System z Application Developers Hands-on-Labs – COBOL and Web services development using RDz

Student Exercises

Gary Mazo (mazo@us.ibm.com)

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 16*

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.

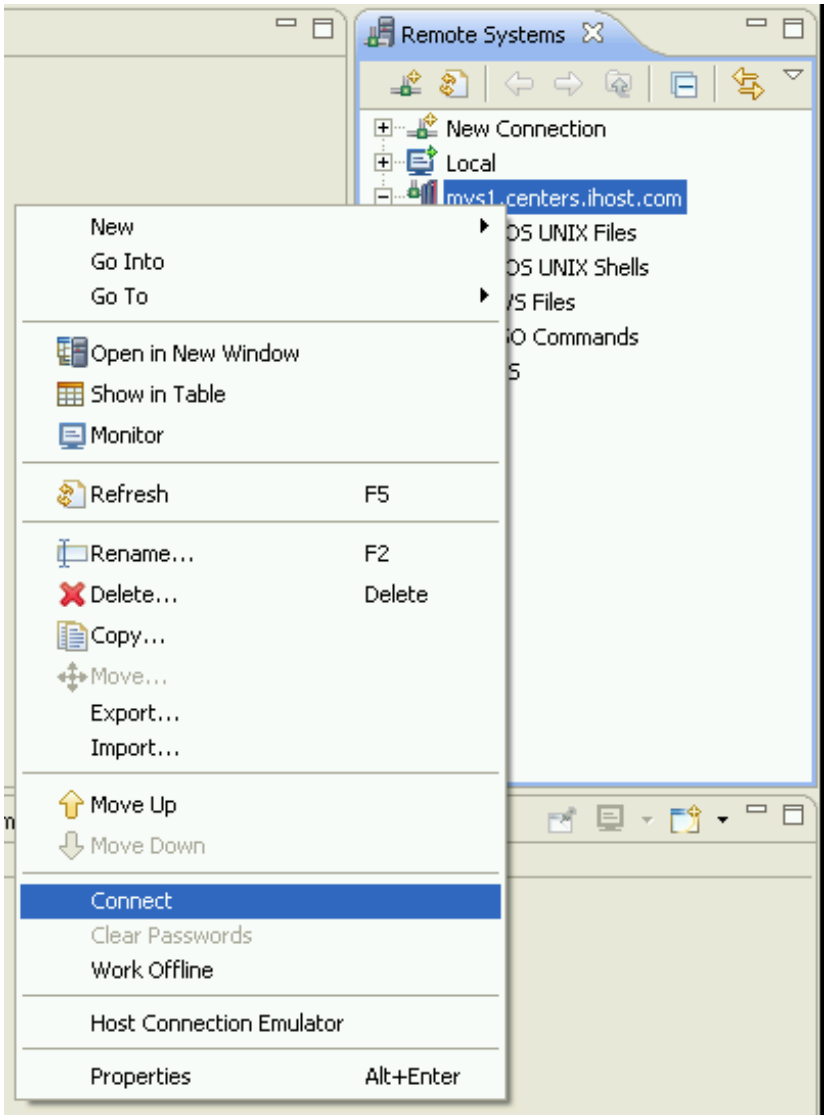
Estimate completion time: 25 minutes

Overview of tasks in this exercise

Connect to the SHARE system
Explore/Browse data sets
Create a filter to view other data sets
Allocate a data set
Copy program into the newly allocated data set
Edit the COBOL program
Verify your changes using the Compare editor
Generate JCL to compile the program, then view the output
(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 "IBM Rational Developer for System z with Java". On the **Workspace Launcher** dialog, select "*c:\Users\csolab\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.

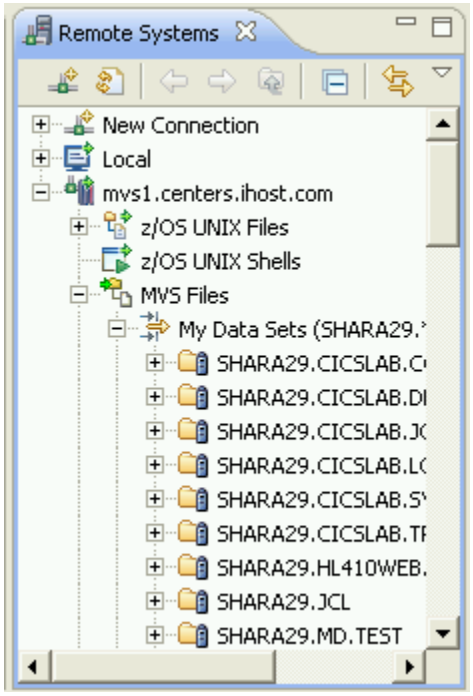
- Enter your assigned userid and password. Note that your userid will be assigned to you at the lab. The userid should look something like *SHARAnn*, where *nn* is your unique id. Your password is "*firstpw*".

- 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.*



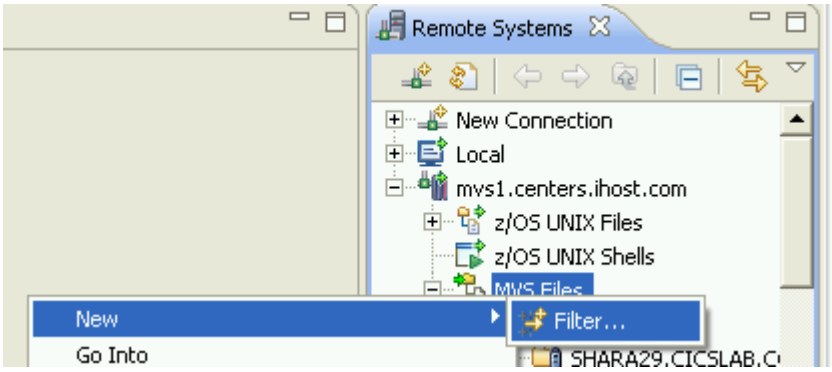
- Expand the various data sets to explore the members and files in the different data sets. (Note: Please do not delete anything unless explicitly instructed to)

