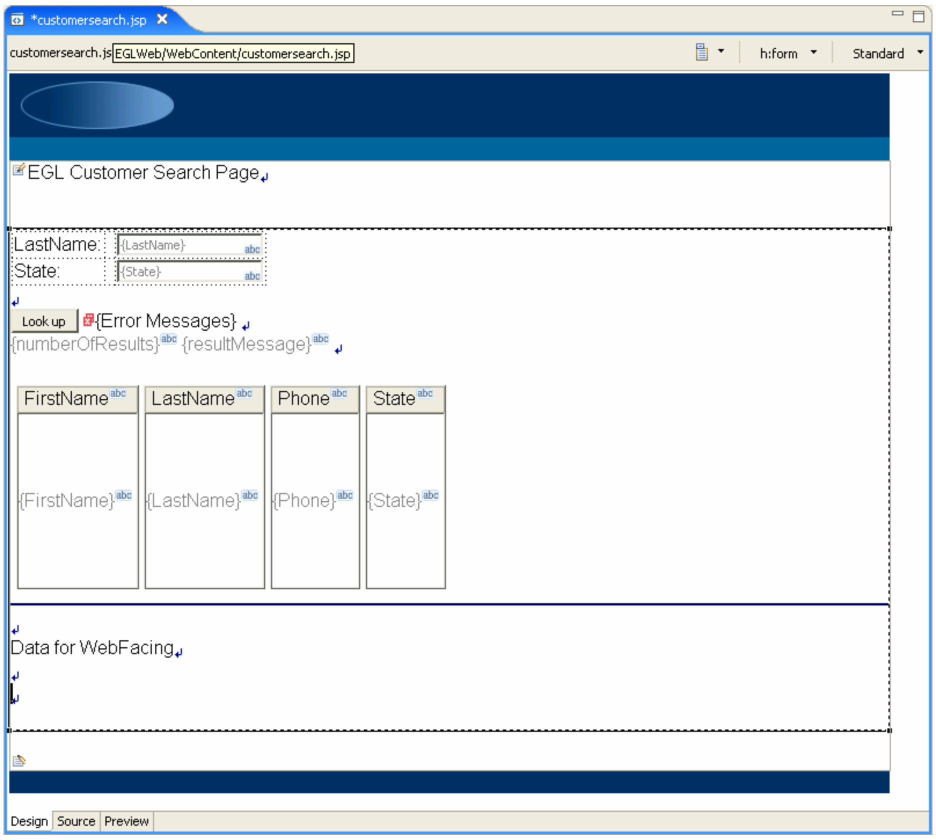
For scenario 2 you need to specify the FRCDTA keyword along with the Application bridge Web Setting on the linkage record to return data and control back to the EGL application after a WebFacing Application is invoked using programmatic invocation.

Hands-on 5: Adding display elements to the EGL Web page.

In this Hands-on you will add display elements that will be used to display the data that would be returned to WebFacing.

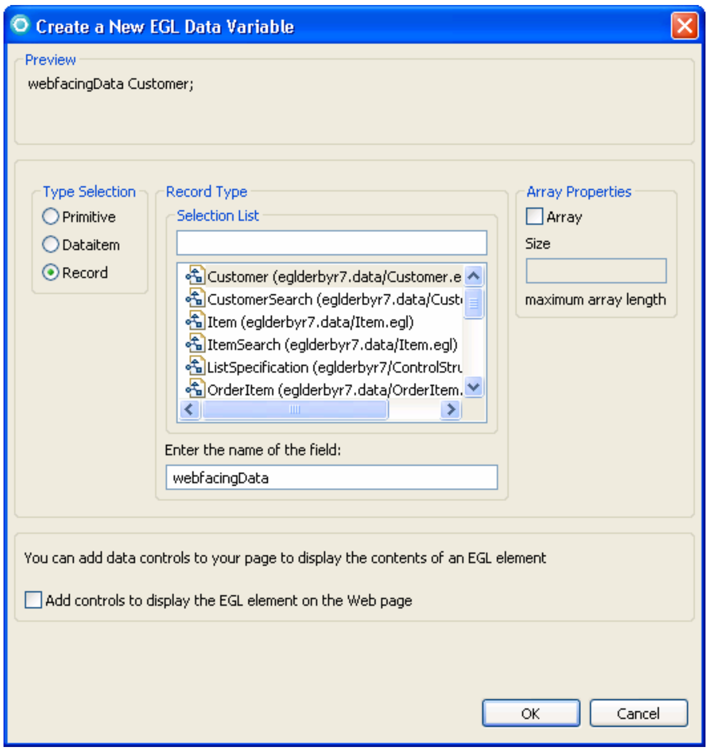
Step 1: Adding the display elements.

1. Open the EGL perspective by selecting **Window > Open Perspective > Other… > EGL.**
2. Open **customersearch.jsp** found under **EGLWeb > WebContent** by double clicking on it.
3. Place cursor at the end, after the horizontal rule and hit Enter three times. After the first line break enter text "Data for WebFacing" as shown below.



4. Find the Palette view, which is typically at the right side of the workbench. If you cannot find that view, click **Window > Show View > Other…** then select **General > Palette**. If the **palette is empty** refer to Troubleshooting section 2.
5. In the Palette view, click the **EGL drawer** to open it.
6. Drag the **New Variable** icon from the Palette view to the customersearch.jsp page in the editor. Create a New EGL Data Variable window opens.

