How to Use Assign Action in the Rules Composer

Applies to:

SAP NetWeaver Business Rules Management. For more information, visit the Business Rules Management homepage.

Summary

This is a short tutorial on how to use Assign Action in the Rules Composer. A business use case has been created for the purpose.

Given details such as buyer's name, credit history, spending habit, the discount for the product purchased is set.

This tutorial guides you to create rules and add Assign Action to calculate discount for buyers.

Authors: Arti Gopalan and Rupa Rajagopalan

Company: SAP Labs India

Created on: 28 May 2008

Table of Contents

Prerequisites

Knowledge Required

- You have basic knowledge in rules modeling
- You are familiar with Business Rules Management System

Software Requirements

- You work in the SAP NetWeaver Developer Studio
- Your SAP NetWeaver Developer Studio version includes the Rules Composer perspective
- You should have a running instance of SAP AS, and should have configured the SAP NetWeaver Developer Studio with this instance

Note: In the SAP NetWeaver Developer Studio, choose *Window -> Open Perspective -> Other*. In the dialog box that appears, choose Rules Composer and choose OK.

Procedure

Creating the Rules Composer DC

1. In the SAP NetWeaver Developer Studio, choose *File -> New -> Project*.
2. In the wizard that appears, expand the *Rules Composer* node and choose *Rules Composer Development Component*. Choose *Next*.
3. In the screen that appears, choose the software component where you want to create the DC.

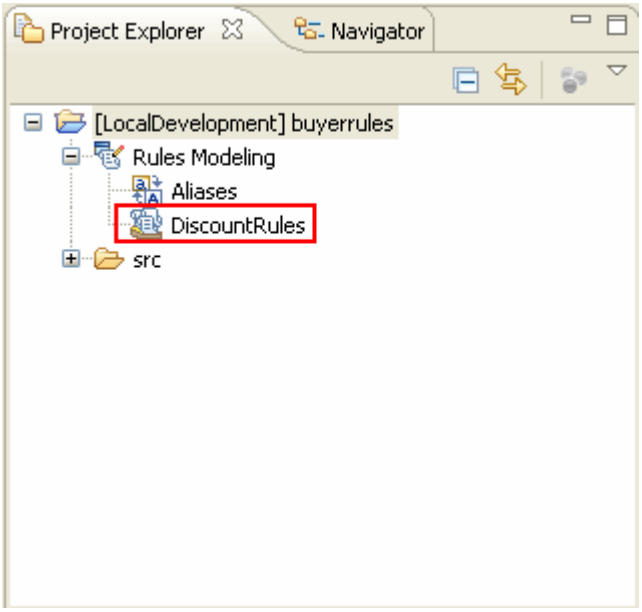
   For example the software component could be *MyComponents [demo.sap.com]* under the Local *Development* node. Choose *Next*.
4. In the screen that appears, enter **buyerrules** in the *Name* field and choose *Finish*.

You should see the Rules Composer DC:*buyerrules* node in the *Project Explorer* view.

Creating the Ruleset

1. In the *Project Explorer* view, expand the Rules Composer DC:*buyyerules* node and in the context menu of the *Rules Modeling* node, choose *New Ruleset*.
2. In the dialog box that appears, enter **DiscountRules** in the field. Choose *OK*.

   You should see the ruleset:*DiscountRules* under the *Rules Modeling* node as shown below:



   You should also see the *DiscountRules* window with the *Overview* tab page open.

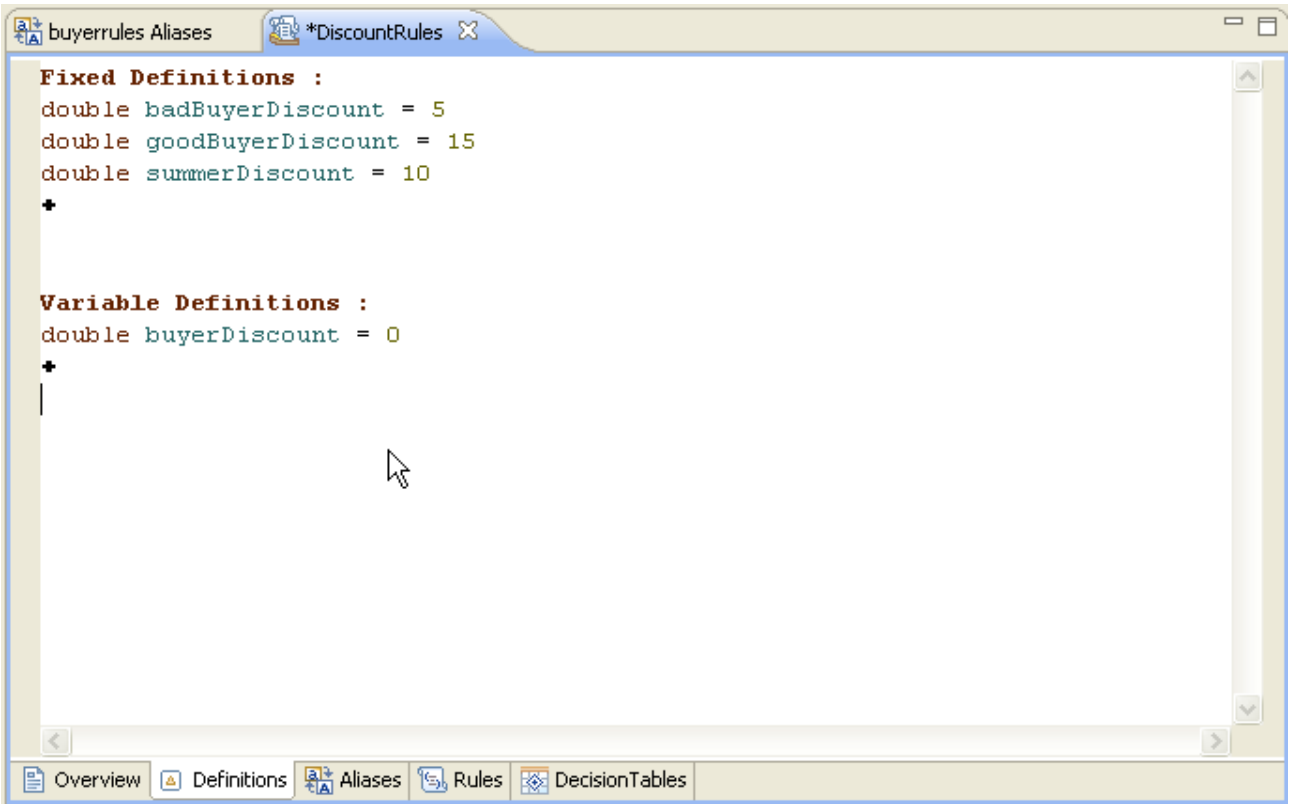
   If the DiscountRules window does not appear, in the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and double-click the *DiscountRules* node.

Creating the Definitions

1. In the *DiscountRules* window, choose the *Definitions* tab.
2. In the Definitions Editor, under the *Fixed Definitions* section, choose the *Add* icon and In the drop down that appears, choose *double*.
3. In the dialog box that appears, enter **badBuyerDiscount** and choose *OK*.

   double badBuyerDiscount = 0 appears.
4. Choose *0* and in the drop down menu that appears enter **5** in the inline text box.
5. Repeat steps 1 to 4 to add two more Fixed Definitions such as *double goodBuyerDiscount =15* and *double summerDiscount = 10.*
6. Repeat steps 1 to 6 to add a Variable Definition such as *double buyerDiscount = 0* under *Variable Definitions* section.

The result must be as follows:



Adding the Classes

Creating the Web Module

1. In the SAP NetWeaver Developer Studio, choose *File - > New -> Project*.
2. In the wizard that appears, expand the *Development Infrastructure* node and choose *Development Component*. Choose *Next*.
3. In the screen that appears, expand the *J2EE* node and choose *Web Module*. Choose *Next*.
4. In the screen that appears, choose the software component where you want to create the DCs. For example expand the *Local Development* node and choose *MyComponents [demo.sap.com]*. Choose *Next*.
5. In the screen that appears, enter *buyer_wm* in the *Name* field. Choose *Finish*.

   The *Java EE* perspective opens and in the *Project Explorer* view, you should see the *buyer_wm* node.

Note: If the Project *Explorer* view does not open, choose *Window -> Show View -> Other* and in the dialog box that appears, expand the *General* node and choose *Project Explorer*. Choose *OK*.

Creating Classes in the Web Module

Make sure you are in the *Java EE* perspective.

1. In the *Project Explorer* view, expand the *buyer_wm* node and in the context menu of *Java Resources: source* node, choose *New Class*.

