

SHARE Session #9777: WebSphere and Rational Developer Hands-on-Labs – COBOL and Web services development using RDz

Lab Exercise (duration estimate)

Part 1: COBOL development on System z (~25 min)..... Pg 2

Part 2: Generating, deploying, and testing a CICS Web service (~25 min).....Pg 15

Introduction: About this lab

There are two exercises in this lab.

The first exercise introduces you to the RDz development environment. You will learn how to use RDz to perform a variety of tasks, including authentication to System z, allocation of data sets, editing of COBOL programs, etc. You will also see how RDz can improve your programming efficiency through content assist (i.e. code assist), JCL generation, and the editor's Compare With feature, which lets you visually compare the changes between versions of the same program.

The second exercise introduces you to the Web services support in RDz. Specifically, you will learn how to use the Enterprise Service Tools of RDz to generate a Web service from a COBOL program running on CICS 3.x, then deploy and unit-test it.

Pre-requisites

- Basic to intermediate level understanding of COBOL and System z (OS/390)
- Basic understanding of CICS Web services

Your User id for this lab is: SHARA

Your password is: firstpw

Exercise 1 – COBOL development on the mainframe

In this scenario, you will connect to the mainframe, view your files, allocate a PDS, edit a program, submit a job to compile the program, and view the job output.

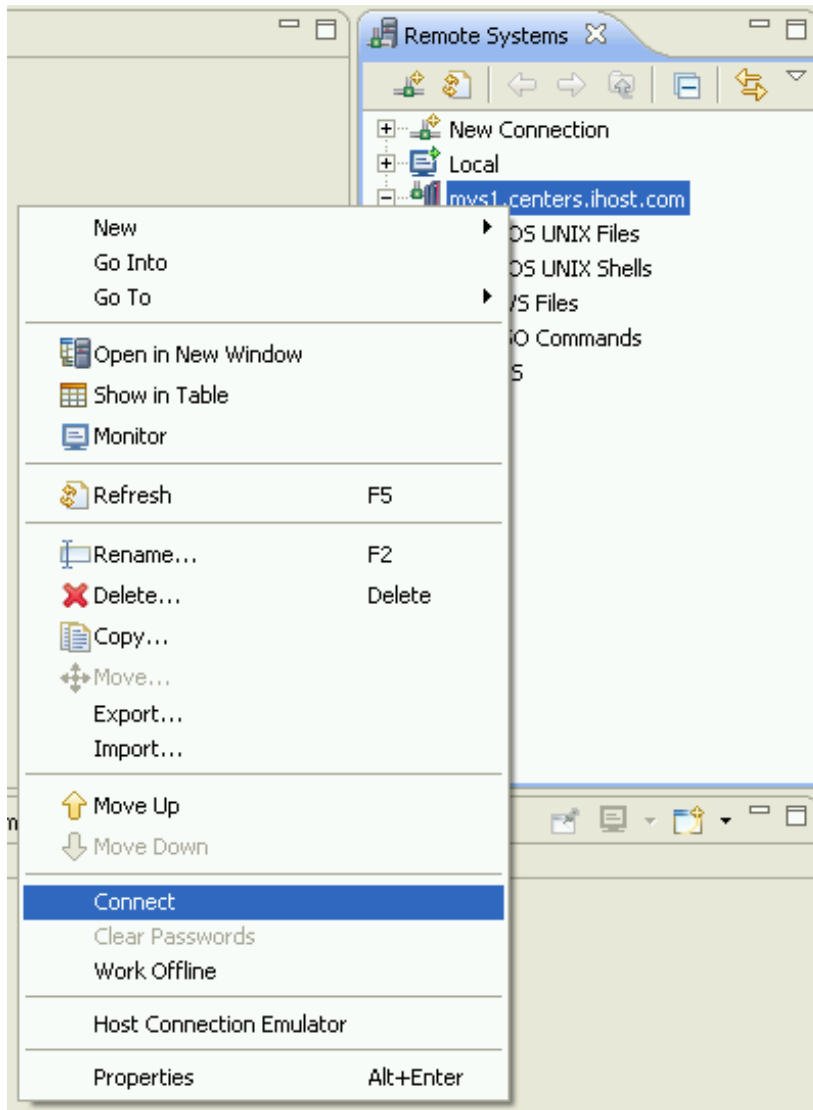
Estimate completion time: 25 minutes

Overview of tasks in this exercise

1. Connect to the SHARE system
2. Explore/Browse data sets
3. Create a filter to view other data sets
4. Allocate a data set
5. Copy program into the newly allocated data set
6. Edit the COBOL program
7. Verify your changes using the Compare editor
8. Generate JCL to compile the program, then view the output
9. (Optional) Build the program

1. Connect to the SHARE system

- If not already done for you, launch RDz by double clicking on the desktop icon “RDz 8.0.1”. On the **Workspace Launcher dialog**, select “C:\workspace\RDzCOBOL”, and click **OK**.
- Once the application is loaded, you will see, in the title area, “z/OS Projects – IBM Rational Developer for System z with Java”. This is the **z/OS Projects perspective**¹.
- On the **Remote Systems** view (right hand side), locate your system connection: mvs1.centers.ihost.com, right-mouse click on it, and select **Connect**



¹ In Eclipse, views (or windows) are organized and laid out as a unit known as *Perspective*. A Perspective is a collection of related views that helps users accomplish specific goals. Some examples of perspective are Java, Debug, z/OS Projects, etc.

- Enter your assigned userid and password. Note that your userid will be assigned to you at the lab. The userid should look something like *SHARAnn*, where *nn* is your unique id. Your password is “*firstpw*”.

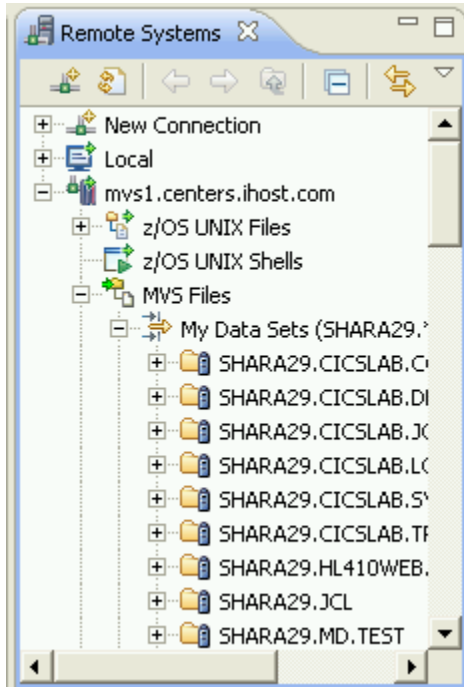


- Once the userid and password are entered, click **OK**. You should now be authenticated to the MVS1 system

2. Explore/Browse data sets

- Expand the “+” sign in front of the connection name, and you will see five different nodes:
 - i. z/OS UNIX Files
 - ii. z/OS UNIX Shells
 - iii. MVS Files
 - iv. TSO Commands, and
 - v. JES
- Expand **MVS Files** by clicking on the “+” sign in front of it. You will see a filter: **My Data Sets** – a filter that is dynamically created when you authenticate to the system.