7. Under Type Selection, click **Record**.
8. Under Record Type, click **Customer**.
9. In the Enter the name of the field, type this text: webfacingData
10. Under Array properties, **uncheck** the Array check box.
11. Make sure the **Add controls to display the EGL element on the Web page** check box is not selected.
12. The Create a New EGL Data Variable window looks like this:

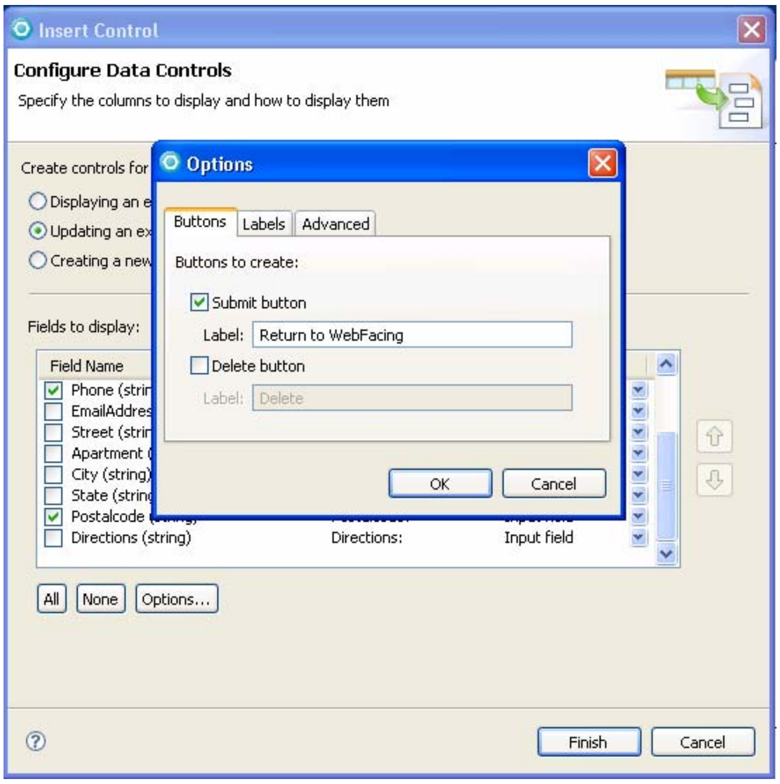

Create a New EGL Data Variable
Preview
webfacingData Customer;
Type Selection
Primitive
DataItem
[✓ Record]
Record Type
Selection List
Customer (eglderbyr7.data/Customer.e
CustomerSearch (eglderbyr7.data/Custo
Item (eglderbyr7.data/Item.egl)
ItemSearch (eglderbyr7.data/Item.egl)
ListSpecification (eglderbyr7/ControlStru
OrderItem (eglderbyr7.data/OrderItem.
Enter the name of the field:
webfacingData
Array Properties
Array
Size
maximum array length
You can add data controls to your page to display the contents of an EGL element
Add controls to display the EGL element on the Web page
OK
Cancel

13. Click **OK**. An item representing the new variable webfacingData appears in the Page Data view under **JSF Handler** > **Data**. If you cannot find that view, click **Window** > **Show View** > **Other…** then select **Web** > **Page Data.**

Step 2: Displaying the data on the Web page.

1. In the Page Data view, click the webfacingData Customer variable to select it.
2. Click and drag the webfacingData Customer variable onto the file customersearch.jsp, releasing it below the Data for WebFacing text.
3. The Insert List Control window opens. This window lists all of the fields in the database record. You can use this window to choose which fields will be shown on the page.
4. Under Create controls for, click the radio button next to **Updating an existing record**.

5. Under Fields to display, click the **None** button. You have deselected all of the fields.
6. Select the check boxes next to these fields:
   - Phone
   - Postalcode
7. Click on the **options** button.
8. Under the buttons tab **uncheck** the **Delete** button.
9. For the **Submit button label** enter Return to WebFacing.
10. The insert controls dialog looks like this:


Insert Control

Configure Data Controls

Specify the columns to display and how to display them

Options

Buttons | Labels | Advanced

Buttons to create:

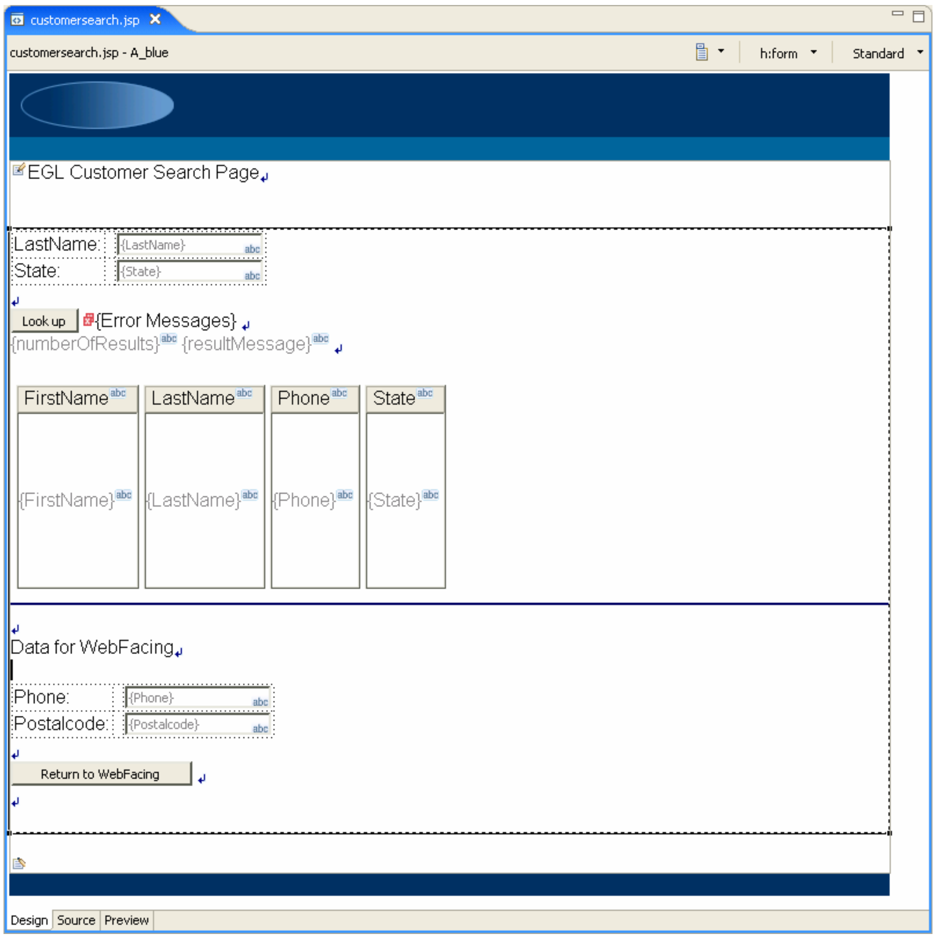
☑ Submit button

Label: Return to WebFacing

☐ Delete button

Label: Delete

11. Click **OK** and then **Finish**.
12. **Save** the page by pressing **Ctrl-S**.
13. The page looks like this:



EGL Customer Search Page

LastName: {LastName}
State: {State}

Look up {Error Messages}
{numberOfResults} {resultMessage}

FirstName	LastName	Phone	State
{FirstName}	{LastName}	{Phone}	{State}

Data for WebFacing

Phone: {Phone}
Postalcode: {Postalcode}

Return to WebFacing

Hands-on 6: Setting up the EGL project for the Application Bridge Scenario 1.

Step 1: Adding the EGL code.

In this hands-on you will setup the interface needed on the EGL side to set up the Application Bridge between the two applications. And you will see how to retrieve and send the Linkage Data.

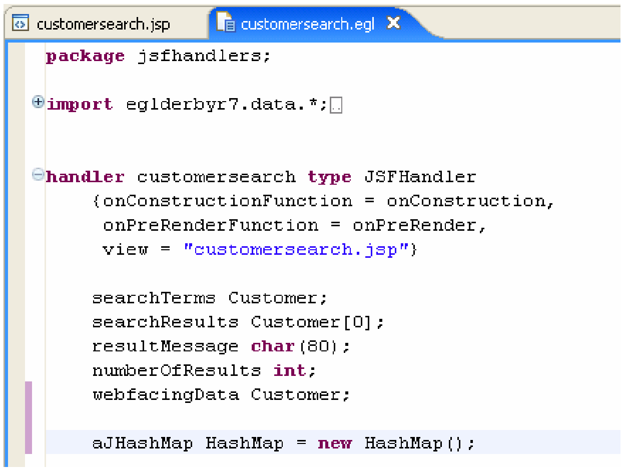
1. WebFacing passes the data to EGL as a Java HashMap Object. This object is set as a request attribute with a key value of “LinkageData”. Thus you need to declare a variable of type HashMap to retrieve the Hash Map.
2. To declare a variable of type HashMap you first need to define a Java HashMap Object. Right click on the **customersearch.jsp** file (in the editor) and select **Edit Page code**. Or open the file customersearch.egl found under EGLWeb > EGLSource > jsfhandlers > customersearch.egl.

   Copy this code after the **last end** in the file:

```
externalType Object type JavaObject

  { JavaName="Object", PackageName="java.lang" }

  function toString() returns ( JavaString );

end
```

```
externalType HashMap type JavaObject
   { JavaName = "HashMap", PackageName = "java.util" }
     function putElement( key String in, value String in ) { javaName="put" };
      function getElement( key String in ) returns ( Object ) { javaName="get" };
end
```

3. Now you can declare a variable of type HashMap in the customersearch handler. After the variable declarations enter this code to declare your HashMap variable: **aJHashMap HashMap = new HashMap();**



```
package jsfhandlers;

import eglderbyr7.data.*;

handler customersearch type JSFHandler
    {onConstructionFunction = onConstruction,
     onPreRenderFunction = onPreRender,
     view = "customersearch.jsp"}

    searchTerms Customer;
    searchResults Customer[0];
    resultMessage char(80);
    numberOfResults int;
    webfacingData Customer;

    aJHashMap HashMap = new HashMap();
```