---

3. Create filters to view data sets

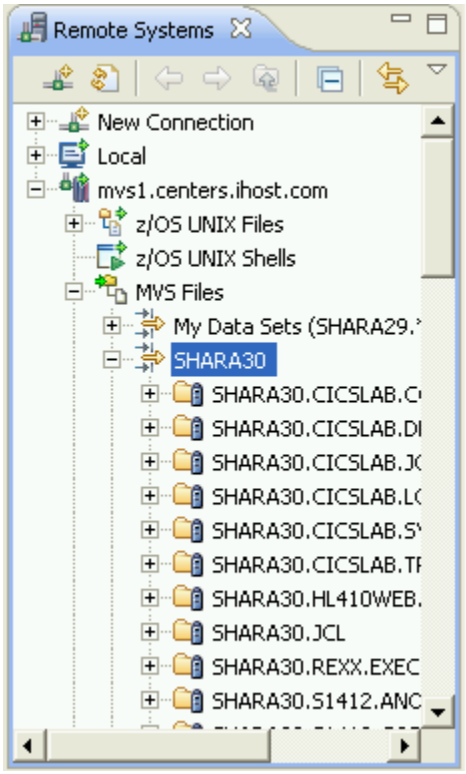
   Create a new filter to view another student's (e.g. SHARA30) data sets

   - 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*)*

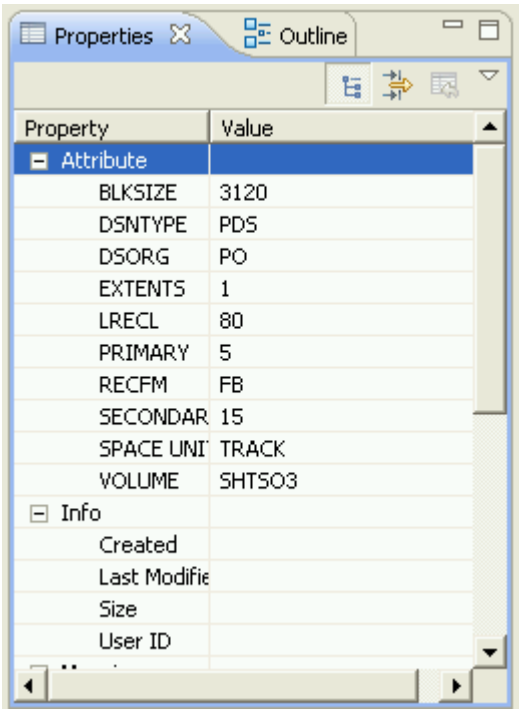


---

4. Allocate a data set

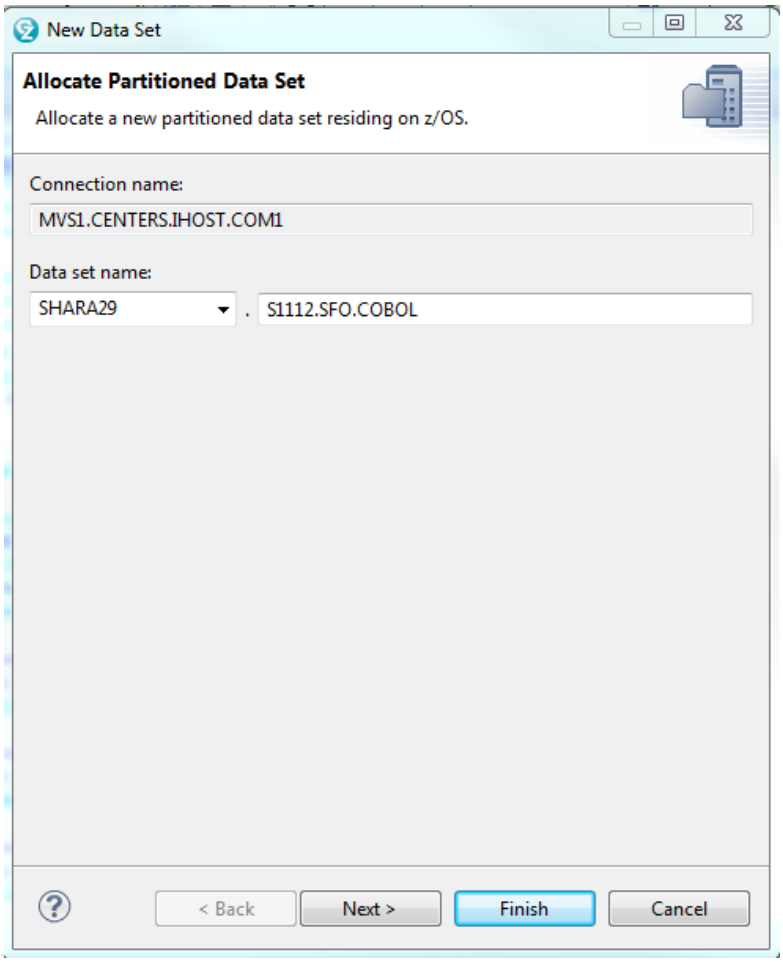
   You can allocate a data set using the characteristics of an existing data set. In this example, allocate a new COBOL data set using the characteristics of SHARAnn.S1112.COBOL. Name the new data set "*SHARAnn.S1112.SFO.COBOL*", where *SHARAnn* is your assigned userid.

- 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:



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

- Enter "*S1112.SFO.COBOL*" in the text field to the right of the drop down.