   Note: If you do not see the *Class* option in the context menu of the *Java Resources: source* node, choose *New -> Other* and in the dialog box that appears, choose *Class*.

2. In the screen that appears, enter *com.sap.buyer* in the *Package* field.
3. Enter **Buyer** in the *Name* field and choose *Finish*.

   You should see the *com.sap.buyer* node under the *src* node.

   The *Buyer.java* window appears.
4. Delete all existing lines and copy the following lines into the window:

```
package com.sap.buyer;
import java.io.Serializable;
public class Buyer implements Serializable{
public Buyer()
	{
	}
	String buyersname;
	String buyersspendinghabit;
	String buyerscredit;
	double setdiscount;

	public String getBuyersname() {
		return buyersname;
	}
	public void setBuyersname(String buyersname) {
		this.buyersname = buyersname;
	}

	public String getBuyersspendinghabit() {
		return buyersspendinghabit;
	}
	public void setBuyersspendinghabit(String buyersspendinghabit) {
		this.buyersspendinghabit = buyersspendinghabit;
	}

	public String getBuyerscredit() {
		return buyerscredit;
	}
	public void setBuyerscredit(String buyerscredit) {
		this.buyerscredit = buyerscredit;
	}

	public double getSetdiscount() {
		return setdiscount;
	}
	public void setSetdiscount(double setdiscount) {
		this.setdiscount = setdiscount;
	}
}
```

5. Press *Ctrl+Shift+F.*
6. Save the changes.

Defining the Public Parts

Make sure you are in the *Development Infrastructure* perspective.

1. In the *Component Browser* view, expand the *MyComponents [demo.sap.com*] node (the software component in which Development Component has been created), and choose the *buyer_wm* node.

Note: If the Component Browser view is not open, choose Window -> Show View -> Other and in the dialog box that appears expand the Development Infrastructure node and choose Component Browser. Choose OK.

2. In the *Component Properties* view, choose the *Public Parts* tab and in the page that appears, choose *Add*.

Note: If the Component Properties view does not open, choose Window -> Show View -> Other and in the dialog box that appears expand the *Development Infrastructure* node and choose *Component Properties*. Choose *OK*.

3. In the screen that appears, enter **public** in the *Name* field. Choose *Finish*.

   You should see the *public* node under *the Defined Public Parts* section.
4. In the context menu of the *public* node, choose *Manage Entities.*
5. In the screen that appears, under the *Entities* section, expand the *Java Class*, *com*, *sap* and *buyer* nodes, select the *Buyer* java class checkbox.
6. Choose *Finish*.

Adding Dependencies to the Rules Composer DC

1. In the *Component Browser* view, expand the *MyComponents [demo.sap.com]* node (the software component in which Development Component has been created) and choose the *buyerrules* node.
2. In the *Component Properties* view, choose the *Dependencies* tab and in the page that appears, choose *Add*.
3. In the dialog box that appears, expand the *MyComponents* node and select the *buyer_wm* checkbox. Choose *Next*.
4. In the screen that appears, under *Dependency Details* section, select the *Design Time*, *Deploy Time* and *Run Time* checkboxes.
5. Choose *Finish*.
6. In the *Dependencies* tab page, expand the *buyer_wm* node and choose the *war* node. Choose *Remove* and in the dialog box that appears choose *Yes*.
7. In the *Component Browser* view, expand the *MyComponents [demo.sap.com]* node and in the context menu of the *buyerrules* node, choose *Build*. Choose *OK*.
8. Check the *Infrastructure Console* for message.

Adding Classes to the Rules Composer DC

Make sure you are in the *Rules Composer* perspective.

1. In the *Project Explorer* view, expand the *buyerrules* node, the *Rules Modeling* node and double-click the *Aliases* node.
2. In the Project Aliases Editor that appears, choose the *Class Aliases* tab and in the tab page that appears, choose the *Add Classes* button.
3. In the dialog box that appears, expand the *com.sap.buyer* node and double-click *Buyer*.
4. Choose *Finish*.
5. In the *Aliases Name* table select all the relevant classes under the *Buyer* node.
6. Save the changes.

Renaming the Class Aliases

In the *Alias Name* table, click each of the aliases. The aliases become editable. Enter an alternative name for the alias.

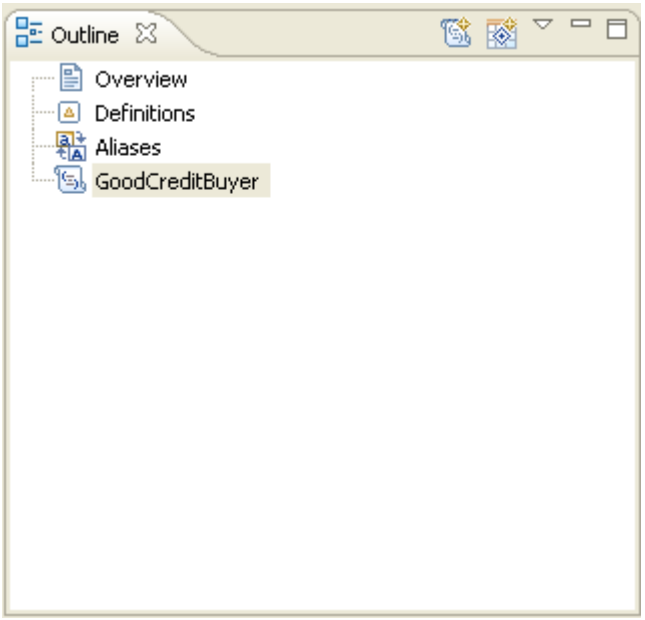
Alias Name	**Rename as**
Buyer.getBuyerscredit	get Buyers credit
Buyer.getBuyersname	get Buyers name
Buyer.getBuyersspendinghabit	get Buyers spending habit
Buyer.getSetdiscount	get Set Discount
Buyer.setSetdiscount	set Set Discount

Creating the Rules

Creating the GoodCreditBuyer rule

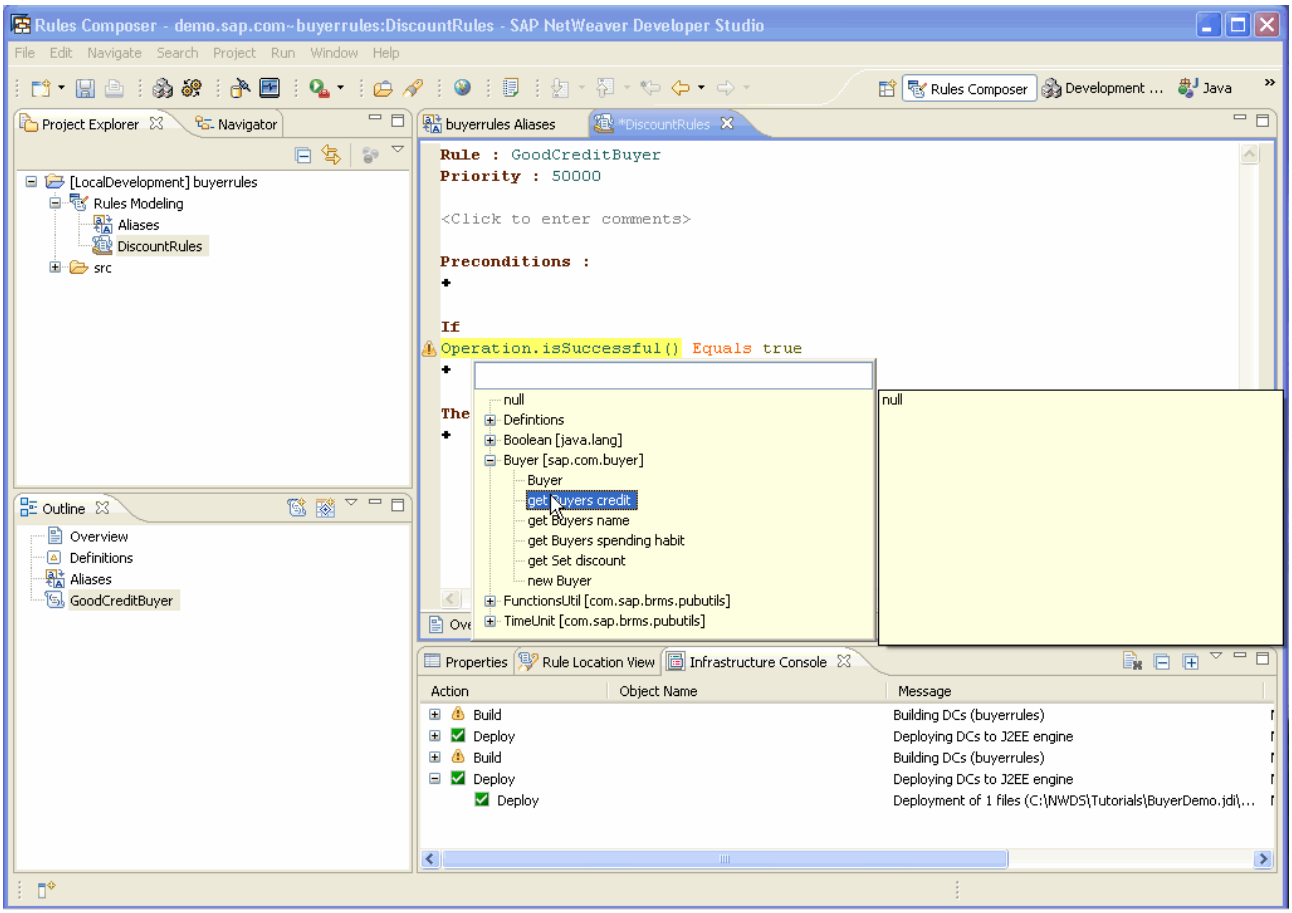
1. In the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and in the context menu of the ruleset:*DiscountRules* node, choose *New Rule.*
2. In the dialog box that appears, enter **GoodCreditBuyer** in the field and choose *OK*.