4. Now you will add code to retrieve the HashMap sent by WebFacing. The HashMap fields follow this convention, Key: the name of the DDS field, Value: the value of the DDS field. Both keys and values are in Unicode string format.
For EGL to do the customer search WebFacing will be passing a customer's last name and state. EGL will then do a search for the customer's Phone and Postal Code and will return this to WebFacing. The code below sets the searchTerms.LastName and searchTerms.State fields from the data passed by WebFacing.
Add this code to the **onPreRender()** function before the last end in the function:

```
hasRequestData    string = getRequestAttr( "forwarded" );
aJHashMap = getRequestAttr( "LinkageData" );
if ( hasRequestData == null || aJHashMap == null )
        resultMessage = resultMessage::" HashMap null or no LinkageData request attribute";
else
        searchTerms.LastName = aJHashMap.getElement( "LASTNAME" ).toString().trim();
        searchTerms.State= aJHashMap.getElement( "STATE" ).toString().toString();
end
```

5. Note that you call the trim() function on the LASTNAME field this is because WebFacing sends data with proper space padding. For example, if the LASTNAME field has length 10 and the field value is “Fili” WebFacing will send the data with six spaces at the end. Thus when WebFacing receives data from other Web Applications it also expects field values of the defined field length. If the length of the field value is not appropriate, the new value is discarded and the old field value is used, that is, if you change the LASTNAME to “Smith” and you do not do proper space padding WebFacing application uses the old value “Fili” not the value passed by EGL or other Web application. The code currently does not pass the LASTNAME back to WebFacing.
6. Note the toString() being called twice in “searchTerms.State= aJHashMap.getElement("STATE").toString().toString();” this is because the first toString converts the object returned by getElement() to a JavaString and the second toString converts the JavaString to an EGL string type.
7. The trim() and toString() helper functions for a JavaString were already defined for you.
8. The onPreRenderfunction() looks like this:

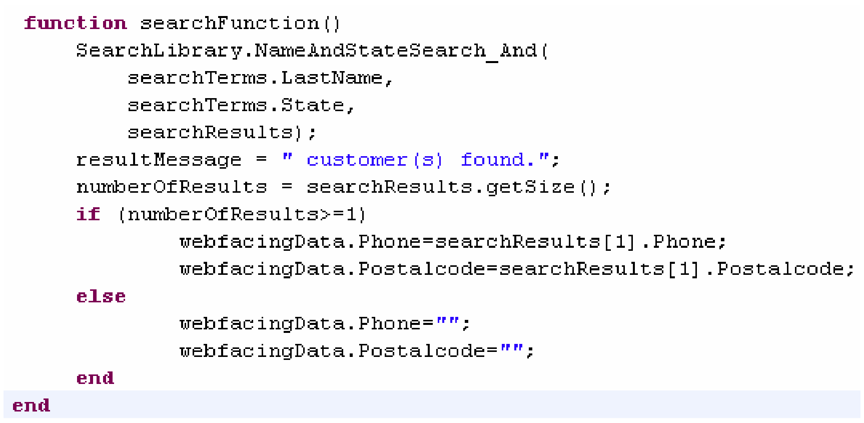
```
function onPreRender()
    if (searchResults.getSize() == 0)
        resultMessage = "No customers found or no search criteria entered.";
    end

    hasRequestData  string = getRequestAttr( "forwarded" );
    aJHashMap = getRequestAttr( "LinkageData" );

    if ( hasRequestData == null || aJHashMap == null )
        resultMessage = resultMessage::"No LinkageData request attribute or Not forwarded";
    else
        searchTerms.LastName = aJHashMap.getElement( "LASTNAME" ).toString().trim();
        searchTerms.State=aJHashMap.getElement( "STATE" ).toString().toString();
        searchFunction();
    end
end
```

9. Now you will edit the **searchFunction()** to display the data in the webFacingData Record you had added to customersearch.jsp. The code below displays the first customer found and this will later be returned to WebFacing. Add this code just **before the last end** of the **searchFunction()**.

```
if (numberOfResults>=1)
        webfacingData.Phone=searchResults[1].Phone;
        webfacingData.Postalcode=searchResults[1].Postalcode;
else
        webfacingData.Phone="";
        webfacingData.Postalcode="";
```

```
end
```

10. The searchFunction () looks like this:

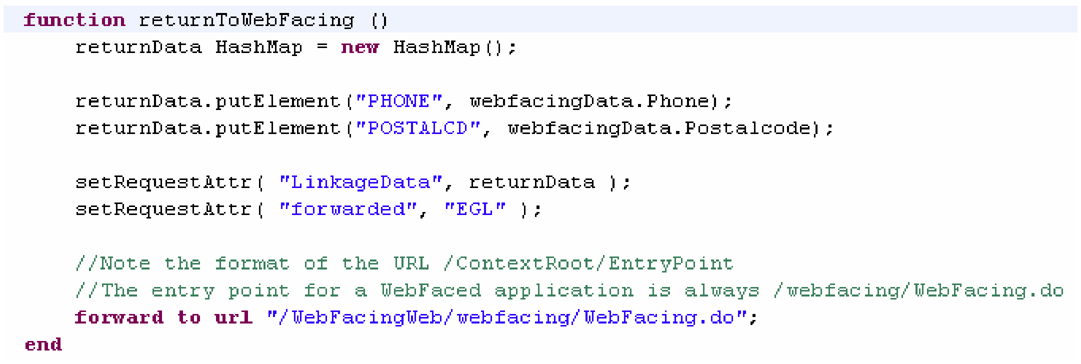


```
function searchFunction()
    SearchLibrary.NameAndStateSearch_And(
        searchTerms.LastName,
        searchTerms.State,
        searchResults);
    resultMessage = " customer(s) found.";
    numberOfResults = searchResults.getSize();
    if (numberOfResults>=1)
            webfacingData.Phone=searchResults[1].Phone;
            webfacingData.Postalcode=searchResults[1].Postalcode;
    else
            webfacingData.Phone="";
            webfacingData.Postalcode="";
    end
end
```