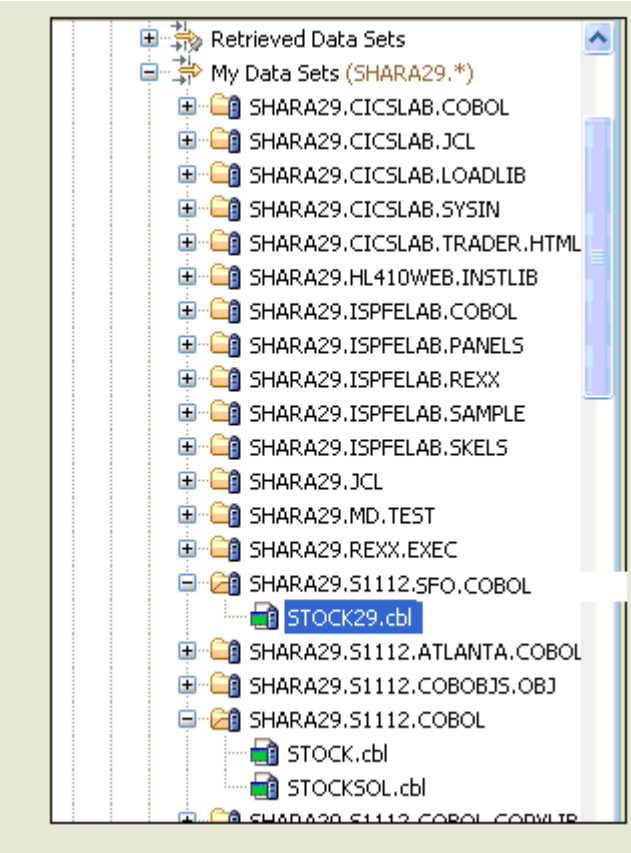
- Click **Next**.
- On the next screen, you should see that the "Copy characteristics from an existing data set" has been selected. Verify that, and click **Next**.
- On the last screen of this wizard, you will see the data set characteristics of the data set to be allocated. These values are identical 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.COBOL, highlight STOCK.cbl. Right-click on it, and select **Copy (or CTRL+C)**
- Navigate to the newly allocated data set: SHARAnn.S1112.SFO.COBOL, 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 STOCK*nn*.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 STOCK*nn*.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:

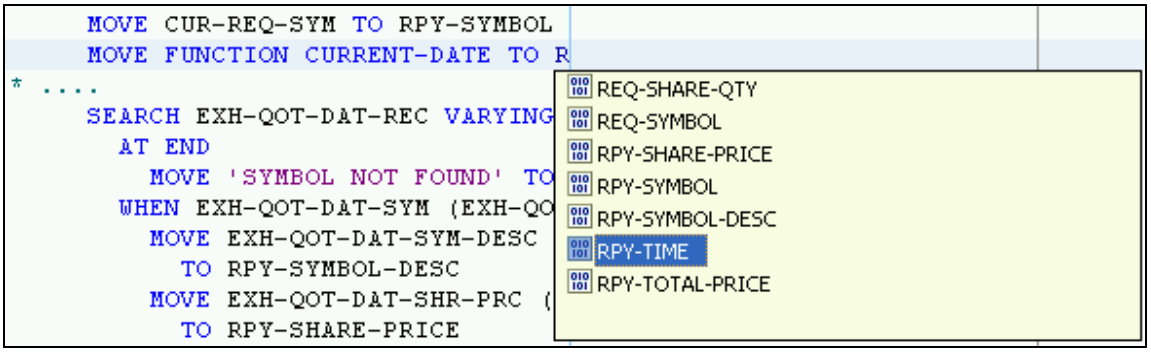
  01 EXH-QOT-DAT-RECS REDEFINES EXH-QOT-DATA**.**

> 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.



  iv. Finally, on the last line of the program, change it to

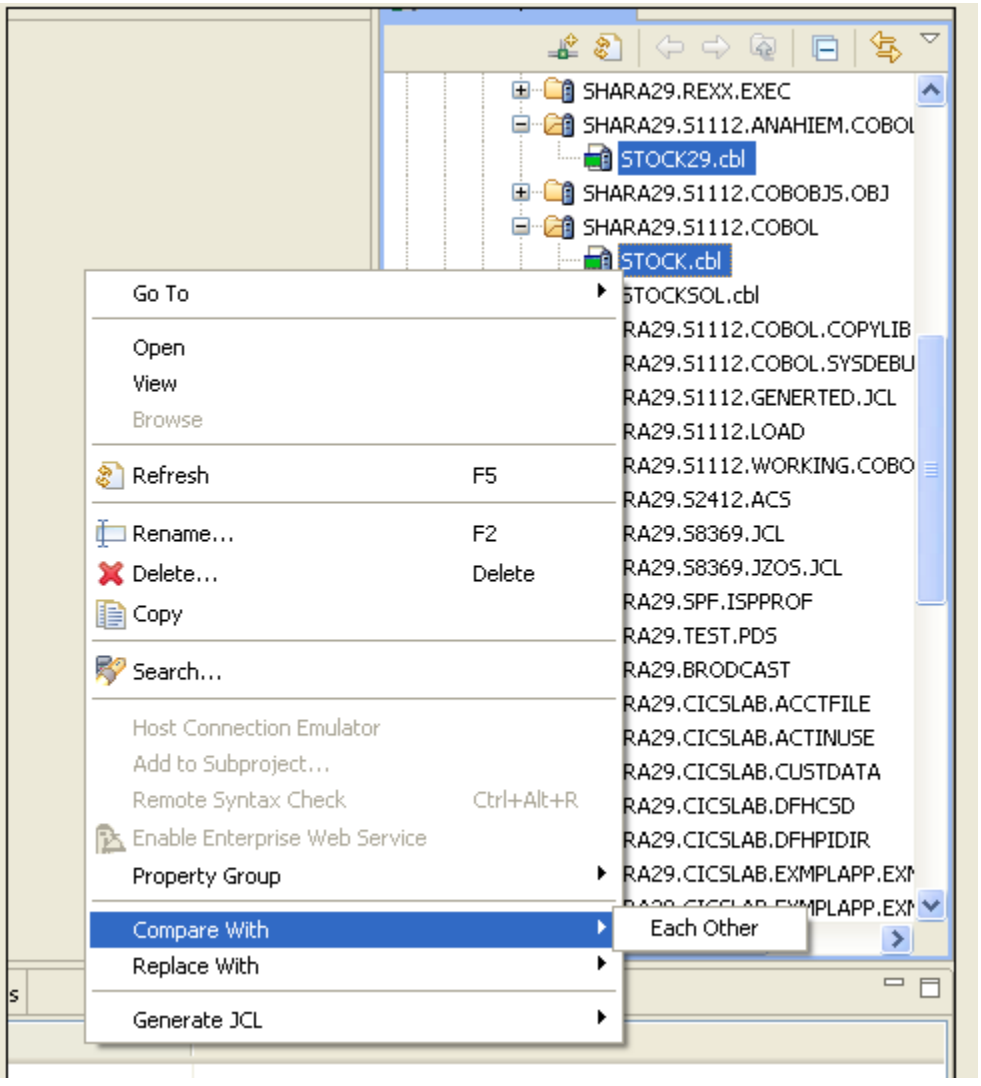
  END PROGRAM 'STOCKnn'. (use your unique id in place of *nn*)

- Save the changes by pressing "CTRL+S"
- Close the program.