   In the *Outline* view, the *GoodCreditBuyer* node appears as shown below:



3. In the Rule Editor that appears, under *If* section, choose the *Add* icon.
4. The default condition: *Operation.isSuccessful Equals true* appears.
5. Edit the default condition as follows:

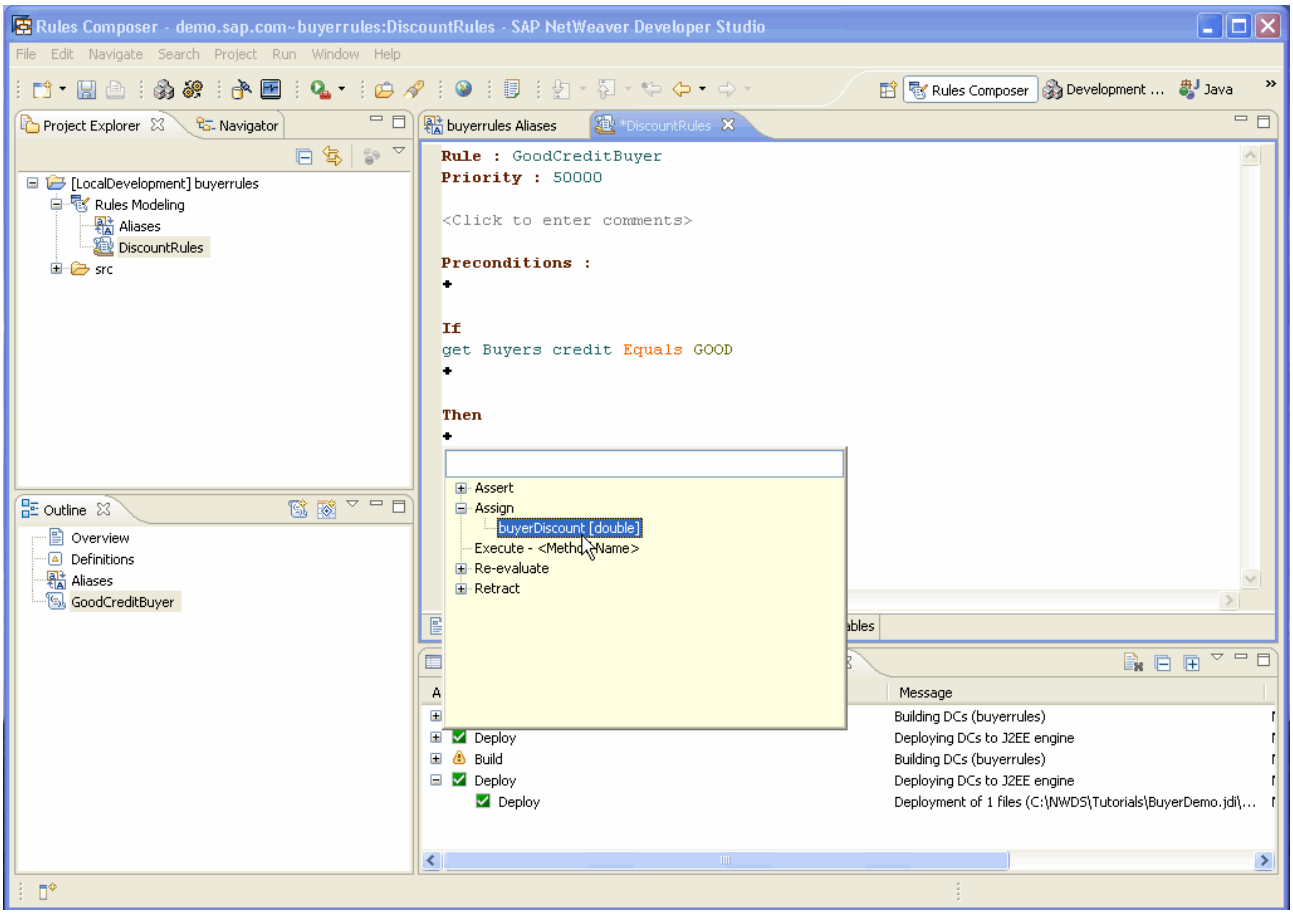
   a. Choose the LValue: *Operation.isSuccessful* and in the drop down menu expand the *Buyer* node and choose *get Buyers credit* as shown below*:*



   b. Leave the comparator:*Equals* as it is.

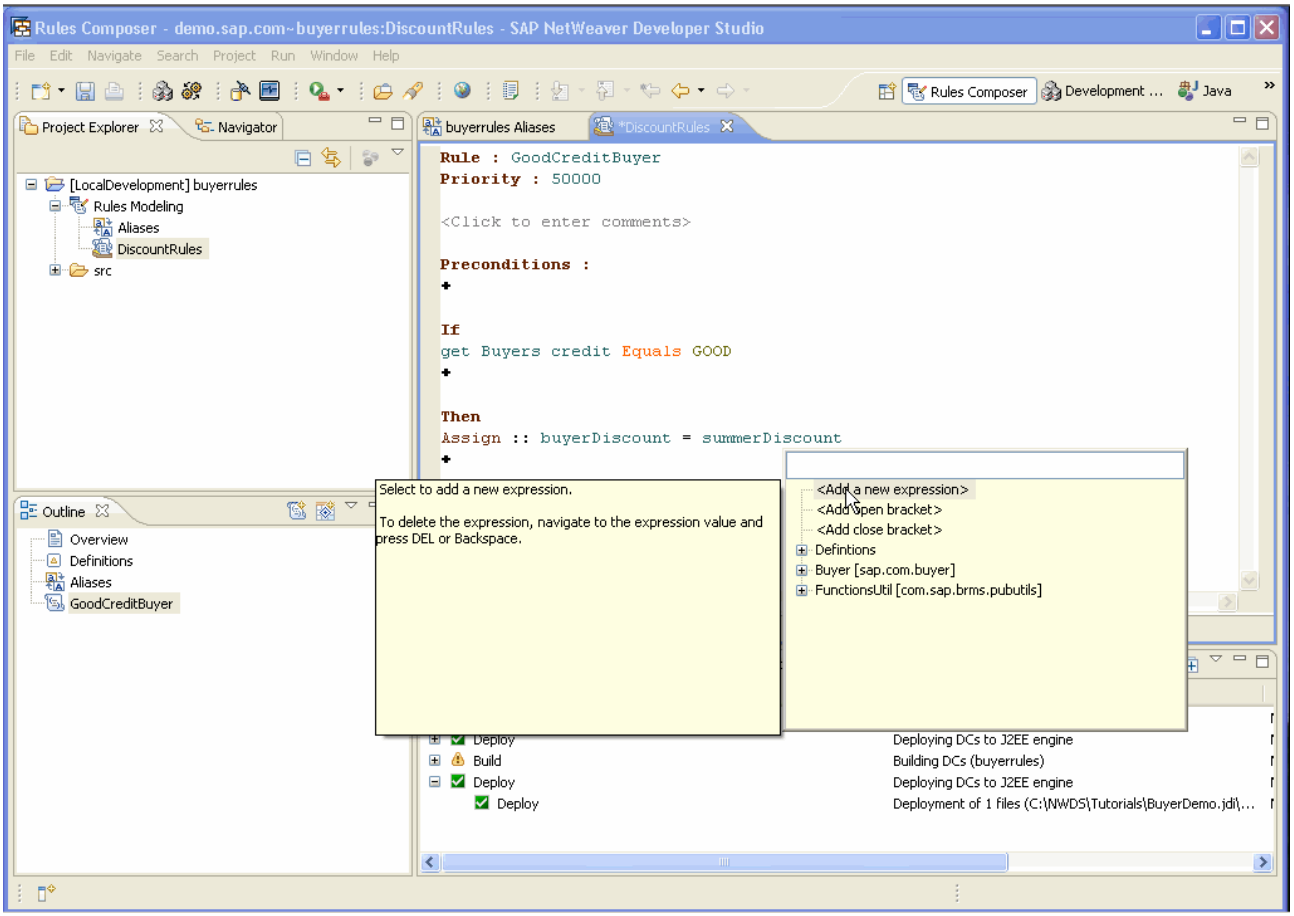
   c. Choose the RValue: *Default Value* and in the inline textbox enter *Good*.