- Expand **My Data Sets (SHARAnn.*)**, and you should see all the data sets under your high level qualifier SHARAnn.*

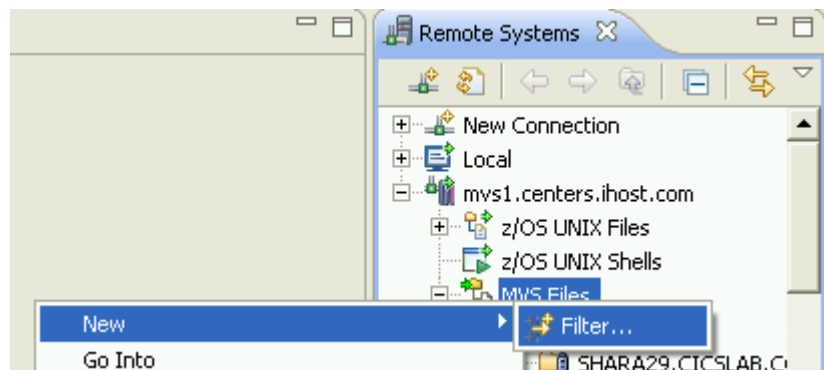


- Expand the various data sets to explore the members and files in the different data sets. (Note: Please do not delete anything unless explicitly instructed to)

3. Create filters to view data sets

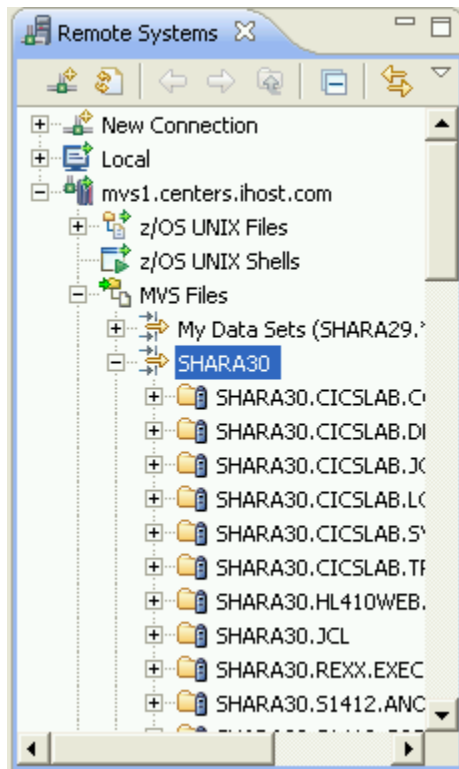
Create a new filter to view another student's (e.g. SHARA30) data sets

- Right-click on **MVS Files**, select **New > Filter...**



- Under **Filter string**, enter *SHARA30.** Click **Next**
- Under **Filter name**: enter a name for this filter. For example, *SHARA30 data sets*. Click **Finish**.

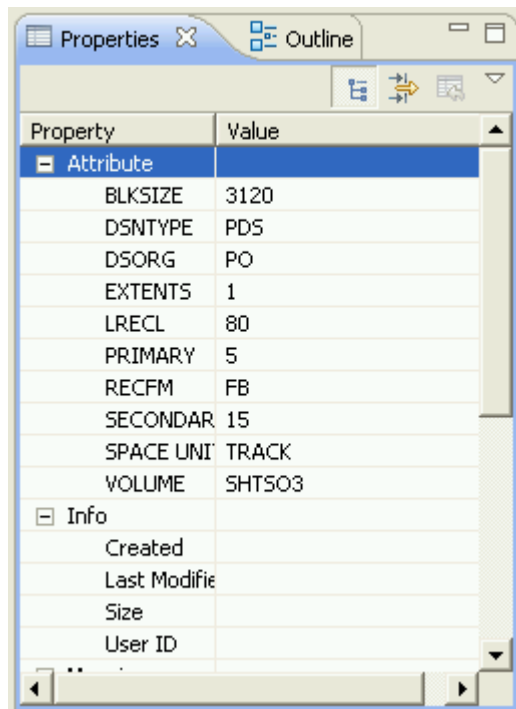
- Expand the newly created filter on the **Remote Systems** view. You should now be able to see the data sets under the id: *SHARA30*. (*Note* that if you are RACF-authorized, you will have the ability to edit the data sets and members)



4. Allocate a data set

You can allocate a data set using the characteristics of an existing data set. In this example, allocate a new COBOL data set using the characteristics of SHARAnn.S1112.COBOL. Name the new data set “SHARAnn.S1112.WORKING.COBOL”, where SHARAnn is your assigned userid.

- i. On the **Remote Systems** view, browse to SHARAnn.S1112.COBOL. Highlight the data set, then look at the **Properties** view, you will see the data set's characteristics:



The screenshot shows a window titled 'Properties' with a sub-tab 'Outline'. It displays a table with two columns: 'Property' and 'Value'. The table is divided into two sections: 'Attribute' and 'Info'. The 'Attribute' section lists various data set characteristics and their values. The 'Info' section lists metadata fields that are currently empty.

Property	Value
Attribute	
BLKSIZE	3120
DSNTYPE	PD5
DSORG	PO
EXTENTS	1
LRECL	80
PRIMARY	5
RECFM	FB
SECONDAR	15
SPACE UNI	TRACK
VOLUME	SHT503
Info	
Created	
Last Modifie	
Size	
User ID	

- ii. Right-click on the data set SHARAnn.S1112.COBOL, select **Allocate Like...** (or press **CTRL+1**)
- iii. Select your high level qualifier (SHARAnn) from the drop down

- iv. Enter “*S1112.WORKING.COBOL*” in the text field to the right of the drop down.