---

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

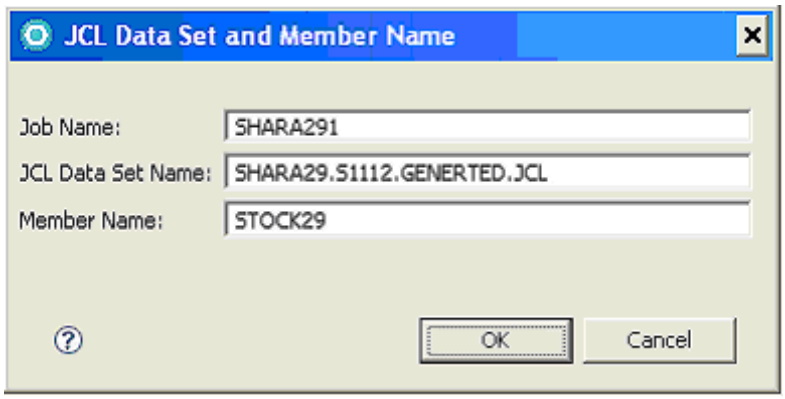
- Go back to the **Remote Systems** view.
- While pressing down the CTRL key, use the mouse to highlight both STOCKnn.cbl and STOCKSOL.cbl (the latter is the "solution" for this exercise, and is found inside the SHAREnn.S1112.COBOL data set).
- With both highlighted, right-mouse click and select **Compare 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

- 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).
- Right-click on the STOCKnn.cbl program, select **Generate JCL > For Compile.** On the pop up, verify that the job name is "SHARAnn*1*", and the JCL Data Set name is: "*SHARAnn.S1112.GENERTED.JCL*". The latter is where the JCL will be generated to. Click **OK**.



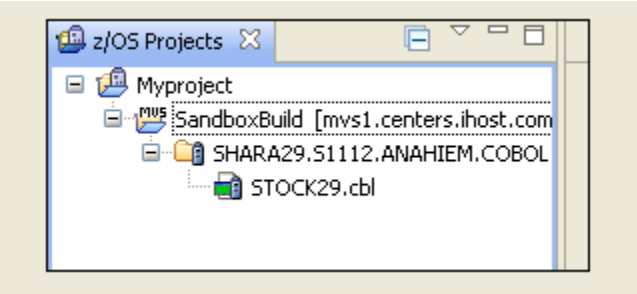
When asked if you want to submit the generated JCL, select **Submit**.

- If the submission is successful, you will see a pop-up dialog, showing you the id for this job. Press **Locate Job**.
- On the **Remote Systems** view, the **Retrieved Jobs** node under **JES** will automatically be selected. Right-click on **Retrieved Jobs** and 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.

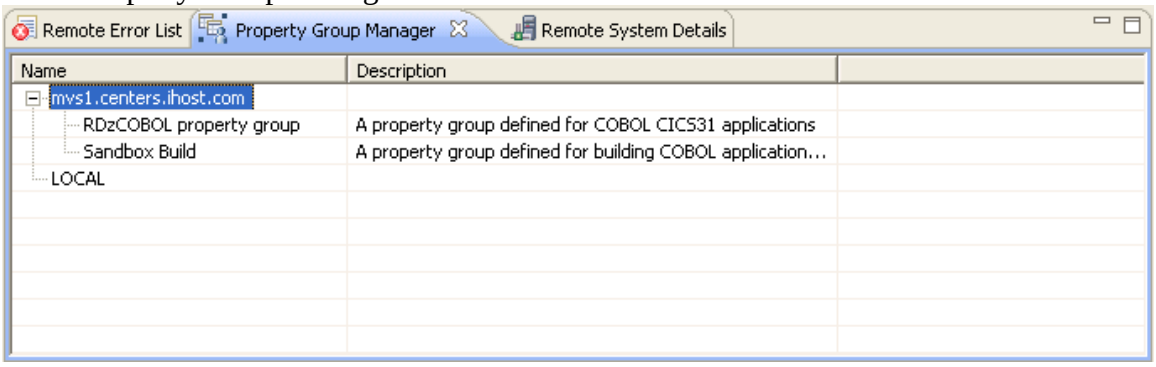
9. (Optional) Building the COBOL program

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

a) 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….**
b) Enter a name for the project, such as "myProject".
c) Under the **Subproject** groupbox below, select to "Create an MVS subproject". Click **Finish**.
d) Enter a name for the subproject, for example, "SandboxBuild".
e) Make sure "Select and add resources to the subproject" is not checked.
f) 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**.
g) You will now see the project on the **z/OS Projects** view.
h) 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)
i) On the pop-up, select your subproject as the copy destination. Your project should look like this:



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

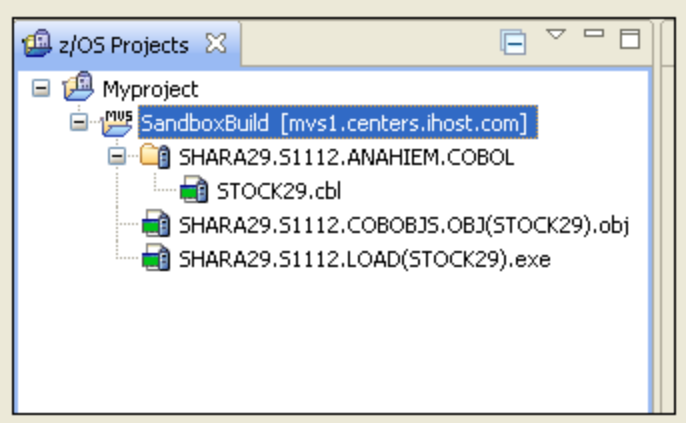


k) Double-click on "Sandbox Build" to open it in the editor. You can view its various attributes such as its name, selected categories, and the properties used in each of the categories such as COBOL options, the different PROCs used, Link options, etc. When you are finished, close the edit session.