11. Now you will write the function that returns the data you stored in the EGL webfacingData record to the WebFaced application.
12. Create a new function named **returnToWebFacing()**. Put this function following the end of the onConstruction() function.
13. In this function you will set the appropriate data and key values in the HashMap and set the HashMap as the “LinkageData” request attribute.
14. You will also set another request attribute forwarded with value of “EGL”. This will tell WebFacing that the data received is forwarded by EGL.
15. You will then forward the control to the WebFaced application by forwarding to URL /appContextRoot/appEntryPoint. The appContextRoot for the WebFacing project is WebFacingWeb and the appEntryPoint for WebFacing is always /webfacing/WebFacing.do
16. Add this code to the **returnToWebFacing()** function:

```
returnData HashMap = new HashMap();


returnData.putElement("PHONE", webfacingData.Phone);

returnData.putElement("POSTALCD", webfacingData.Postalcode);


setRequestAttr( "LinkageData", returnData );

setRequestAttr( "forwarded", "EGL" );

//Note the format of the URL /appContextRoot/appEntryPoint

//The entry point for a WebFaced application is always /webfacing/WebFacing.do

forward to url "/WebFacingWeb/webfacing/WebFacing.do";
```

17. The returnToWebFacing() function looks like this:



```
function returnToWebFacing ()
    returnData HashMap = new HashMap();

    returnData.putElement("PHONE", webfacingData.Phone);
    returnData.putElement("POSTALCD", webfacingData.Postalcode);

    setRequestAttr( "LinkageData", returnData );
    setRequestAttr( "forwarded", "EGL" );

    //Note the format of the URL /ContextRoot/EntryPoint
    //The entry point for a WebFaced application is always /webfacing/WebFacing.do
    forward to url "/WebFacingWeb/webfacing/WebFacing.do";
end
```

18. Save the changes by pressing **CTRL-S**.

Step 2: Binding the returnToWebFacing function to the Web page

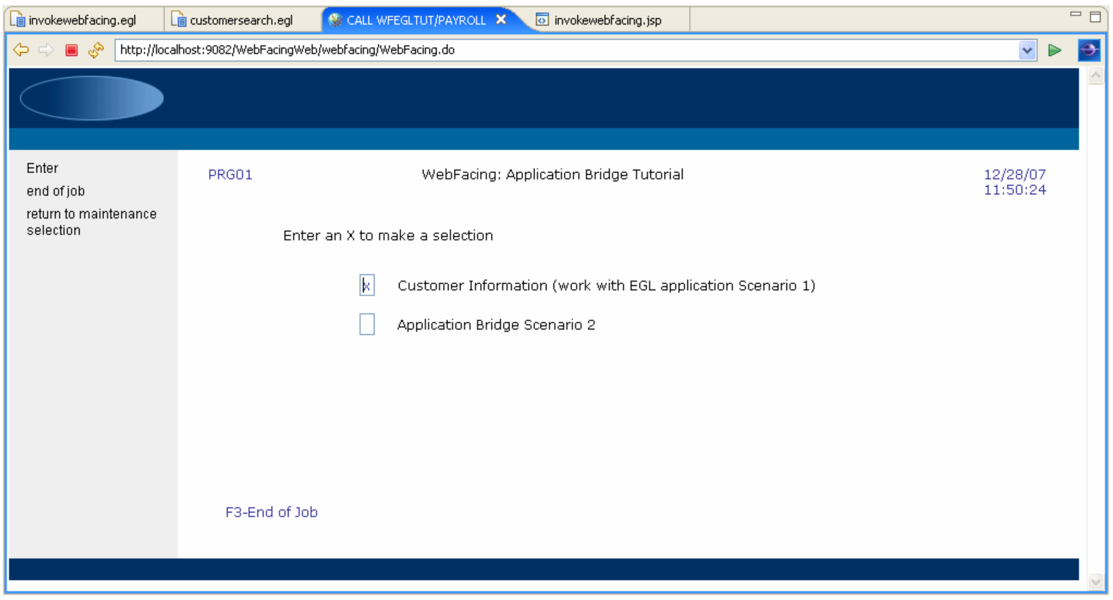
1. Now you will bind the returnToWebFacing() function to the **Return to WebFacing** button you had created on the customersearch.jsp webpage.
2. Open the customersearch.jsp page.
3. In the Page Data view, expand **JSF Handler > Actions**.
4. Drag and drop returnToWebFacing( ) directly onto the Return to WebFacing button on the page. The appearance of the page does not change, but the function is now bound to the button.
5. Save the page by pressing **CTRL-S**.

Hands-on 7: Testing the Application Bridge Scenario 1.

Step 1: Running the Application.

In this hands-on, you will run and test Application Bridge Scenario 1 in the WebSphere test environment.