6. Under *Then* section, choose the *Add* icon and in the drop down menu that appears, expand the *Assign* node and choose *buyerDiscount[double]* as shown below:

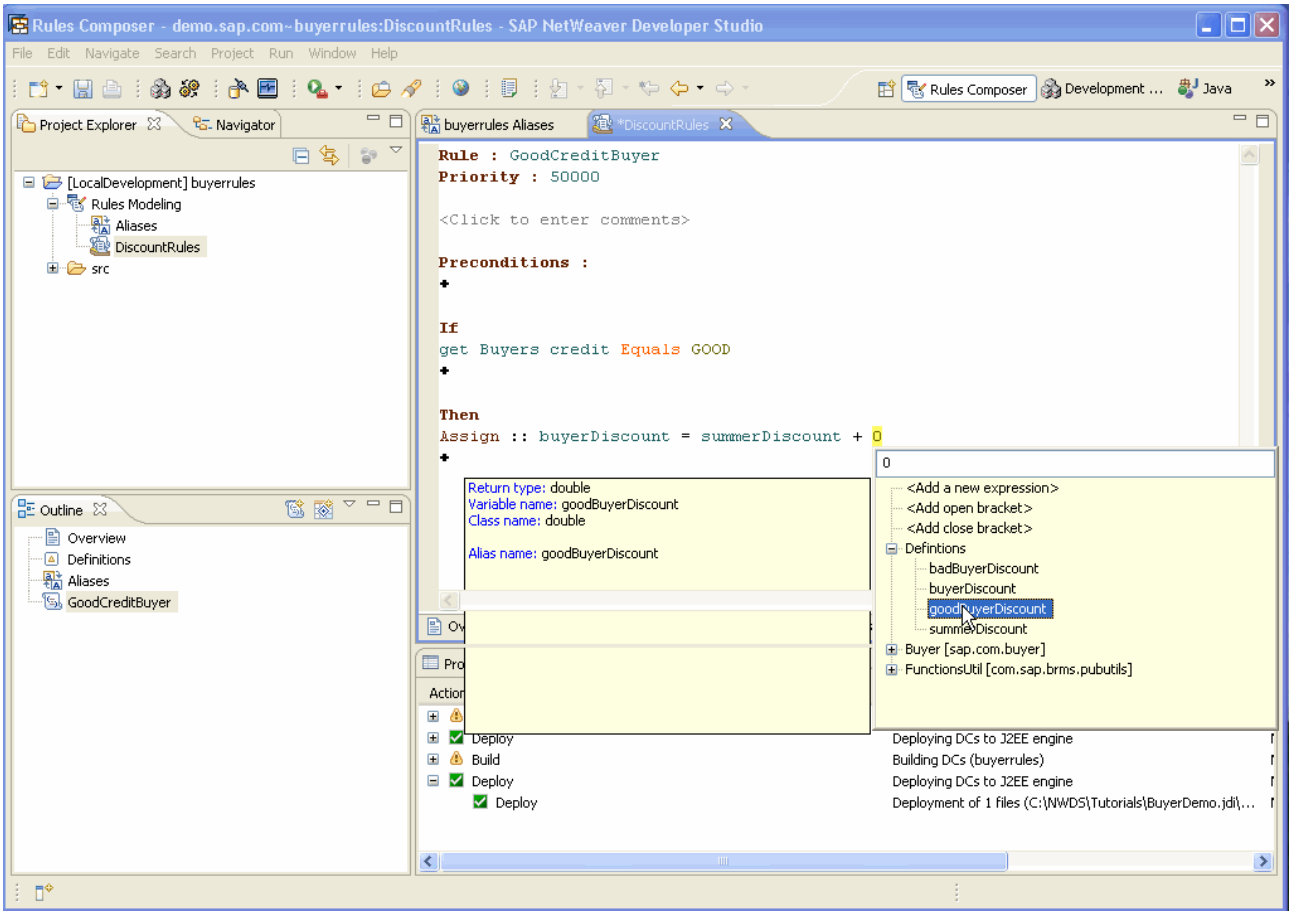


7. Choose the RValue: *0* and in the drop down menu expand the *Definitions* node and choose *summerDiscount*.

8. Choose *summerDiscount* and in the drop down menu choose *Add a new expression* as shown below:

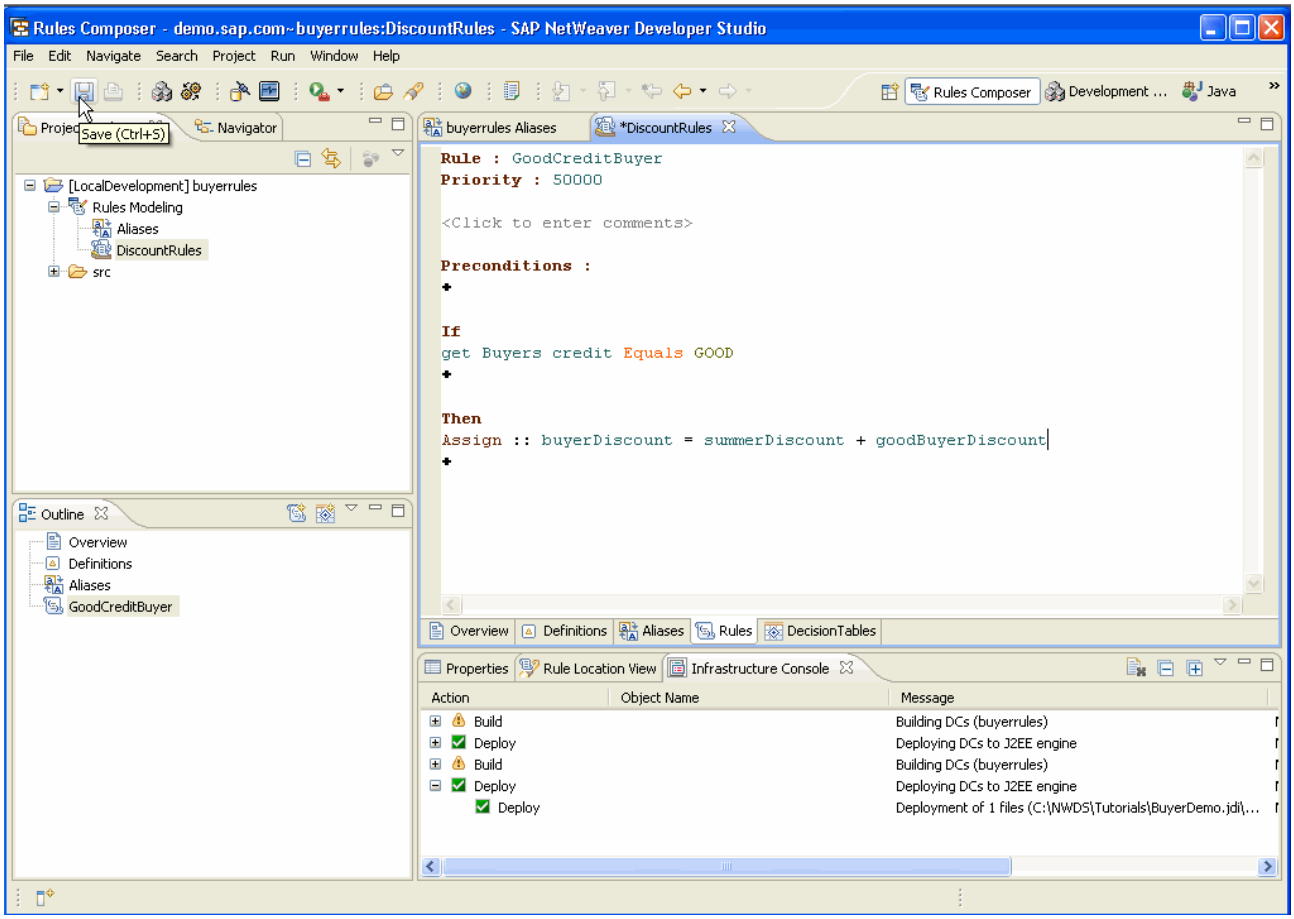


9. Choose the value *0* and in the drop down menu expand the *Definitions* node and choose *goodBuyerDiscount* as show below*:*