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

m) Right-click on the subproject "SandboxBuild", and select "Rebuild Subproject":

n) 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 5.1 runtime, and then test to verify that it works.

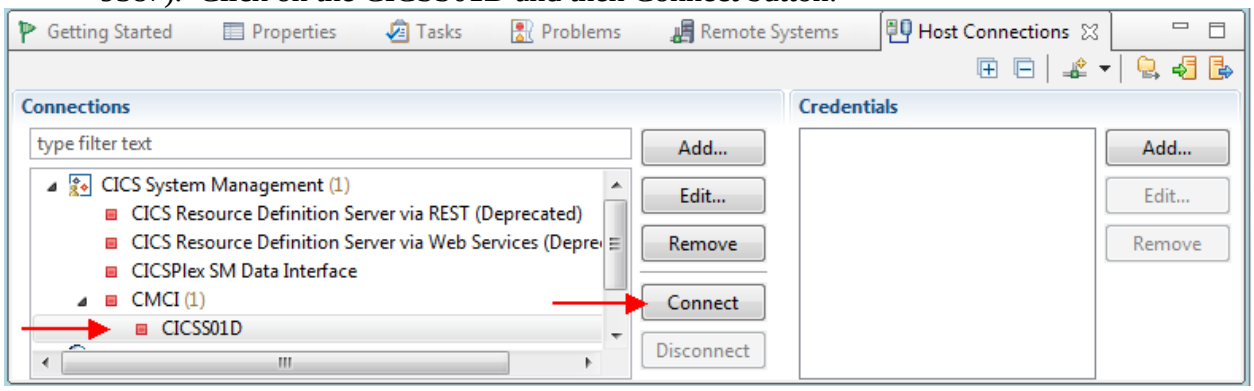
Estimate completion time: 25 minutes

Overview of tasks in this exercise
Open the Enterprise Service Tools
Create a Web service for CICS project
Generate Web service artifacts.
Deploy the Web service to CICS
Verify the Web service installation
Test the Web service

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

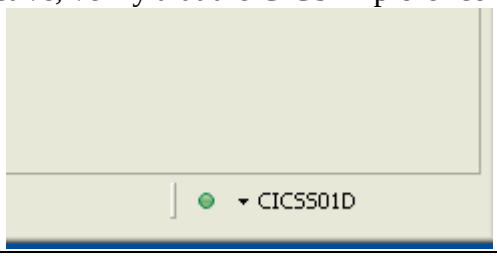
1. Open the **Enterprise Service Tools** perspective

   a) Select **Window > Open Perspective > Others > Enterprise Service Tools**
   b) In the lower portion of the perspective, select **Host Connections** view tab
   c) Currently, there is a connection defined for the region "CICSS01D" (via Port: 9387). Click on the CICSS01D and then Connect button:



   d) In the CMCI Sign on dialog, enter a name for the credentials (for example, SHARA29) and then your User ID (for example, SHARA29) and your password. When finished click **OK**.

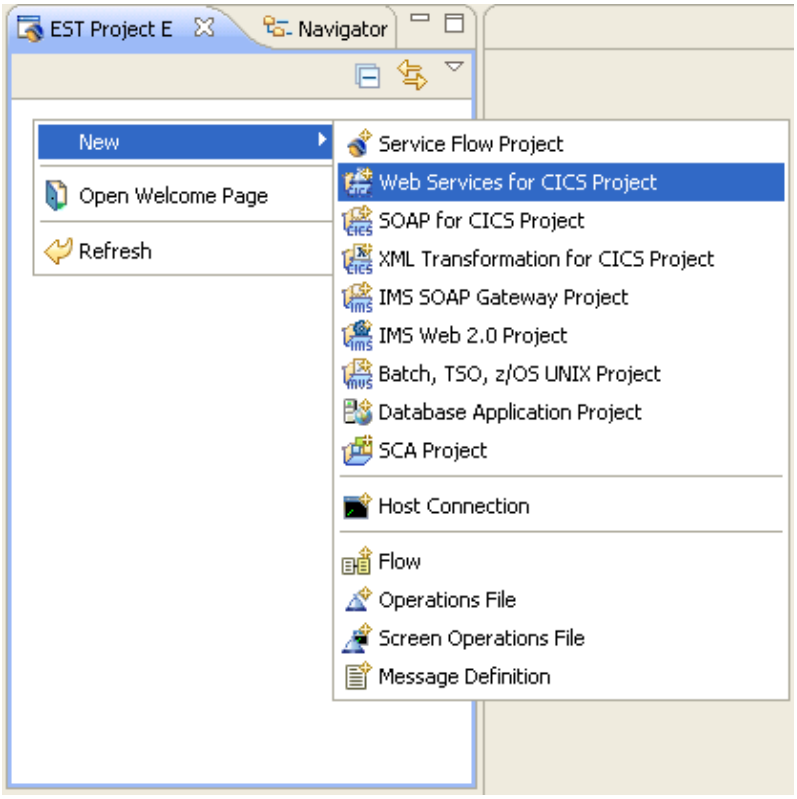
   e) In the Connections pane and in the lower right hand corner of the EST perspective, verify that the CICS Explorer connection is active:

a. 