1. Right click on the **WebFacingWeb** project.
2. From the pop-up menu, select **Run As > Run on Server.**
3. You will see the **Run on Server** window select **WebSphere Application Server v6.1** and click **Next**.
4. On the Add and Remove projects page select the EGLWeb project and click **Add >,** so that the configured projects column has both the WebFacing and the EGL project.
5. Click **Finish**.
6. The index.jsp page for WebFacing will come up.
7. Click on Launch CALL WFEGLTUT/PAYROLL button.
8. You will need to enter the user ID and password for your profile.
9. Select Customer Information (work with EGL application Scenario 1) by entering “x” in the entry field and press **Enter**.


invokewebfacing.egl | customersearch.egl | CALL WFEGLTUT/PAYROLL | invokewebfacing.jsp

http://localhost:9082/WebFacingWeb/webfacing/WebFacing.do

Enter
end of job
return to maintenance selection

PRG01 WebFacing: Application Bridge Tutorial 12/28/07 11:50:24

Enter an X to make a selection

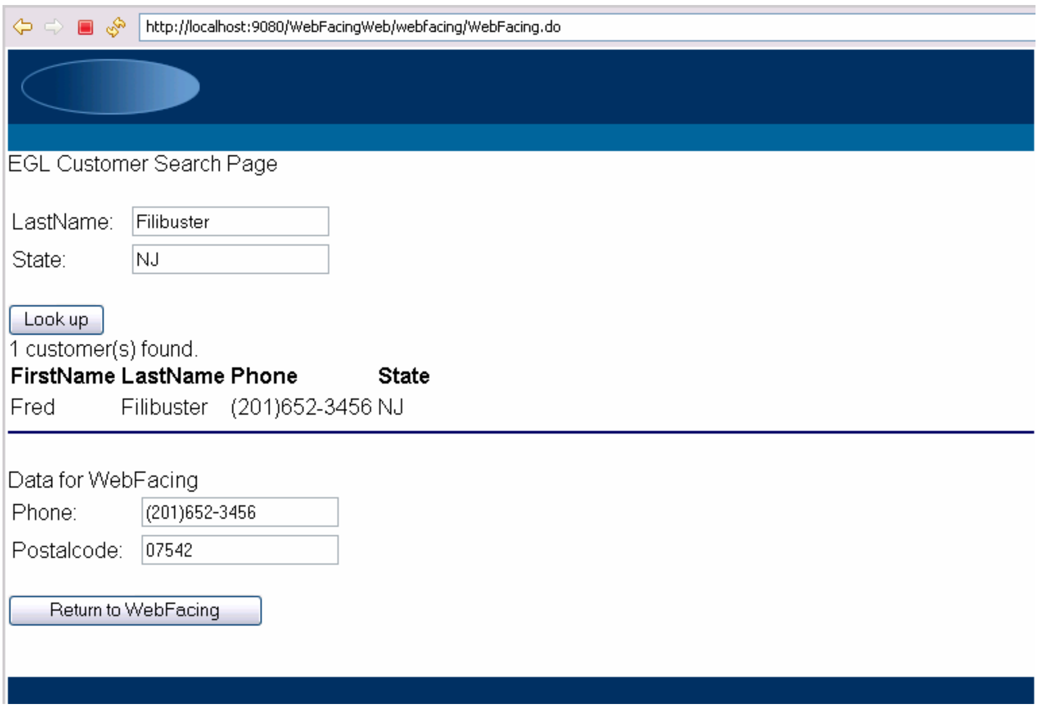
[x] Customer Information (work with EGL application Scenario 1)

[ ] Application Bridge Scenario 2

F3-End of Job

10. The Customer Information page with one customer's information will come up.

11. Press **Enter.**
12. After Enter is pressed, the application program copies the data read from the above displayed record to the record LINKSCEN1 and does a write on it. Since you had specified an Application Bridge Web Setting on this record WebFacing detects that a WRITE to this record was done and this is a linkage record. Thus it transfers control and the data in the record to the EGL application.
13. The EGL code you had specified retrieves the linkage record and sets the search terms Last Name and State.
14. **Click on the Lookup** button to search for the customer in the database.
15. The EGL page displayed is:


http://localhost:9080/WebFacingWeb/webfacing/WebFacing.do

EGL Customer Search Page

LastName:	Filibuster
State:	NJ

1 customer(s) found.

FirstName	LastName	Phone	State
Fred	Filibuster	(201)652-3456	NJ

Data for WebFacing

Phone:	(201)652-3456
Postalcode:	07542

If you get a database not found error follow the instructions in Troubleshooting Section number 1.

16. Click on the **Return to WebFacing** button to send the data and control back to WebFacing.
17. On receiving control back WebFacing retrieves the data passed back in the HashMap and puts it in the linkage record LINKSCEN1. The application program then copies the data into another record to display it.

18. Press **Enter** to end the application.

Hands-on 7 checkpoint:

The WebFacing and the EGL projects must be deployed to the same server instance and both the projects must be started on the server for the Application Bridge to work.

The application program needs to do a WRITE on the linkage record (to transfer control and data to EGL) immediately followed by a READ on the linkage record (to get the data received from EGL in the linkage record).
Note: It is not necessary to display the data being passed and received to/from EGL. Only for tutorial purposes the data from the linkage record is copied to a display record and then the display record is displayed. A WRITE on the linkage record transfers control and data to the other application without displaying it. All the fields in a linkage record are of usage H (Hidden).