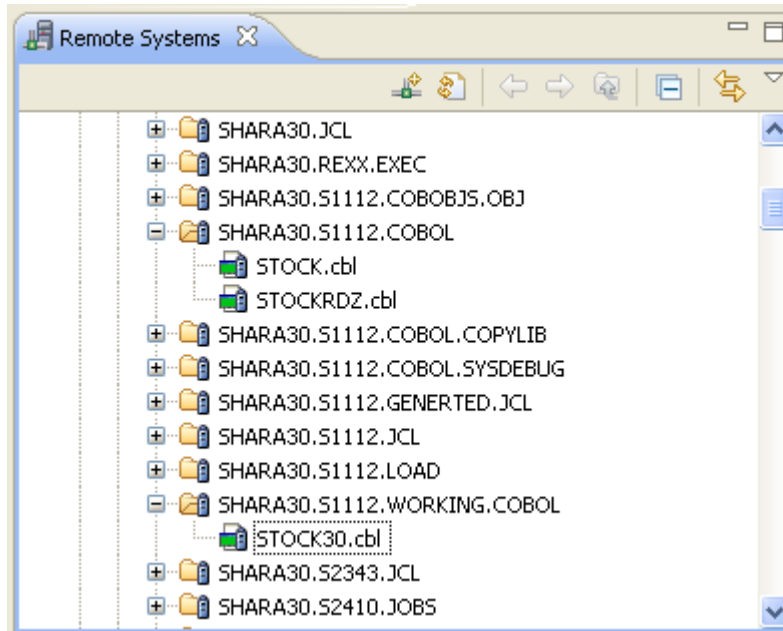
The screenshot shows a 'New Data Set' wizard window. The title bar is blue with the text 'New Data Set' and standard window control buttons. The main area has a light beige background. At the top, it says 'Allocate Partitioned Data Set' and 'Allocate a new partitioned data set residing on z/OS.' Below this, there are two sections: 'Host Name:' with a dropdown menu showing 'mvs1.centers.ihost.com', and 'Data Set Name:' with a dropdown menu showing 'SHARA30' and a text input field containing 'S1112.WORKING.COBOL'. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', 'Finish', and 'Cancel'.

Click **Next**.

- v. On the next screen, you should see that the “Copy characteristics from an existing data set” has been selected. Verify that, and click **Next**.
- vi. On the last screen of this wizard, you will see the data set characteristics of the data set to be allocated. These values are identical to what you saw earlier on the **Properties** view. You have the option to further customize these values if you want to. For this exercise, simply click **Finish**. Your data set is allocated
-

5. Copy program into the newly allocated data set

- Expand SHARAnn.S1112.COBOLE, highlight STOCK.cbl. Right-click on it, and select **Copy (or CTRL+C)**
- Navigate to the newly allocated data set: SHARAnn.S1112.WORKING.COBOLE, right-click on it, select **Paste (or CTRL+V)**. The COBOL program is copied into your data set.
 - i. After copying the member, select **Rename... (or press F2)** from the context menu, and rename the new member to STOCKnn.cbl
 - ii. You should now have one copy of this program in each of the two data sets:



6. Edit the program

- Double-click on `STOCKnn.cbl`. The program will be opened in the System z LPEX editor.

TIP: To maximize the edit window, you can double-click on the title tab of the edit session. To restore the original layout, simply double-click on the title again.

- Make the following changes:
 - i. Update program-id to `STOCKnn` (use your unique id in place of *nn*)
 - ii. On Line 41: `01 EXH-QOT-DAT-RECS REDEFINES COMPANY-STOCK-INFO` is incorrect. Modify it to redefine the correct data definition, as in:

`01 EXH-QOT-DAT-RECS REDEFINES EXH-QOT-DATA.`

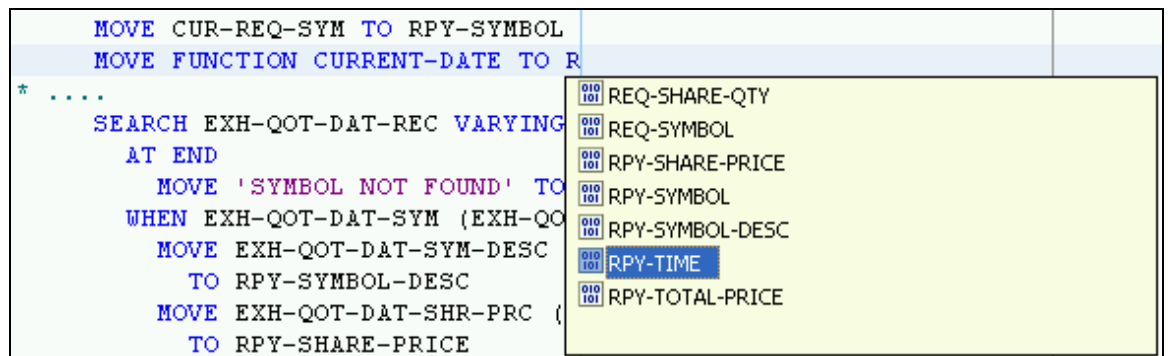
TIP1: You can type the line number directly into the command line area of the editor to jump to the line.

TIP2: Content Assist is a feature that helps you complete your declaration, variables, etc. To invoke, simply press `CTRL+SPACE`. For example, in the above, you can first delete the wrong reference, and when the cursor is at the text insertion point, press `CTRL+SPACE` to see a list of available

- iii. Under the **Procedure Division** section, locate line 95, `MOVE CUR-REQ-SYM TO RPY-SYMBOL`, then insert this new line below it (press `CTRL+ENTER`, or enter “I” in the Prefix area, to add a new line after the existing one):

`MOVE FUNCTION CURRENT-DATE TO RPY-TIME`

Try to use Content Assist in your coding.



The screenshot shows the LPEX editor with the following code:

```
MOVE CUR-REQ-SYM TO RPY-SYMBOL
MOVE FUNCTION CURRENT-DATE TO RPY-TIME
* . . . .
SEARCH EXH-QOT-DAT-REC VARYING
  AT END
  MOVE 'SYMBOL NOT FOUND' TO RPY-SYMBOL
  WHEN EXH-QOT-DAT-SYM (EXH-QOT-DAT-SYM)
  MOVE EXH-QOT-DAT-SYM-DESC TO RPY-SYMBOL-DESC
  MOVE EXH-QOT-DAT-SHR-PRC (EXH-QOT-DAT-SHR-PRC) TO RPY-SHARE-PRICE
```

A Content Assist popup is visible on the right, listing the following options:

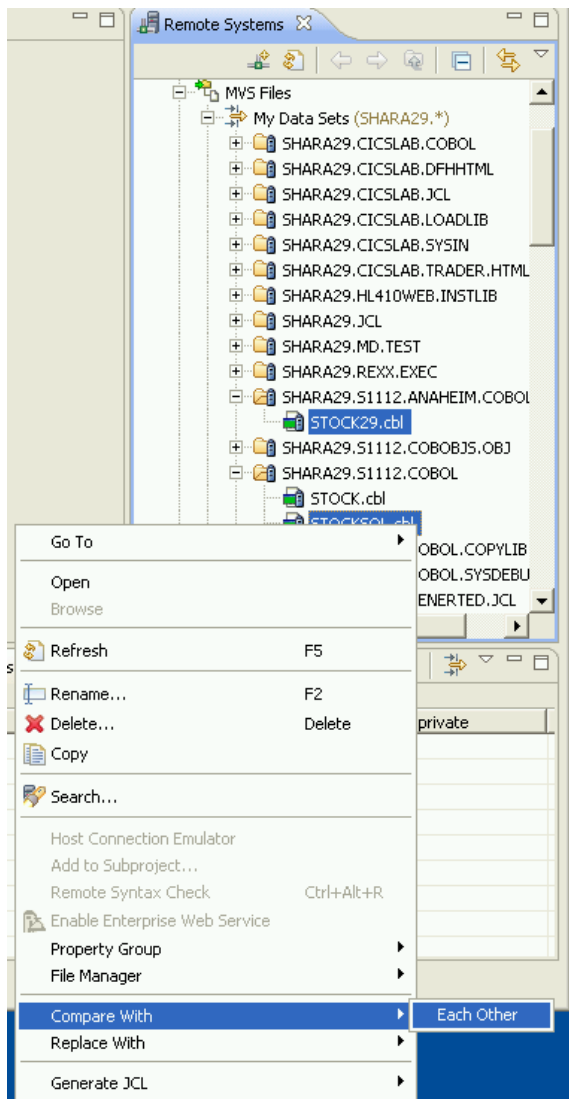
- REQ-SHARE-QTY
- REQ-SYMBOL
- RPY-SHARE-PRICE
- RPY-SYMBOL
- RPY-SYMBOL-DESC
- RPY-TIME** (highlighted)
- RPY-TOTAL-PRICE