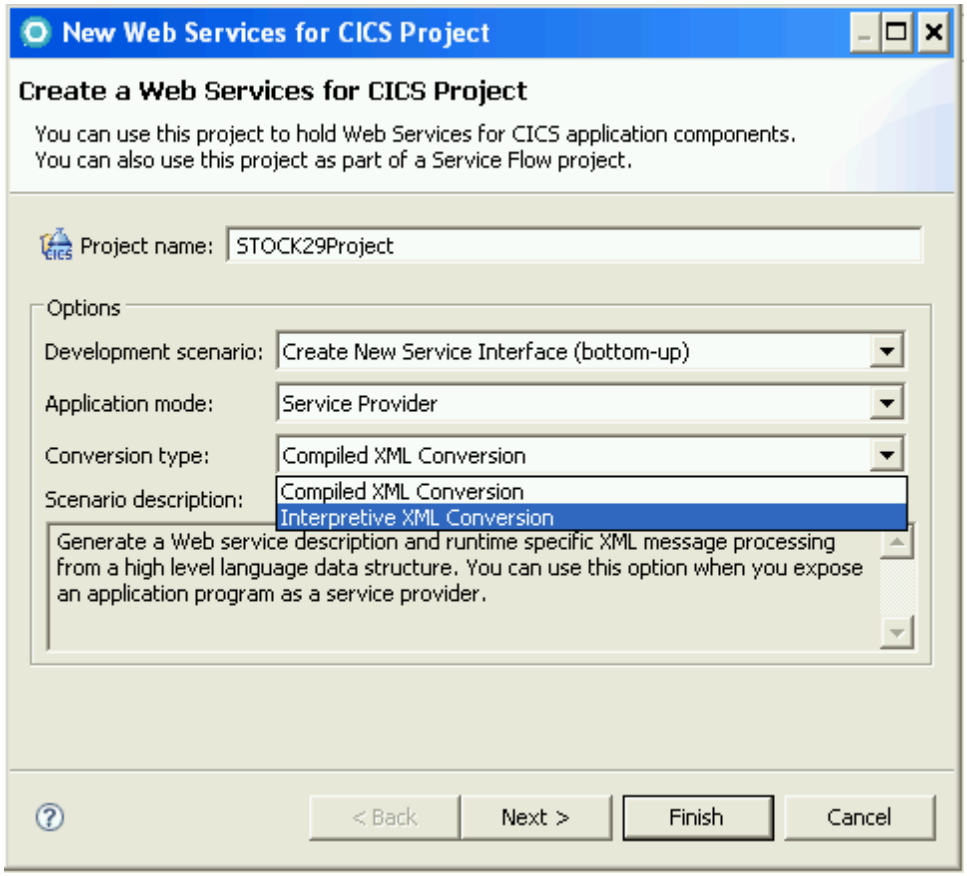
---

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:
    i. Development scenario: *Create New Service Interface (bottom-up)*
    ii. Application mode: *Service Provider*
    iii. 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.

---

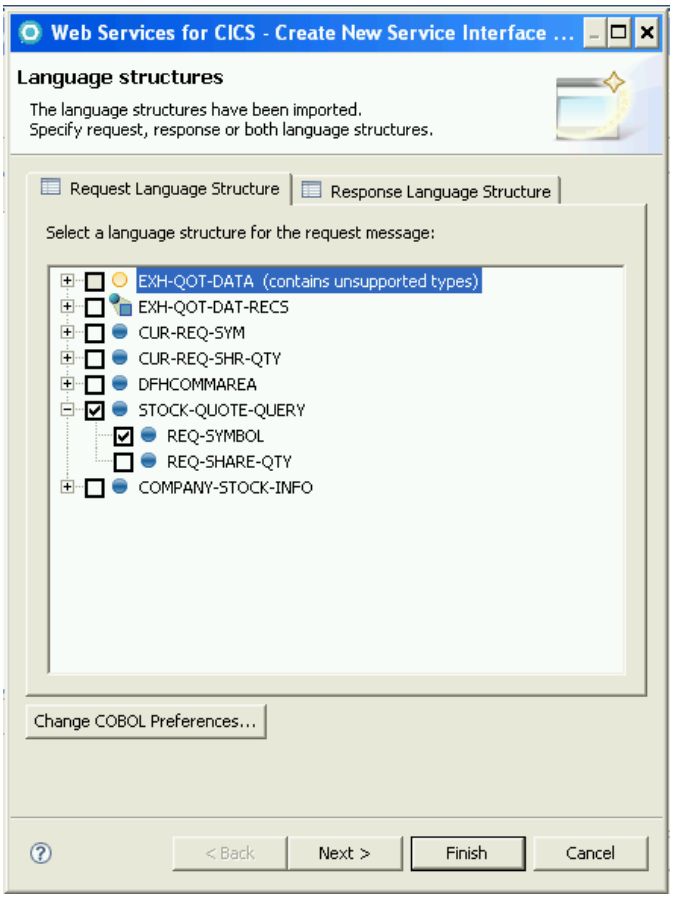
3. Generate Web service

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

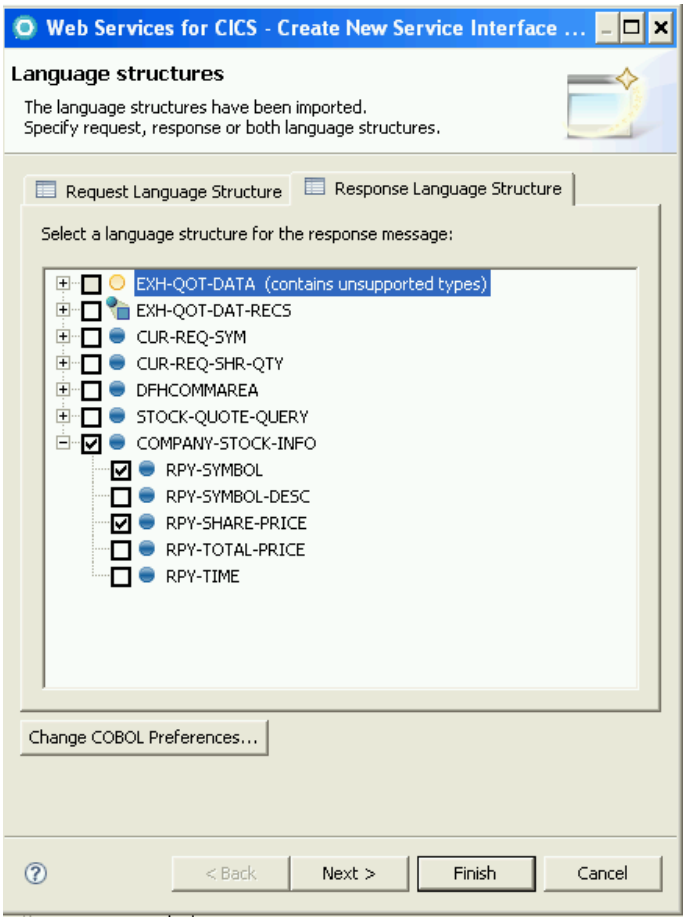
program: STOCKSOL is already built, and is currently running in the CICS region CICSS01D.

a) 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.

i.



b) 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.