10. Save the changes.

The result must be as shown below:



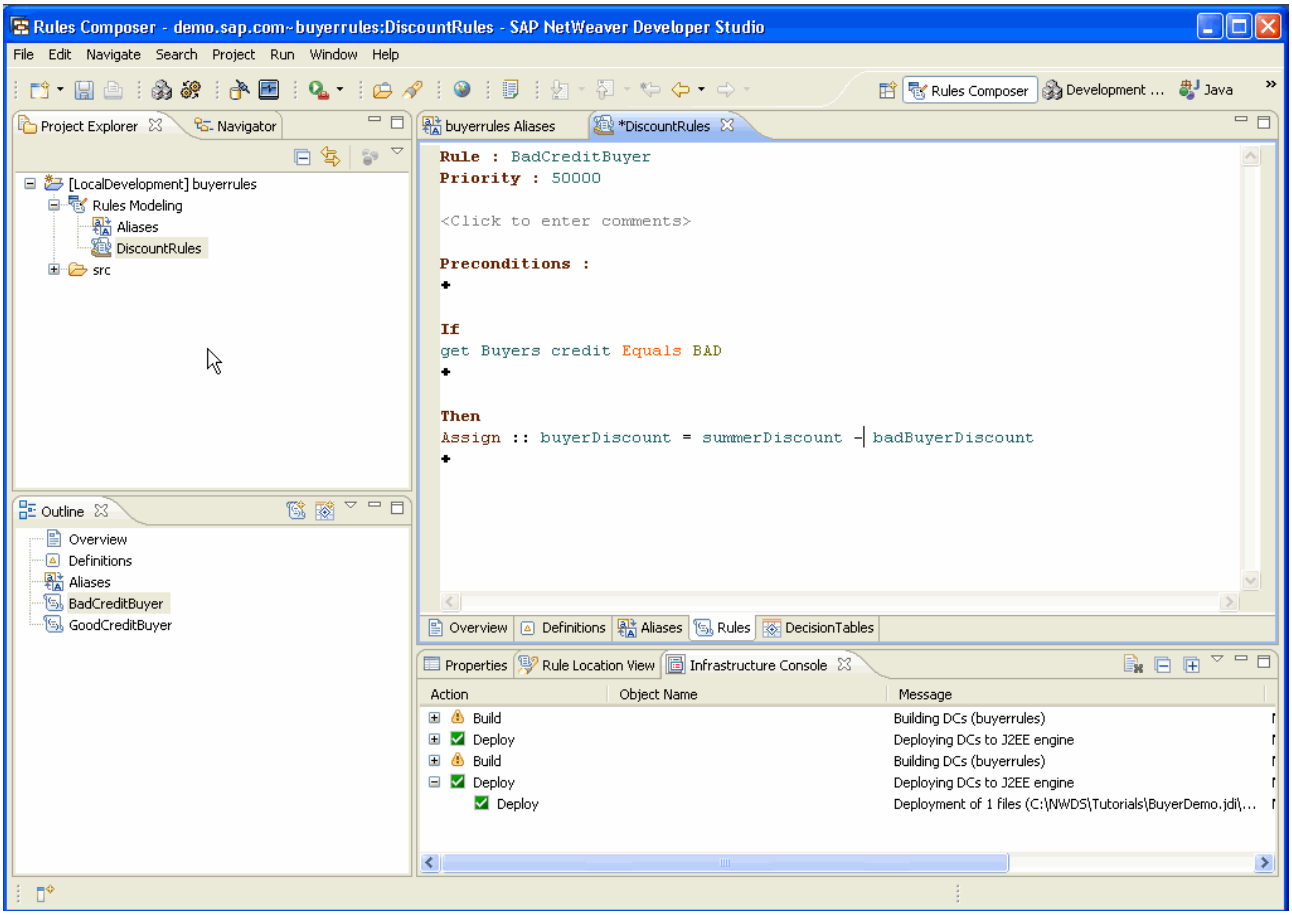
Creating the BadCreditBuyer rule

1. In the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and in the context menu of the ruleset:*DiscountRules* node, choose *New Rule.*
2. In the dialog box that appears, enter **BadCreditBuyer** in the field and choose *OK*.

   In the *Outline* view, the *BadCreditBuyer* node appears.
3. In the Rule Editor that appears, under *If* section, choose the *Add* icon.
4. The default condition: *Operation.isSuccessful Equals true* appears.
5. Edit the default condition as follows:
   a. Choose the LValue: *Operation.isSuccessful* and in the drop down menu expand the *Buyer* node and choose *get Buyers credit.*
   b. Leave the comparator:*Equals* as it is.
   c. Choose the RValue: *Default Value* and in the inline textbox enter *Bad*.
6. Under *Then* section, choose the *Add* icon and in the drop down menu that appears, expand the *Assign* node and choose *buyerDiscount[double].*
7. Choose the RValue: *0* and in the drop down menu expand the *Definitions* node and choose *summerDiscount*.
8. Choose *summerDiscount* and in the drop down menu choose *Add New Expression*.

9. Choose *the value 0 and in the drop down menu* expand the *Definitions* node and choose *badBuyerDiscount.*
10. Choose the mathematical operator (+) and in the drop down menu choose (-).
11. Save the changes.

The result must be as shown below:



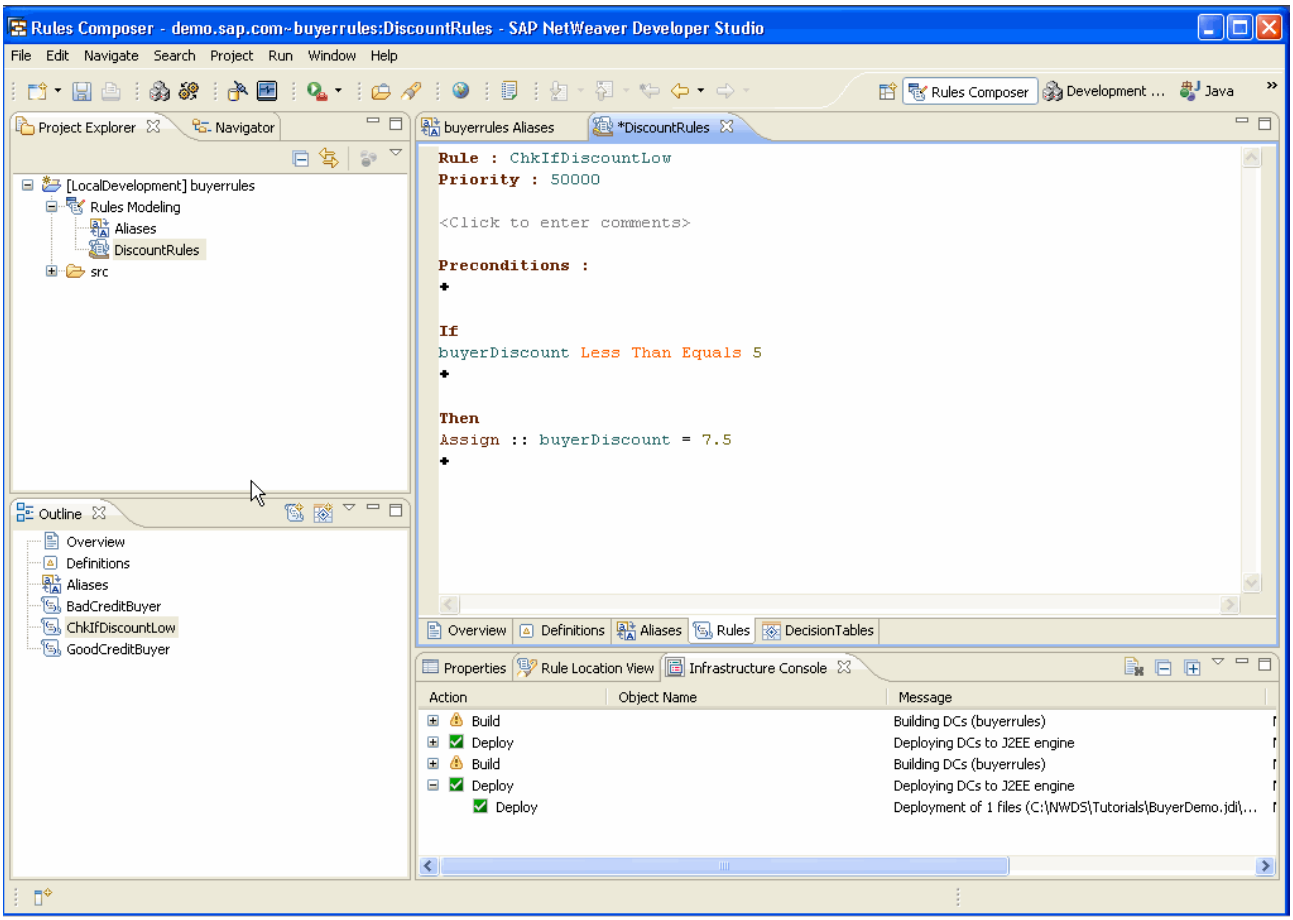
Creating the ChkIfDiscountLow rule

1. In the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and in the context menu of the ruleset*:DiscountRules* node, choose *New Rule.*
2. In the dialog box that appears, enter **ChkIfDiscountLow** in the field and choose *OK*.