- iv. Finally, on the last line of the program, change it to `END PROGRAM 'STOCKnn'`. (use your unique id in place of *nn*)

- Save the changes by pressing “`CTRL+S`”
- Close the program.

7. Verify your changes using the **Compare editor**

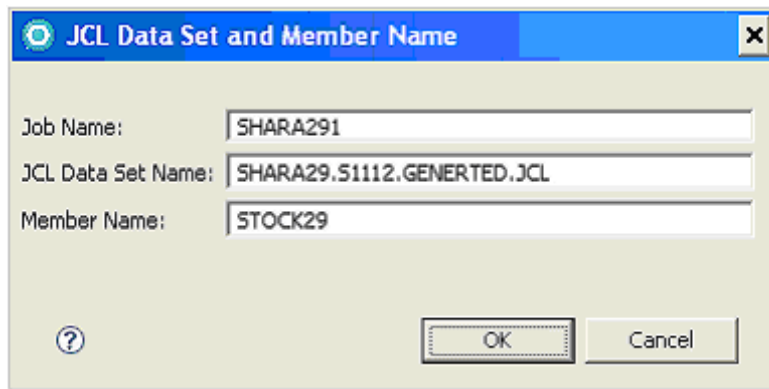
- Go back to the **Remote Systems** view.
- While pressing down the CTRL key, use the mouse to highlight both STOCKnn.cbl and STOCKSOL.cbl (the latter is the “solution” for this exercise, and is found inside the SHAREnn.S1112.COBOLE data set).
- With both highlighted, right-mouse click and select **Compare With > Each Other**. The Compare editor allows you to see the differences between the two files line by line. Verify that the only differences you see are the program IDs. (If you see an error message complaining about incompatible record length, just ignore it and proceed.)



- When you're done, close the editor by clicking on the “x” in the title of the editor window.

8. Generate JCL to compile the program, and view the output

- i. On the **Remote Systems** view, right-click on the STOCKnn.cbl program, select **Property Group > Associate Property Group....** On the pop up dialog, select the checkbox for “*S1112RDz Property Group*”. Click **OK**. RDz will use the build definitions defined in *S1112RDz Property Group* to generate the JCL. (If interested, you can go to the **Property Group Manager** view to open this property group in the editor, so that you can inspect its definition).
- ii. Right-click on the STOCKnn.cbl program, select **Generate JCL > For Compile**. On the pop up, verify that the job name is “SHARAnnI”, and the JCL Data Set name is: “*SHARAnn.S1112.GENERTED.JCL*”. The latter is where the JCL will be generated to. Click **OK**.



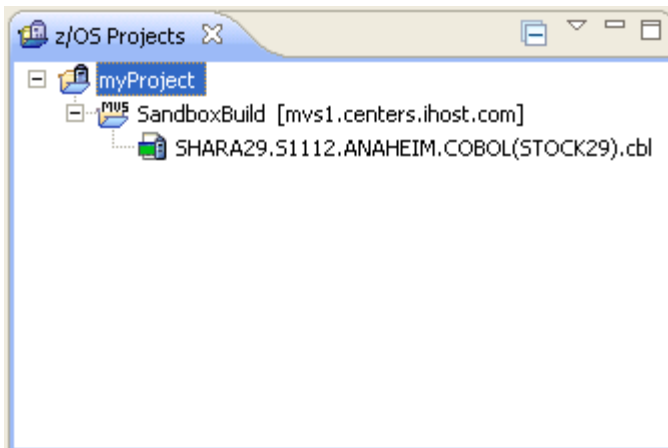
When asked if you want to submit the generated JCL, select **Submit**.

- If the submission is successful, you will see a pop-up dialog, showing you the id for this job. Remember this ID.
 - On the **Remote Systems** view, locate the **JES** node, expand it to see **My Jobs**, then right-click on it, select **Show in Table**. The **Remote System Details** view will come into focus. Locate the job you have just submitted.
 - Under the **User return code** column, you will see the job status of your job. To see the job output, double click on the job. The job output will be shown in the editor window. (Hint, you can reorganize the columns in the table. To do so, you can simply drag and drop a column to the position of your choice.
-

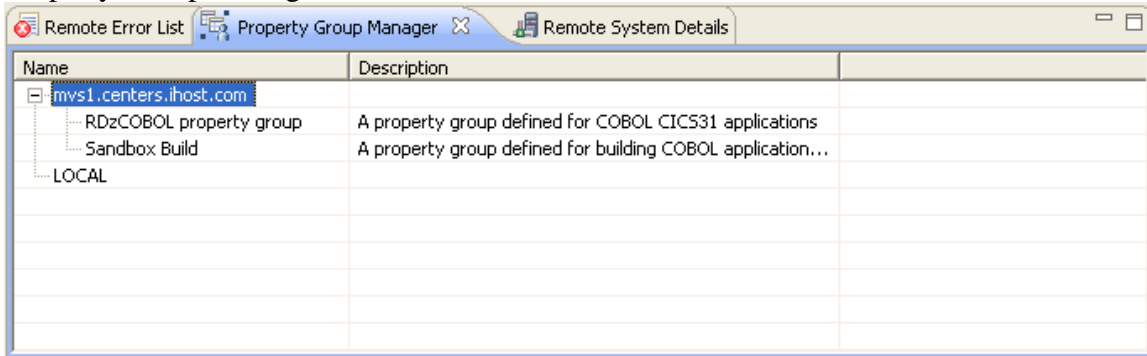
9. (Optional) Building the COBOL program

To build the program you have just written, you will use the **z/OS Projects** capability. Follow the steps below:

- i. Go to the z/OS Projects view (on the left hand side of the z/OS Projects perspective), right-click on the blank space, select **New > z/OS Project....**
- ii. Enter a name for the project, such as “myProject”.
- iii. Under the **Subproject** groupbox below, select to “Create an MVS subproject”. Click **Finish**.
- iv. Enter a name for the subproject, for example, “SandboxBuild”.
- v. Also on this page, select the property group “Sandbox Build” to associate to this subproject. This property group contains the build definition you will need to build this subproject. Click **Finish**.
- vi. You will now see the project on the **z/OS Projects** view.
- vii. Now, return to the **Remote Systems** view, highlight your program “STOCKnn.CBL”, right-click and then select “**Add to subproject...**” (Hint: you can also drag and drop this file into the subproject to add it)
- viii. On the pop-up, select your subproject as the copy destination. Your project should look like this:



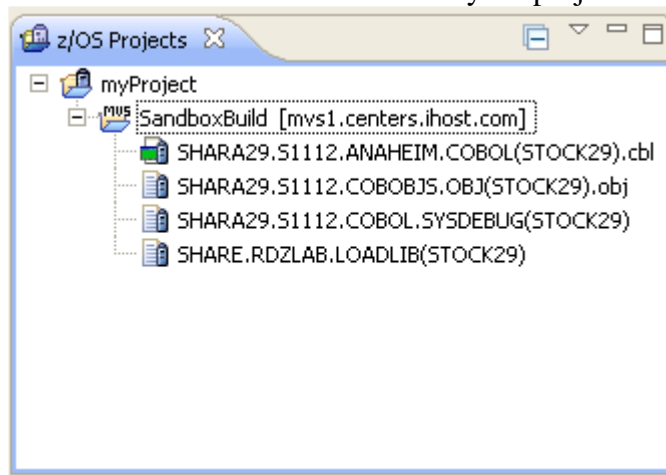
- ix. Take a look at the property group definition before initiating the build. Navigate to the Property Group Manager view



Double-click on “Sandbox Build” to open it in the editor. You can view its various attributes such as its name, selected categories, and the properties used in each of the categories such as COBOL options, the different PROCs used, Link options, etc. When you are finished, close the edit session.

(Shortcut: - You can right-mouse click on your MVS subproject, select **Property Group > Edit Associated Property Group to bring up the edit session)**

- x. Right-click on the subproject “SandboxBuild”, and select “Rebuild subproject”:
- xi. This will generate the JCL and submit a job to build the program. When it is finished, you should see the load module built inside your project:



Exercise 2 – Generating, deploying, and testing a CICS COBOL Web service

In this exercise, you will transform an existing COBOL application to a Web service provider using the Enterprise Service Tools feature of RDz. Specifically, you will generate a Web service for a COBOL application that allows external clients to pass in the stock symbol of a company, and retrieves the stock quote of that company. To simplify things, some sample stock quotes have been hard-coded into the program so that we can easily test the program without database access.

Once the Web service is generated, you will deploy it to a CICS 3.1 runtime (note that CICS 3.2 and CICS 4.1 are also supported), and then test it to verify it works.

Estimate completion time: 25 minutes

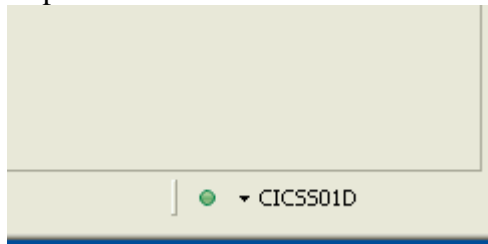
Overview of tasks in this exercise

1. Open the Enterprise Service Tools
2. Create a Web service for CICS project
3. Generate Web service artifacts.
4. Deploy the Web service to CICS
5. Verify the Web service installation
6. Test the Web service

(Note) If not already done so, authenticate to the SHARE system using the **Remote Systems view** (see Pg 3-4)

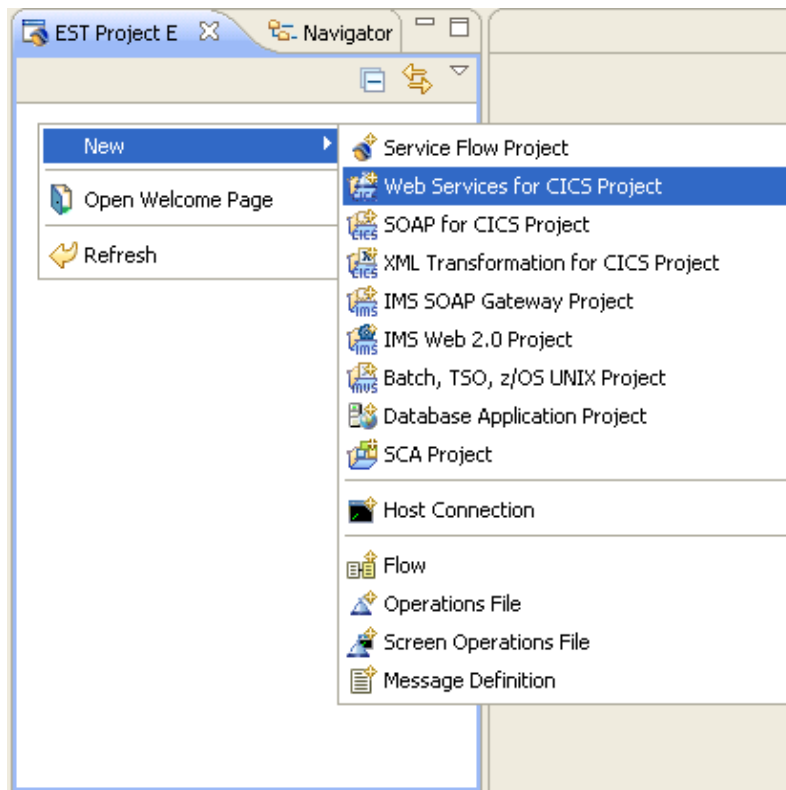
1. Open the **Enterprise Service Tools** perspective

- Select **Window > Open Perspective > Others > Enterprise Service Tools**
- Select **Window > Preferences**. Highlight **Connections** on the left hand side navigation menu. This is the **CICS Explorer** connection page. Currently, there is a connection defined for the region “CICSS01D” (via Port: 9387). On this page, click on the link “Credentials” to update the User ID and password with yours. When you’re finished. Click **OK**.
- On the lower right hand corner of the EST perspective, verify that the CICS Explorer connection is active:

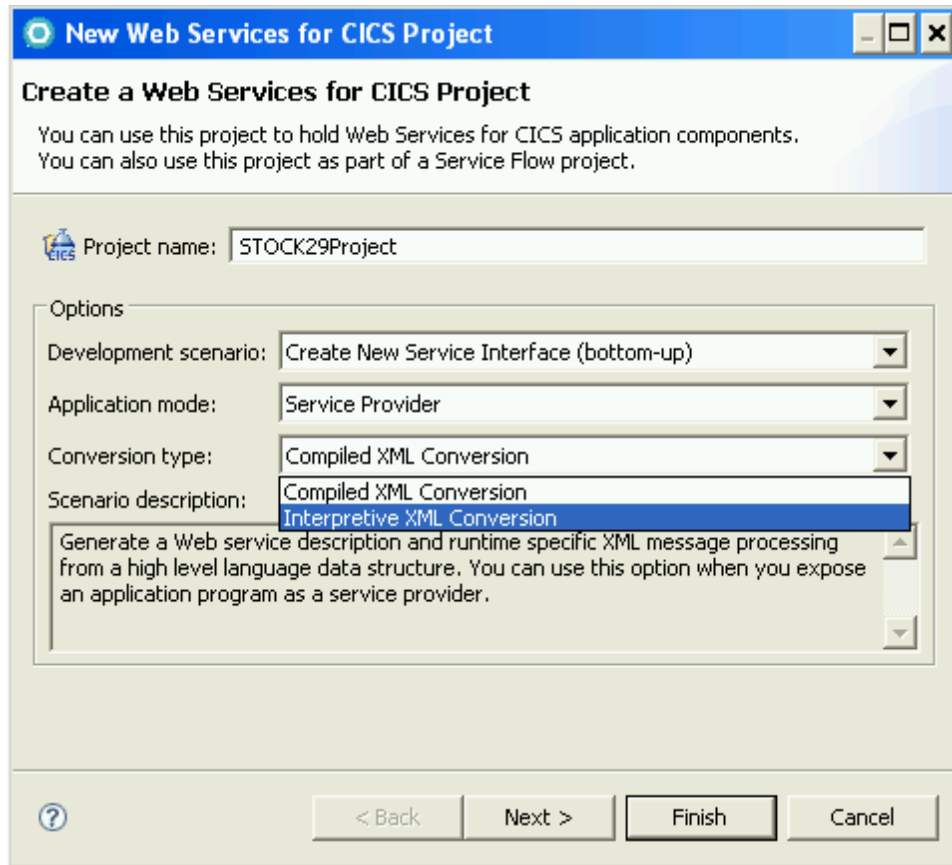


2. Create a **Web services for CICS** project

- On the **EST Project Explorer** view, right-mouse click on the blank space, select **New > Web Services for CICS Project**. This will bring up the project wizard.



- For project name, enter “STOCK nn Project”
- Under Options, verify the following options are selected:
 - a. Development scenario: *Create New Service Interface (bottom-up)*
 - b. Application mode: *Service Provider*
 - c. Conversion type: *Interpretive XML Conversion*



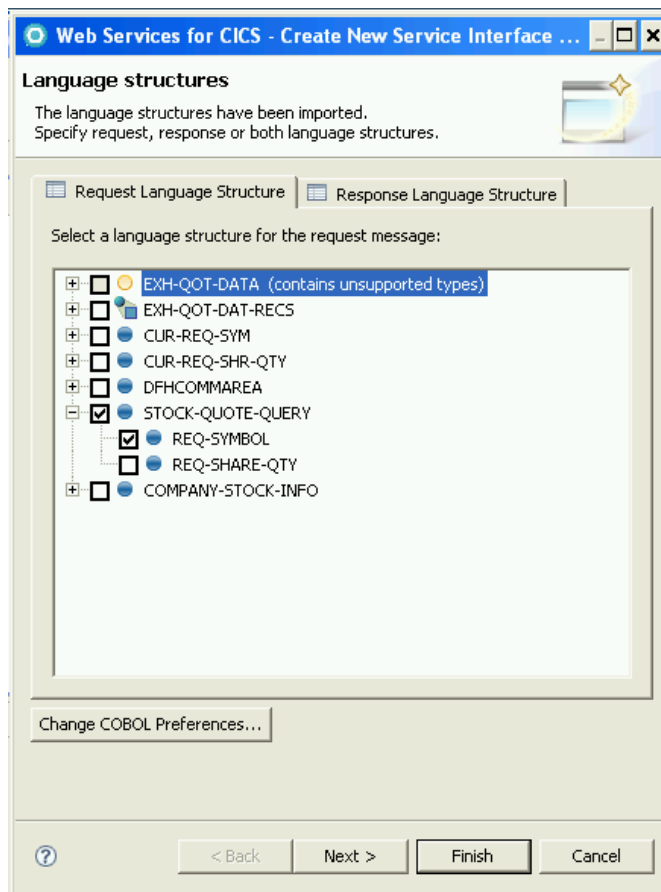
Click **Next**.

- On the Import source files page, select the **Remote...** button to import the source from the mvs1 system.
- Browse to select file. First, expand “mvs1.centers.ihost.com – MVS Files”, then expand “My Data Sets”. Navigate to SHARAnn.S1112.COBOL, and select STOCKSOL.cbl. Click **OK**.
- When asked whether to check dependencies, select “Import the selected source only” since this program doesn’t have any external dependencies. Click **OK**. (Note: If the other option is selected, and if there are external copybooks that this program is dependent on, you will see these identified, and have a chance to import them to the project as well)
- Click **Finish**. Your project is created, and the Web Services for CICS wizard is launched.

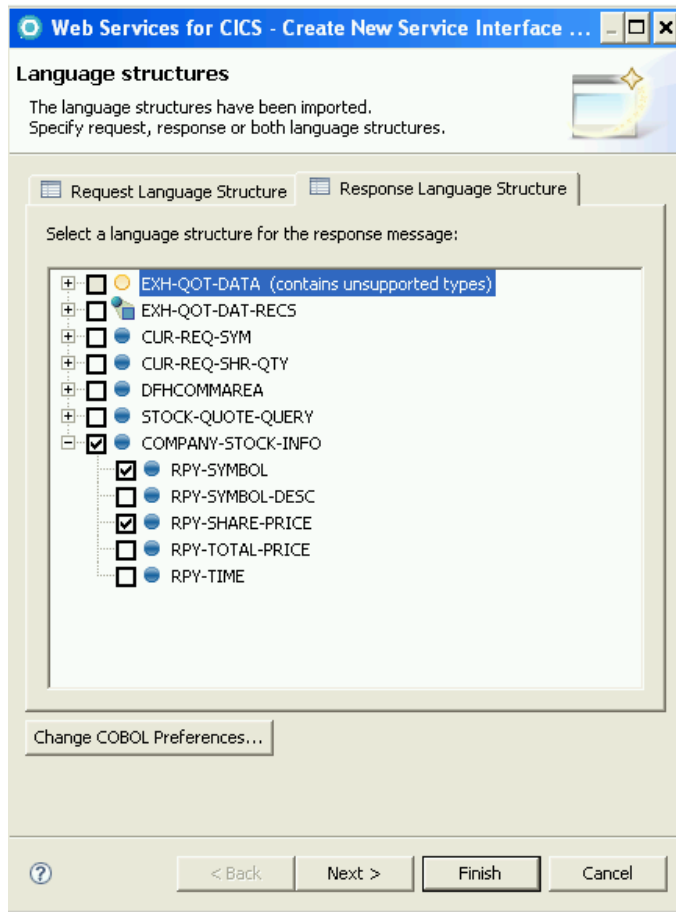
3. Generate Web service

You will generate a Web service for a COBOL application that allows external program clients to query the stock quote of a company. The external application will pass in the stock symbol, and your Web service will return the stock quote using the CICS Web Services support. To simplify the lab, the load module for this COBOL program: STOCKSOL is already built, and is currently running in the CICS region CICSS01D.