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

c) Specify the **Application Properties** and the **Service Properties**.

i. 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

ii. Click on the **Service Properties** tab, and specify the following for **service location**: *http://mvs1.centers.ihost.com:9387/cics/services/SHARAnn/STOCKSOL*. (IMPORTANT: Note ***nn*** is still your user number. Please do not use other numbers as it may interfere with other users and cause unpredictable results.) This location information entered will be written to the WSDL and WSBIND files that will be generated from this wizard.

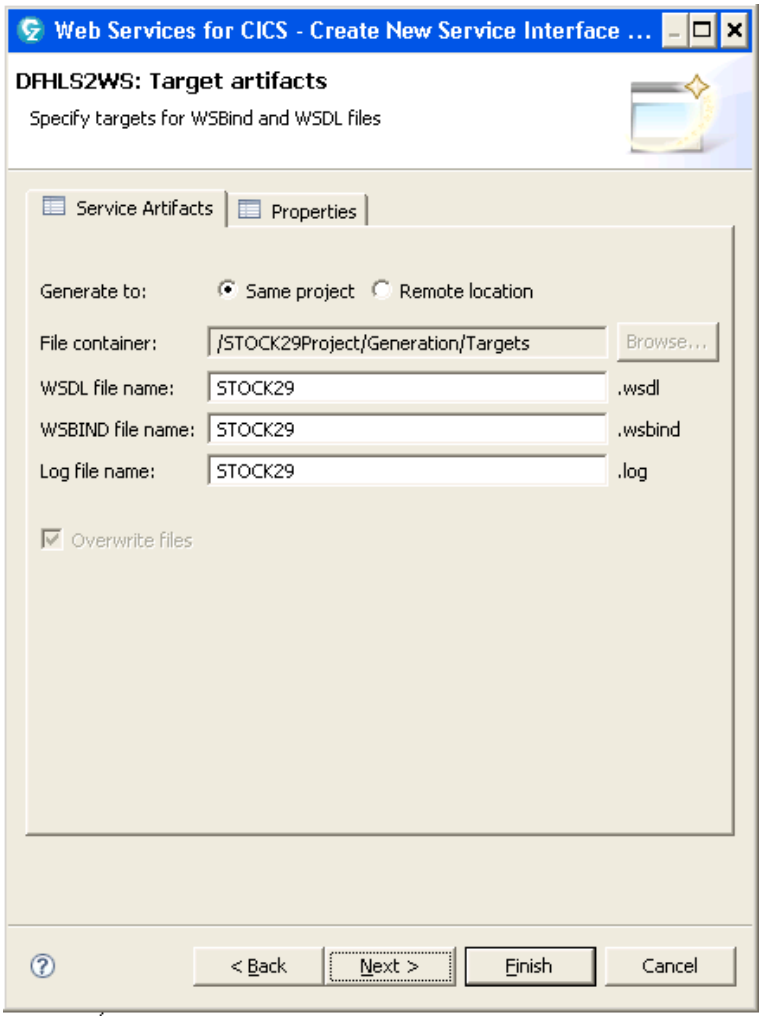
iii. Click **Next**.

d) For the interpretive XML conversion, both a WSDL file[2] and a wsbind file[3] will be generated.

   a. Under the **Service Artifacts** tab, rename the WSBIND file name to STOCK*nn*. This is needed to avoid name conflicts since all the students are deploying their wsbind files to the same pickup directory.

   b. For the WSDL file, you can just accept the default name or change it similar to the dialog below replacing "29" with your user number. You will later use the WSDL file to test your Web service.

i.



   c. 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.

e) The last page of this wizard is the "**CICS Deployment Options**" page. On this page:

   i. 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.

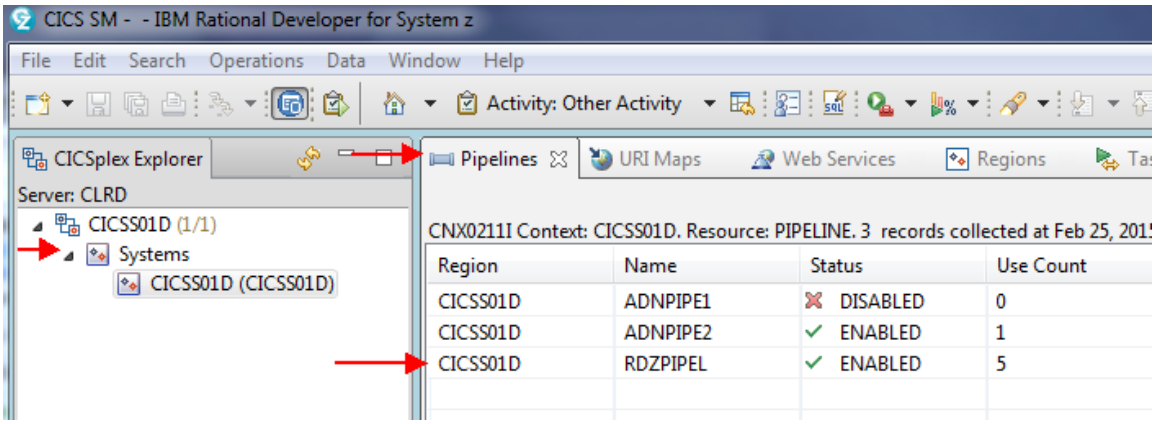
   ii. For the Target CICS region, you will see the region "CICSS01D" and the available pickup directories. Highlight the pickup directory "**shareuser/stevenm/wspickupprovider"** – this is where the generated wsbind file will be copied to. Also note that this pickup directory corresponds to the pipeline called "**RDZPIPEL**"

   iii. Click **Finish**.