   In the *Outline* view, the *ChkIfDiscountLow* node appears.
3. In the Rule Editor that appears, under *If* section, choose the *Add* icon.
4. The default condition: *Operation.isSuccessful Equals true* appears.
5. Edit the default condition as follows:
   a. Choose the LValue: *Operation.isSuccessful* and in the drop down menu, expand the *Defintions* node and choose *buyerDiscount.*
   b. Choose the comparator:*Equals* and in the drop down menu choose *Less Than Equals*.
   c. Choose the RValue: *0* and in the inline textbox enter *5.*
6. Under *Then* section, choose the *Add* icon and in the drop down menu that appears, expand the *Assign* node and choose *buyerDiscount[double].*

7. Choose the RValue: *0* and enter *7.5* in the inline text box.
8. Save the changes.

The result must be as shown below:



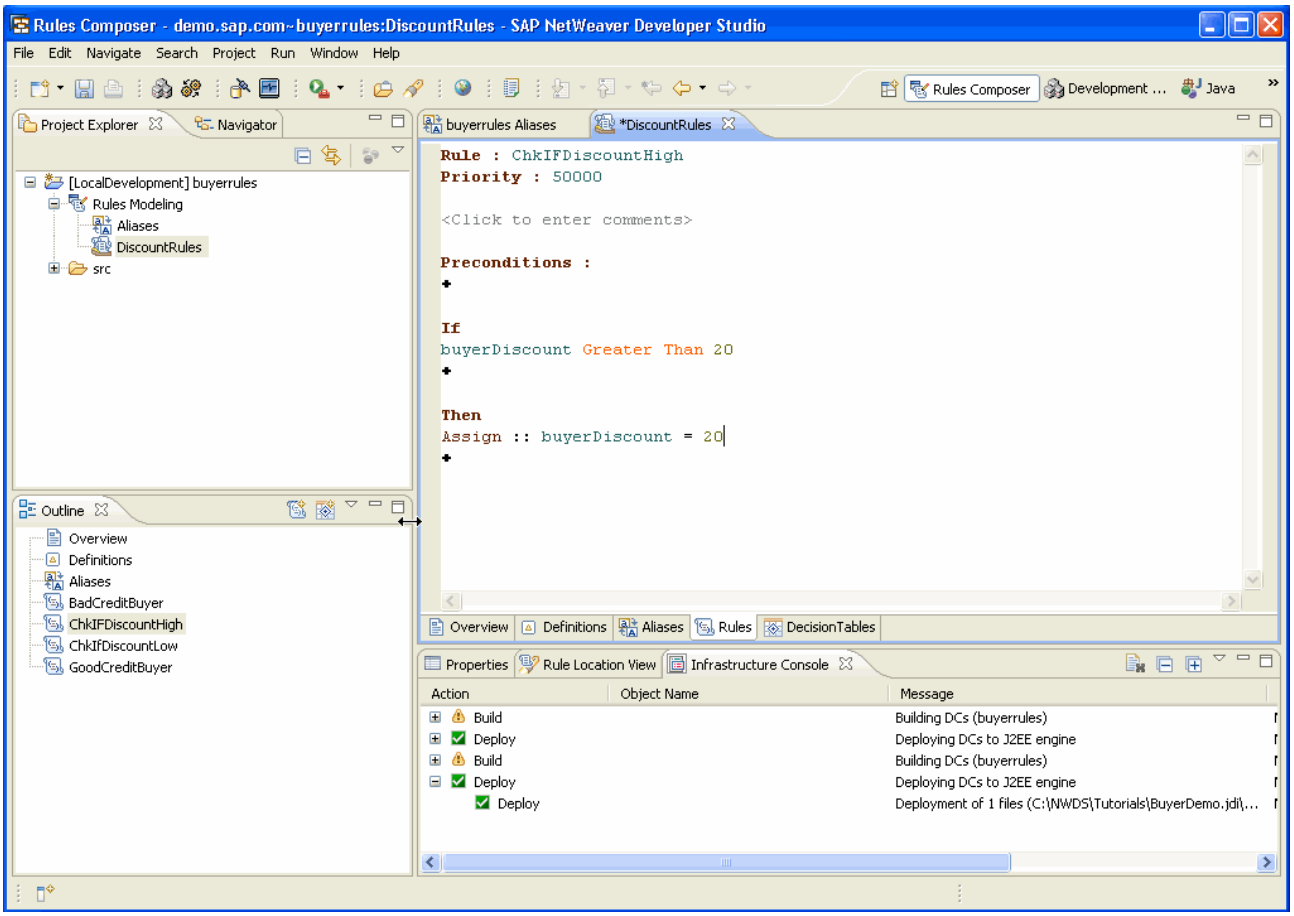
Creating the ChkIfDiscountHigh rule

1. In the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and in the context menu of the ruleset:*DiscountRules* node, choose *New Rule.*
2. In the dialog box that appears, enter **ChkIfDiscountHigh** in the field and choose *OK*.

   In the *Outline* view, the *ChkIfDiscountHigh* node appears.
3. In the Rule Editor that appears, under *If* section, choose the *Add* icon.
4. The default condition: *Operation.isSuccessful Equals true* appears.
5. Edit the default condition as follows:
   a. Choose the LValue: *Operation.isSuccessful* and in the drop down menu, expand the *Defintions* node and choose *buyerDiscount.*
   b. Choose the comparator:*Equals* and in the drop down menu choose *Greater Than.*
   c. Choose the RValue: *0* and in the inline textbox enter *20.*
6. Under *Then* section, choose the *Add* icon and in the drop down menu that appears, expand the *Assign* node and choose *buyerDiscount[double].*
7. Choose the RValue: *0* and enter *20* in the inline text box.

8. Save the changes.

The result must be as follows:



Creating the SetbuyerDiscount rule

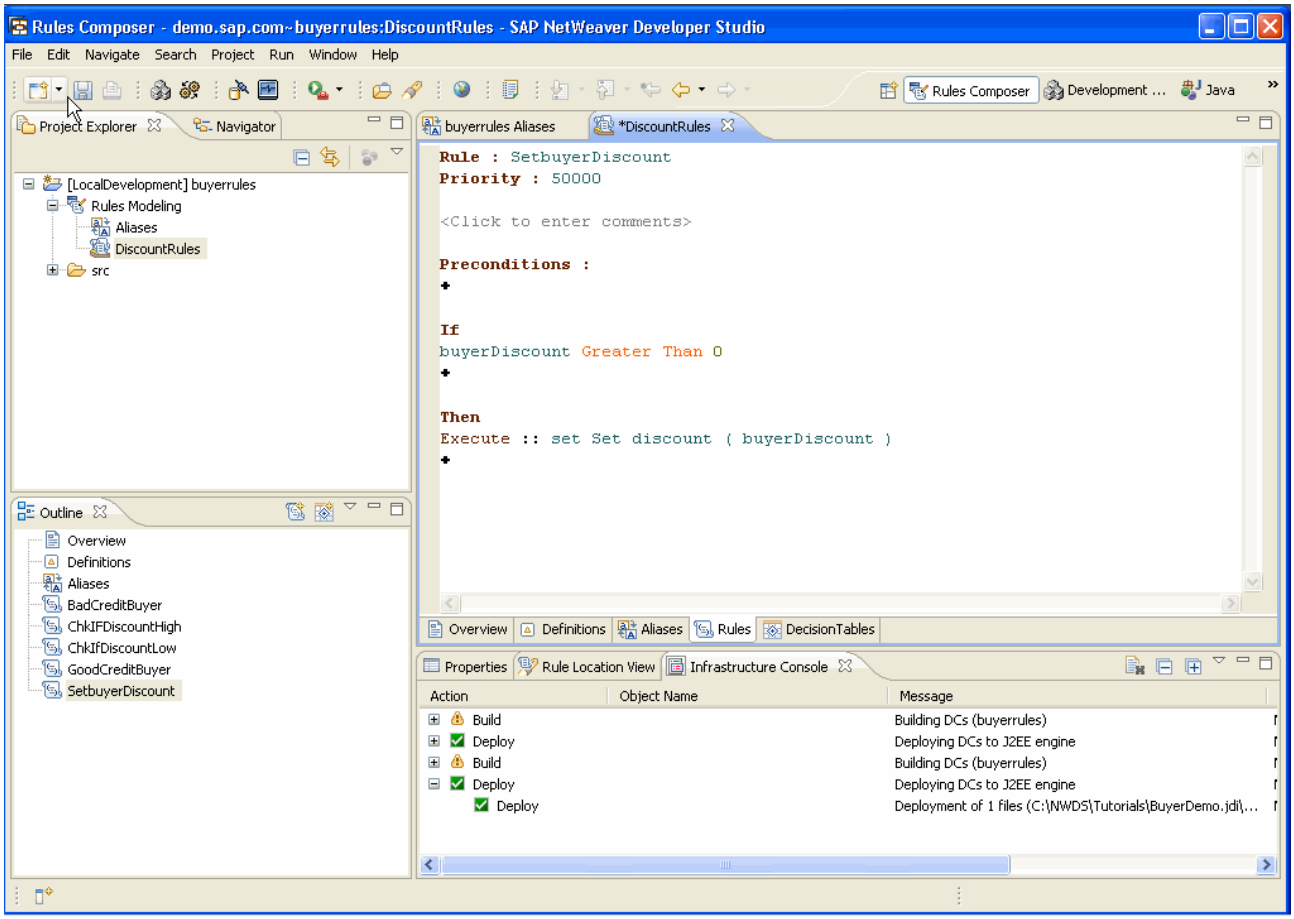
1. In the *Project Explorer* view, expand the Rules Composer DC:*buyerrules* node, the *Rules Modeling* node and in the context menu of the ruleset:*DiscountRules* node, choose *New Rule.*
2. In the dialog box that appears, enter **SetbuyerDiscount** in the field and choose *OK*.

   In the *Outline* view, the *SetbuyerDiscount* node appears.
3. In the Rule Editor that appears, under *If* section, choose the *Add* icon.
4. The default condition: *Operation.isSuccessful Equals true* appears.
5. Edit the default condition as follows:
   a. Choose the LValue: *Operation.isSuccessful* and in the drop down menu, expand the *Defintions* node and choose *buyerDiscount..*
   b. Choose the comparator:*Equals* and in the drop down menu choose *Greater Than.*.
   c. Leave the RValue: *0* as it is.
6. Under *Then* section, choose the *Add* icon and in the drop down menu that appears, choose *Execute <Method>* and choose *buyerDiscount[double].*

   The result must be as shown below:
7. Choose the *<Action Method>* and in the drop down menu expand the Buyer node and choose *set Set Discount ({double}).*

8. *Choose ({double}) and* in the drop down menu expand the Definitions node and choose *buyerDiscount*.
9. Save the changes.

The result must be as shown below:



Deploying the Rules

1. In the *Project Explorer* view, in the context menu of the Rules Composer DC:*buyerrules* node, choose *Development Component -> Build.*
2. In the dialog box that appears, make sure the *buyerrules* checkbox is selected and choose *OK*.

   Choose *Window -> Show View -> Other* and in the dialog box that appears, expand the *Development Infrastructure* node and double-click *Infrastructure Console* to check if the build has happened successfully.
3. In the context menu of the Rules Composer DC:*buyerrules* node, choose *Development Component -> Deploy*.
4. In the dialog box that appears, make sure the *buyerrules* checkbox is selected and choose *OK*.
5. Open the *Infrastructure Console*, to check if the deploy has happened successfully.

Executing the Rules

Creating the Web Module

We have already created a Web Module named *buyer_wm*.

Adding Dependency to the Web Module

1. Choose *Window -> Open Perspective -> Other*.
2. In the dialog box that appears, choose *Development Infrastructure*. Choose *OK*.

Note: If the Component Browser view is not open, choose Window -> Show View –> Other.
In the dialog box that appears, expand the Development Infrastructure node and choose Component Browser. Choose OK.

3. In the *Component Browser* view, expand the *MyComponents[demo.sap.com]* node and choose the *buyer_wm* node.

Note: If the Component Properties view does not open, choose Window -> Show View –> Other.
In the dialog box that appears, expand the Development Infrastructure node and choose Component Properties. Choose OK.

4. In the *Component Properties* view, choose *Dependencies*.
5. Choose the *Add* button and in the wizard that appears, expand the *BRMS-FACADE[sap.com]* node and select the *tc/brms/facade* checkbox. Choose *Next*.
6. In the screen that appears, select the *Design Time*, *Deploy Time*, *Run Time* checkboxes. Choose *Finish*.

Note: Make sure you are in the Java EE perspective.
Unzip the project file and do the following:

1. Expand the web module: buyer_wm node and in the context menu of the Java Resources:source node, choose New ->Other.
2. In the wizard that appears, expand the Java node and choose Package. Choose Next.
3. In the screen that appears, enter **com.sap.helper** in the Name field.
4. Choose Finish.
5. Copy EngineInvoker.java file into com.sap.helper.
6. Expand the Web Content node under the buyer_wm node and copy the following files into it:
7. BuyerDemo.jsp, index.jsp and invoker.jsp

Creating the Enterprise Application

1. In the SAP NetWeaver Developer Studio, choose *File -> New ->* Project.
2. In the wizard that appears, expand the *Development Infrastructure* node and choose *Development Component*. Choose *Next*.
3. In the screen that appears, expand the *J2EE* node and choose *Enterprise Application*. Choose *Next*.
4. In the screen that appears, choose the software component where you want to create the DCs. For example expand the *Local Development* node and choose *MyComponents [demo.sap.com*]. Choose *Next*.
5. In the screen that appears, enter **buyer_ear** in the *Name* field. Choose *Next.*
6. Skip the screen that appears.

7. In the *New EAR Project screen* select the

   LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com checkbox. Choose *Finish*.

   In the *Project Explorer* view, you should see the *buyer_ear* node.

Adding Dependency to the Enterprise Application

Make sure you are in the *Development Infrastructure* perspective.

1. In the *Component Browser* view, expand the *MyComponents[demo.sap.com]* node and choose the *buyer_ear* node.
2. In the *Component Properties* view, choose the *Dependencies tab*.
3. Choose the *Add* button and in the wizard that appears, expand the *BRMS-FACADE[sap.com]* node and select the *tc/brms/facade* checkbox. Choose *Next*.
4. In the screen that appears, select the *Design Time*, *Deploy Time*, *Run Time* checkboxes. Choose *Finish*.

Note: In the context menu of the *buyer_wm* and *buyer_ear* nodes, choose *Sync/Create Project Sync Used DCs and* in the dialog box that appears, choose *OK.*

Creating the application.xml

Make sure that you are in the *Java EE* perspective.

1. Expand the enterprise application: *buyer_ear* node and in the context menu of the *Deployment Descriptor: LocalDevelopment~LocalDevelopment~buyer_ear~demo.sap.com* node, choose *create application.xml*

   You should see the *application.xml* window with the following lines:

```
 <?xml version = "1.0" encoding = "ASCII"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:application="http://java.sun.com/xml/ns/javaee/application_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd" version="5">
<display-name>LocalDevelopment~LocalDevelopment~buyer_ear~demo.sap.com</display-
name>
<module>
<web>
<web-uri>demo.sap.com~buyer_wm.war</web-uri>
<context-root>LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com</context-
root>
</web>
</module>
</application>
```

2. Replace *<context-root>LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com</context-root>* with **<context-root>BuyerRule</context-root>**

Note: Instead of *LocalDevelopment~LocalDevelopment~ buyer_wm ~demo.sap.com*, you need to enter the customized application name i.e in this tutorial, the name of the application is *BuyerRule*.

Building and Deploying

Make sure you are in the *Development Infrastructure* perspective.

1. In the context menu of the *buyerr_wm* and *buyer_ear* nodes, choose *Build* .
2. In the dialog box that appears, choose *OK*.
3. In the context menu of the *buyer_ear* nodes, choose *Deploy*.
4. In the dialog box that appears, choose *OK*.
5. Open the *Infrastructure Console*, to check if the build and deploy actions have happened successfully.

Note: You can also build and deploy in the *Java EE* perspective.