- On the first page of the wizard, you are asked to specify the Request and the Response language structures to generate Web service from. Under the **Request Language Structure** tab, expand *STOCK-QUOTE-QUERY* and select the checkbox for REQ-SYMBOL as the input.



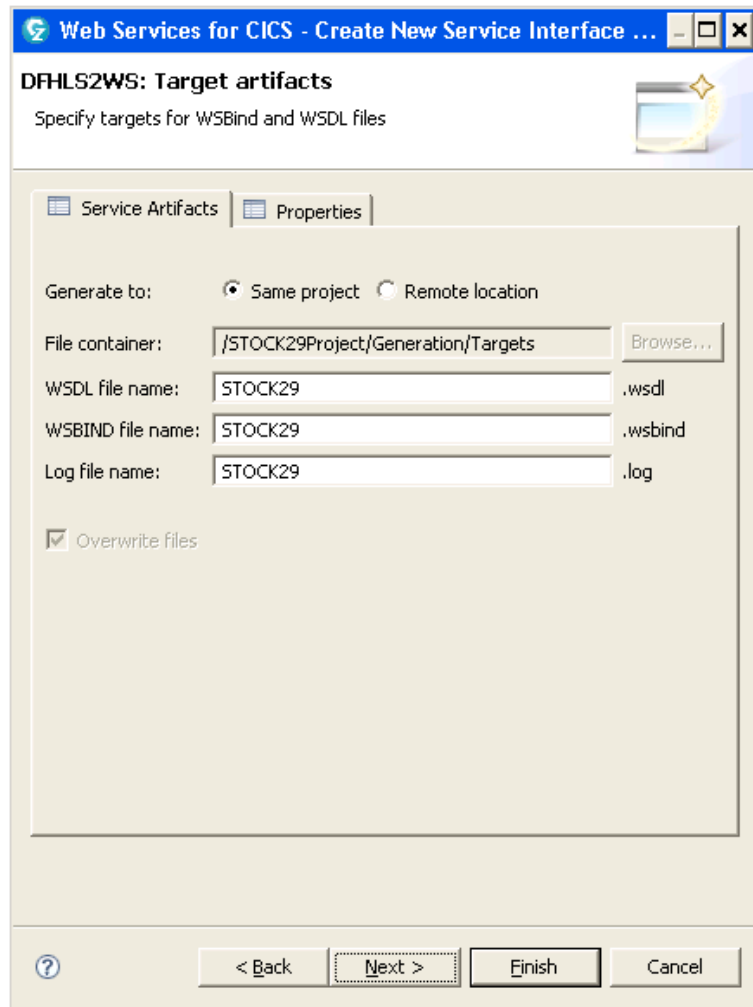
- On the same page, click on the **Response Language Structure** tab, expand *COMPANY-STOCK-INFO*. This data structure contains the information you want to return to the requesting client.



Select the checkboxes for RPY-SYMBOL and RPY-SHARE-PRICE.
Click **Next**.

- Specify the **Application Properties** and the **Service Properties**.
 - Under the **Application Properties** tab, ensure that the program name is *STOCKSOL* – this is the name of the load module running in the CICS region
 - Click on the **Service Properties** tab, and specify the following for **service location**:
<http://mvs1.centers.ihost.com:9387/cics/services/SHARAnn>. This location information entered will be written to the WSDL and WSBIND files that will be generated from this wizard.
 - Click **Next**.

- For the interpretive XML conversion, both a WSDL file² and a wsbind file³ will be generated.
 - Under the **Service Artifacts** tab, rename the WSBIND file name to **STOCK n** . This is needed to avoid name conflicts since all the students are deploying their wsbind files to the same pickup directory.
 - For the WSDL file, you can just accept the default name. You will later use this to test your Web service.



- When you are ready, click **Next**.

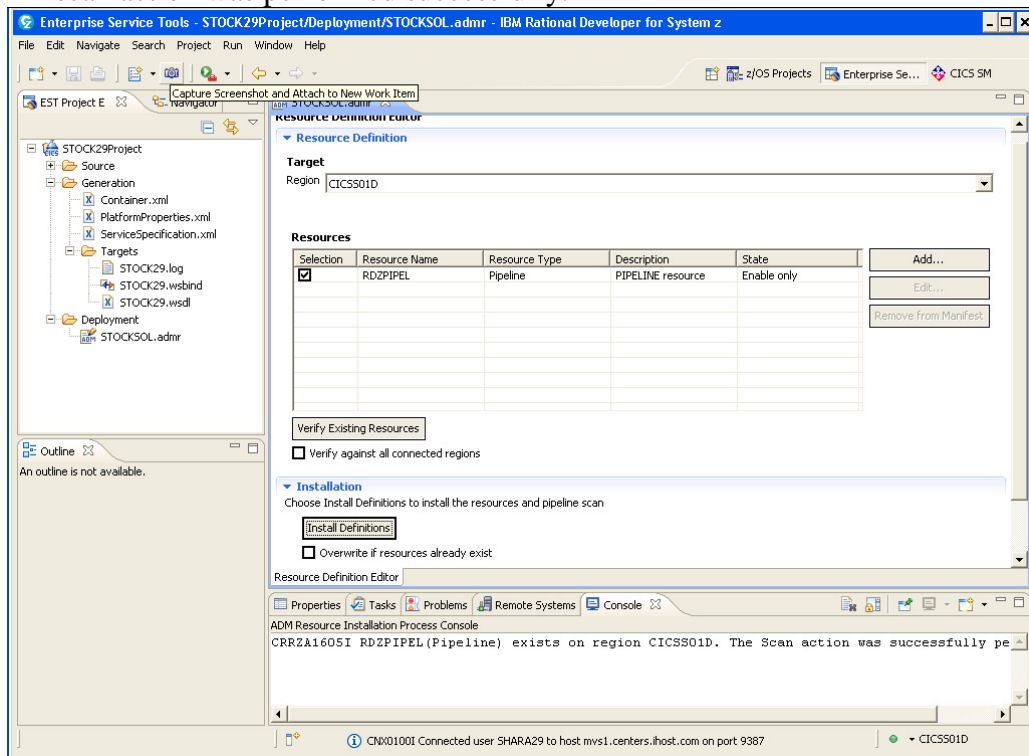
² WSDL stands for Web Services Description Language. In EST Web Services for CICS project, the WSDL file describes the service such as the mapping between the operations and the language structures of the COBOL application.

³ The wsbind file is used at runtime to convert between the SOAP body and the language structures.