Hands-on 8: Setting up the Application Bridge Scenario 2.

Step 1: Adding the EGL code.

In this hands-on, you will do the setup for Application Bridge Scenario 2. In Hands-on 4 you have already set up the interface needed at the WebFacing side as a linkage record LINKSCEN2 returning to the EGL Webpage invokewebfacing.faces.
The EGL Webpage has already been setup for you. You will just add a simple call to programmatically invoke the WebFacing job. Note: For this the WebFacing project must be setup for programmatic invocation. This has been done for you in the tutorial project. To learn how to setup projects for programmatic invocation refer to **Programmatically invoking WebFacing applications from other Web applications** in the WebFacing Tool product help.

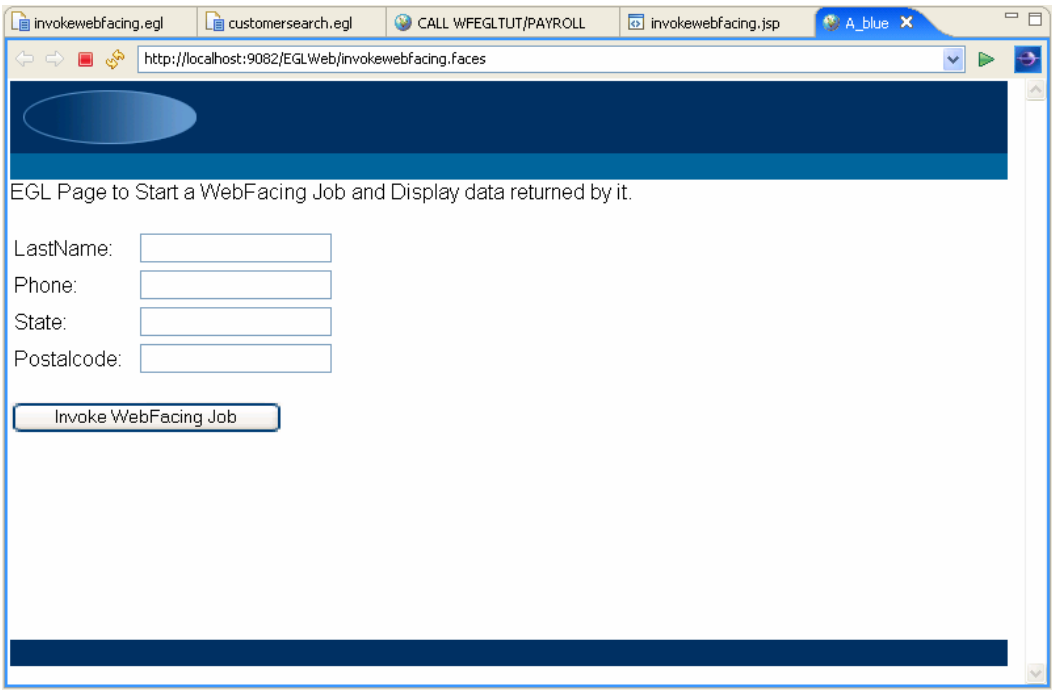
1. Open **invokewebfacing.egl** found under **EGLWeb** > **EGLSource** > **jsfhandlers.**
2. In the **invokeWFPGM**() function enter this code:
   ```
   forward to url "/WebFacingWeb/WFInvocation.do?inv=INV1";
   ```
3. The function invokeWebFacing() has already been bound to the button Invoke WebFacing Job in the EGL page invokewebfacing.jsp
4. Save the change by pressing **CTRL-S**.

Hands-on 9: Testing the Application Bridge Scenario 2.

Step 1: Running the application.

In this hands-on, you will run and test Application Bridge Scenario 2 in the WebSphere test environment.

1. In the Project Explorer view under EGLWeb > WebContent right click on invokewebfacing.jsp and select Run As > Run on Server.
2. The page looks like this:

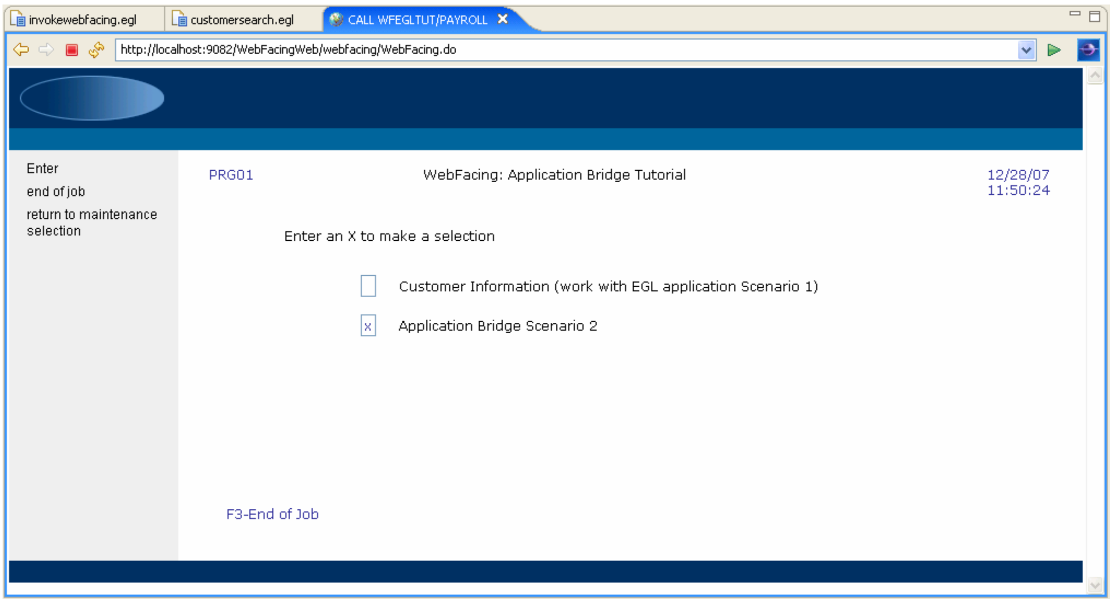


EGL Page to Start a WebFacing Job and Display data returned by it.

LastName:
Phone:
State:
Postalcode:

Invoke WebFacing Job

3. Click on the Invoke WebFacing Job button.
4. The control will then be transferred to WebFacing.
5. Select Application Bridge Scenario 2 by entering “x” in the entry field and press **Enter**.


invokewebfacing.egl
customersearch.egl
CALL WFEGLTUT/PAYROLL
http://localhost:9082/WebFacingWeb/webfacing/WebFacing.do
Enter
end of job
return to maintenance selection

PRG01 WebFacing: Application Bridge Tutorial 12/28/07 11:50:24

Enter an X to make a selection

[ ] Customer Information (work with EGL application Scenario 1)

[x] Application Bridge Scenario 2

F3-End of Job

6. Customer data will be displayed press **Enter**.
7. On pressing enter the application program copies the data from the displayed record to the linkage record LINKSCEN2 and does a WRITE on LINKSCEN2. As this is a linkage record WebFacing transfers control to the EGL application that was specified in the Application Bridge Web Setting.
8. The EGL page displays the data passed by WebFacing.

Hands-on 9 checkpoint:
You learned that WebFacing can be invoked programmatically from EGL. You also learned to transfer control and data to EGL you do a WRITE on the linkage record.

Tutorial checkpoint:

The tutorial demonstrated how to setup and use an Application Bridge between WebFacing and an EGL application. The concept learned in this tutorial can also be applied to create an Application Bridge between WebFacing and any other Web application.

If you are using the interoperability feature of the HATS toolkit to integrate your WebFacing and HATS Web applications, the Application Bridge feature described in this tutorial will also work to pass data between:

- HATS and EGL - when only the linkage record is converted/WebFaced in your WebFacing project
- WebFacing and EGL - when all DDS source files in your WebFacing project have been converted
- WebFacing, HATS, and EGL - when part of your application is WebFaced and the other part runs as a HATS project

This tutorial only covered the Application Bridge feature for WebFacing and EGL as described for Scenario 1 and 2, but it is easy to try out the other scenarios if you link a WebFacing project with a HATS project using the interoperability feature and only convert the linkage record and then convert some of the DDS files in the WebFacing project.

In a linked project again a WebFaced linkage record will be used for the Application Bridge communication and if required, a HATS or WebFaced record could be used to display the data received from the other application. Remember i5/OS V5R4 is required for HATS/WebFacing linked projects.

Troubleshooting tips:

1. Getting error message on running the application: Derby database connection not found.

Note: To just see that the Application Bridge pass data back to WebFacing you can manually enter 5 numeric characters in the postal code field and press the Return to WebFacing button. If you want to do a lookup from the database and then return the data, follow the steps below.

i. Remove the project from the server and restart the server.
ii. Do a clean project on the EGL project
iii. Open the build descriptor for the EGL project by double-clicking the EGLWeb/EGLSource/EGLWeb.eglbld file in the Navigator view. The build descriptor opens in the build parts editor.
iv. Check to see that the EGL Runtime Data Source window has set the build descriptor options based on the connection information. The build descriptor options should look like this:

v. If the build descriptor options do not look like the above screen capture, follow the instructions below.

Setting up the build descriptor options for the database connection:
a. In the Load DB options using Connection list, select your EGLDerbyR7 connection. Several of the options are set, except for the sqlJNDIName option.
b. Set the sqlJNDIName option to the following JNDI name, exactly as shown: jdbc/EGLDerbyR7
Note: To open the sqlJNDIName option for editing, click twice slowly in the Value column next to that option. Also, you can click three times quickly in the Value column.
The values of the build descriptor options now match those described above.
c. Save and close the build descriptor.
Optionally, set the EGL Runtime Database Connection window to make these changes in the future by enabling the associated preference. Click **Window > Preferences** and then click **EGL > Default Build Descriptor**. Under Update default build descriptor options for project when runtime data source is

modified, select **Always** to update the build descriptor options automatically, or select Prompt to give you the option. This preference takes effect the next time you use the EGL Runtime Database Connection window.

2. What if the palette view is empty?

- Switch perspective to any other perspective and then back to the EGL perspective with the JSP file open in the editor. This should fix the problem otherwise see the bullet below.
- Restart the Workbench, that is, close HATS Toolkit and open it again for the same workspace.

3. What if you press the Return to WebFacing button and nothing happens?

- First check if you followed the instructions to bind the returnToWebFacing() function to this button. If you did, check that you copied the code properly.