1. In the *Project Explorer* view, in the context menu of the *buyerr_wm* and *buyer_ear* nodes, choose *Development Component > Build*
2. In the dialog box that appears, choose *OK*.
3. In the context menu of the *buyer_ear* node, choose *Development Component > Build*.
4. In the dialog box that appears, choose *OK*.

Running the Web Module

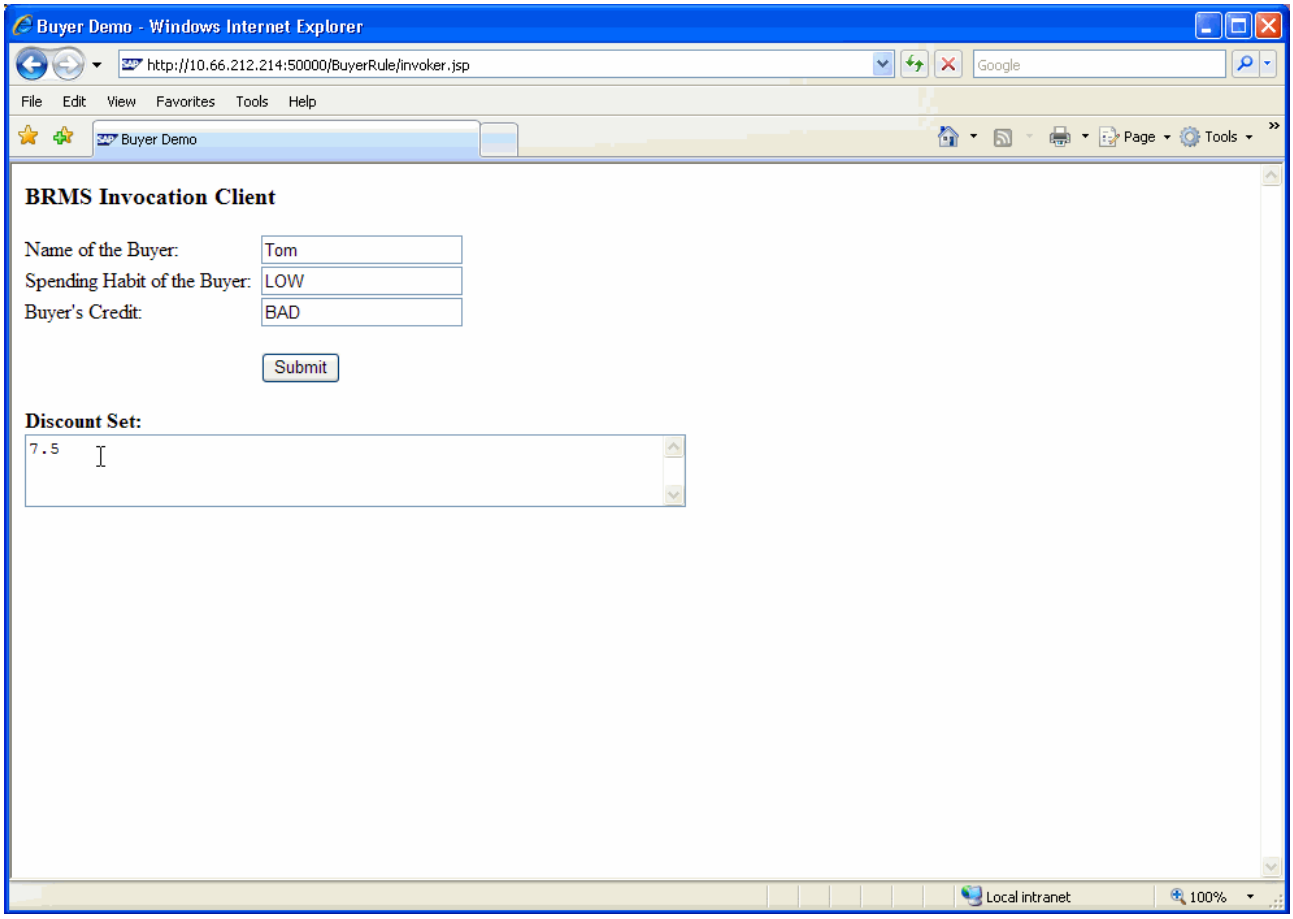
1. Open the browser and enter the Application Server address followed by the port name and the Web Module name: *BuyerRule*.
2. Enter the following data in the respective fields:

Field	User Entry
Name of the Buyer	Tom
Spending Habit of the Buyer	LOW
Buyer's Credit	BAD

3. Choose *Submit*.

   You should get the Discount as 7.5.

Here is a snapshot of the web module:



Also try this:

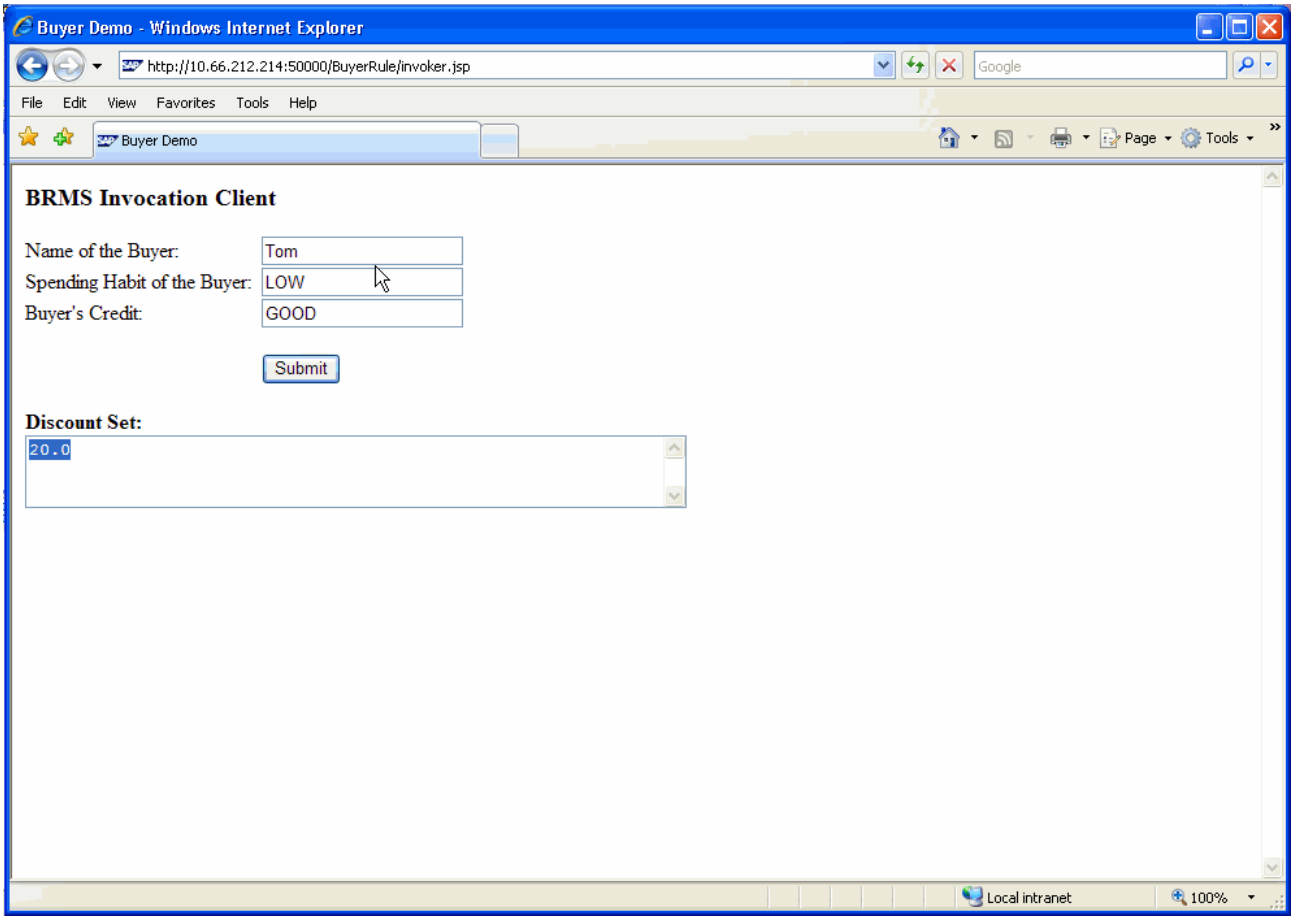
4. Enter the following data in the respective fields:

Field	User Entry
Name of the Buyer	Tom
Spending Habit of the Buyer	LOW
Buyer's Credit	GOOD

5. Choose *Submit*.

   You should get the Discount as 20.0.

Here is a snapshot of the web module:



Related Content

For more information, visit the Business Rules Management homepage.

Copyright