- The last page of this wizard is the “**CICS Deployment Options**” page. On this page:
 - Select the checkbox for “Copy the wsbind file to the selected pickup directory”. This will enable the wizard to copy the generated wsbind file to the pickup directory upon successful generation.
 - Select the checkbox for “Generate deployment manifest file”. You will use this manifest file to install this Web service shortly.
 - For the Target CICS region, you will see the region “CICSS01D” and the available pickup directories. Highlight the pickup directory “**shareuser/stevenm/wspickupprovider**” – this is where the generated wsbind file will be copied to. Also note that this pickup directory corresponds to the pipeline called “**RDZPIPEL**”
 - Click **Finish**.
 - The Web services artifacts are generated and you will find them under the **Targets** folder of your project.
-

4. Deploying the Web service on to CICS

- Examine your project. You will notice that the generated artifacts are stored in the **Targets** folder. In addition, you will also see a **Deployment** folder. Expand the folder, locate the manifest file “STOCKSOL.admr”, double-click to open it.
- The manifest file “STOCKSOL.admr” is opened in the Manifest editor. You will use the editor to install the Web service using the information contained in this manifest.
- Review the manifest file. You should see that the RDZPIPEL pipeline resource is listed. To install the Web service you have just generated, you will need to refresh this pipeline so that CICS is aware of the newly copied wsbind file. Scroll down to the **Installation** section of the editor, click **Install Definitions**. This will cause the selected pipeline (RDZPIPEL) to be refreshed.
- On the **Console view** below, you should see a confirmation message that the scan action was performed successfully.



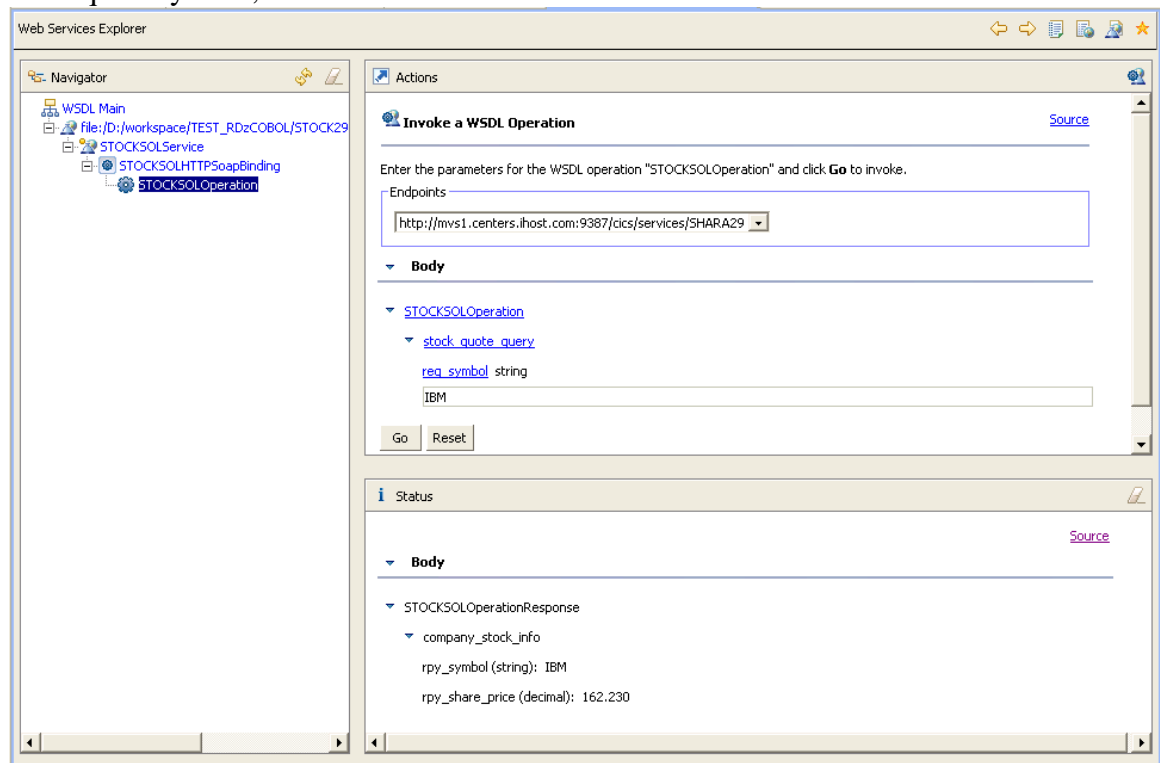
- Click on the “x” on the editor tab to close this edit session.
- Your Web service is installed.

5. Verifying the Web service using CICS Explorer

- Open the **CICS SM** perspective (**Window > Open Perspective > Other...**, then select **CICS SM**)
 - On the **CICSplex Explorer** view, highlight **CICSS01D**.
 - If not already opened, bring up the **Web Services** view (From the menu, select **Operations > Web Services**)
 - You should see the Web service you have just installed, and the pickup directory where it was installed under.
-

6. Testing the Web service using Web Services Explorer

- Return to the **Enterprise Service Tools** perspective. Locate the WSDL file “STOCKnn.WSDL” inside the **Targets** folders of your project. Right-click on the WSDL file, select **Web Services > Test with Web Services Explorer**.
- Double-click on the title bar of the Explorer to maximize the window.
- Once the Web Services Explorer comes up, click on the hyperlink “[STOCKSOLOperation](#)”. This will bring up the screen “Invoke a WSDL Operation”
- In the entry field under [req_symbol](#), enter a stock symbol such as “IBM”, or “INTC”, or “MSFT” (case sensitive, and without quotes). Press **Go**.
- Under the **Status** panel below, you should see the stock quote, along with the request symbol, returned.



---- This is the end of the lab ----

Appendix

About the technology

Eclipse, an open source project, provides a sophisticated, extendable, integrated development environment (IDE) for writing Java applications. With the exception of its team support, which enables groups of users to work with a code repository such as CVS, Eclipse operates as a single user IDE in which all work is performed in a workspace on your local hard drive.

Many commercial products have been built to extend the base Eclipse capabilities. *IBM Rational Developer for System z* provides access to the mainframe and lets you manipulate artifacts that reside on the mainframe with a modern, intuitive, and familiar user interface. It “includes capabilities that can help make traditional mainframe development, Web development, and integrated service-oriented architecture (SOA)-based composite development fast and efficient. COBOL, PL/I, C, C++, High-Level Assembler, and Java™ developer communities can also be more productive when they take advantage of these functions. IBM Rational Developer for System z integrates with and extends the IBM Rational Software Delivery Platform (SDP).⁴

Reference

Title	Location
IBM Rational Developer for System z	http://www-306.ibm.com/software/awdtools/rdz/
RDz articles/infos on developerWorks	http://www.ibm.com/developerworks/rational/products/rdz/
RDz product feature demos	http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp
RDz online forum	http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1131
COBOL cafe	http://www-949.ibm.com/software/rational/cafe/community/cobol

⁴ <http://www-306.ibm.com/software/awdtools/rdz/>