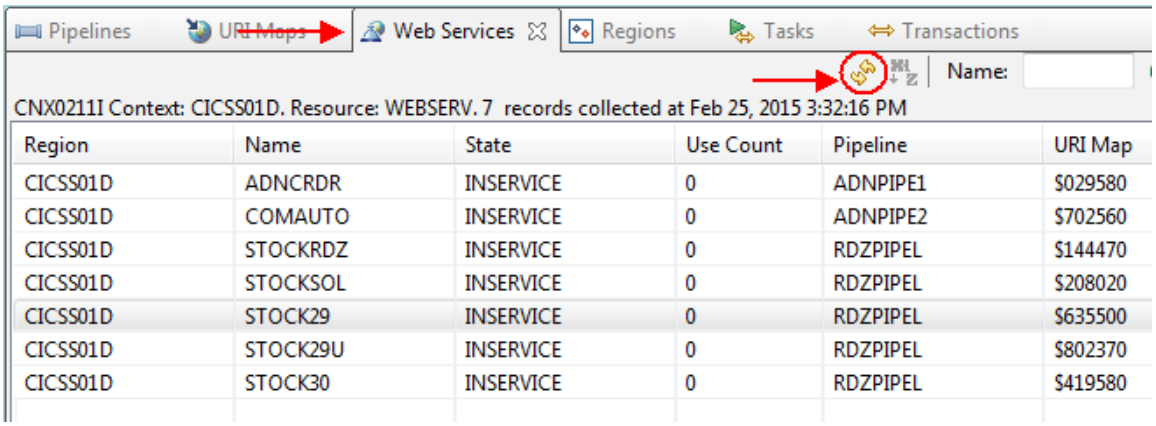
f) The Web services artifacts are generated and you will find them under the **Targets** folder of your project.

---

4. Deploying the Web service into CICS
    - In the tool bar, select **Window,** then **Open Perspective-> CICS SM**
      In this perspective you work with CICS System Management objects.
      In the left pane you see CICSPlex Explorer. If the CICSS01D region in the Explorer shows as not connected, expand the Systems view under the CICSS01D region object and click on the Pipelines tab in the center pane:



- Right mouse click on the **RDZPIPEL** pipeline, and select **Scan**. Click **OK** in the **Perform SCAN operation** dialog.
- Select the **Web Services** tab. Refresh the tab contents by using the refresh button:

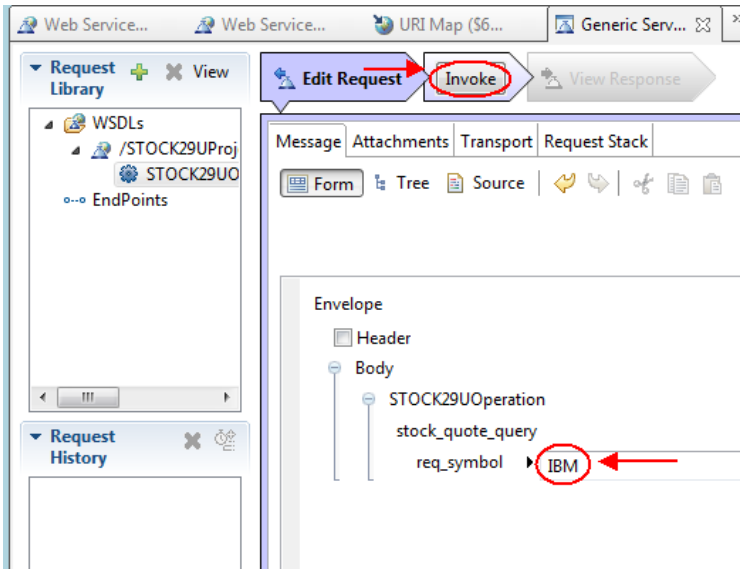


- Observe the name of your service on the list.
- Select the **URI Maps** tab.
- You will observe that the Path column will contain the path to the service you specified in step 3.c) ii above.
- Your Web service is installed.

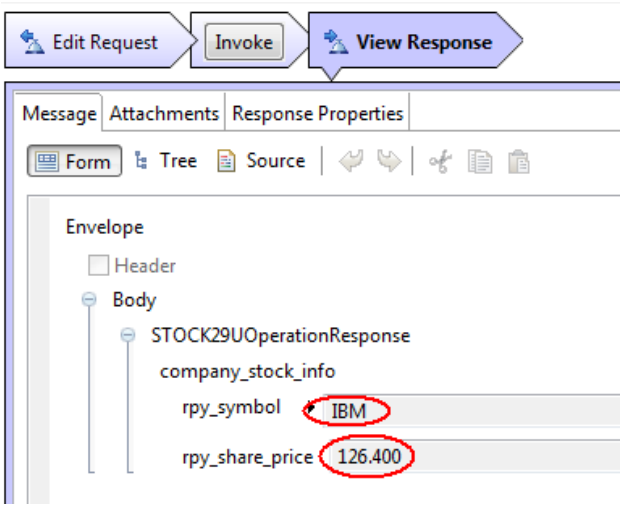
6. Testing the Web service using Generic Service Client

- 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 Generic Service Client**.

- Once the **Generic Service Client** comes up, enter a stock symbol into the **req_symbol** field (such as IBM, MSFT, DIS, or HPQ will do) and click **Invoke**:



- The Generic Service Client should display the response in the View Response form:



---- 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://www-01.ibm.com/support/knowledgecenter/rational_iea/com.ibm.iea.rdz/plugin_coverpage.dita
RDz online forum	http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1131
COBOL cafe	https://www.ibm.com/developerworks/community/blogs/31c890c6-ace1-4eeb-af6b-5950f3a1a5d1/?lang=en

[4] http://www-306.ibm.com/software/awdtools/rdz/

Notices

© Copyright IBM Corporation 1992, 2015. All rights reserved.

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this documentation in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this documentation. The furnishing of this documentation does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for WebSphere Software IBM Corporation
3600 Steeles Ave. East
Markham, Ontario
Canada L3R 9Z7

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this documentation and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Copyright license

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1992, 2007. All rights reserved

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

- IBM
- z/OS
- System z
- Rational

Intel® and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT® and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.