Writing Rules on Java Classes in Rules Composer

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

Summary
This tutorial helps you get started with 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 to calculate discount for buyers.

Authors:  Arti Gopalan and Rekha Narasapuram
Company:  SAP Labs India
Created on: 28 May 2008
Table of Contents

Prerequisites ....................................................................................................................................................... 3

Knowledge Required ....................................................................................................................................... 3

Software Requirements ................................................................................................................................... 3

Procedure ........................................................................................................................................................... 3

Creating the Rules Composer DC .................................................................................................................. 3

Adding the Classes ......................................................................................................................................... 4

Creating the Web Module ............................................................................................................................... 4

Creating Classes in the Web Module ............................................................................................................... 5

Defining the Public Parts .................................................................................................................................. 8

Adding Dependencies to the Rules Composer DC .......................................................................................... 9

Adding Classes to the Rules Composer DC .................................................................................................... 11

Creating the Ruleset ..................................................................................................................................... 13

Creating the Rules ........................................................................................................................................ 14

Editing Any Buyer rule .................................................................................................................................. 15

Editing Big Buyer rule .................................................................................................................................... 18

Editing No Credit Buyer rule .......................................................................................................................... 19

Deploying the Rules ..................................................................................................................................... 20

Executing the Rules ...................................................................................................................................... 21

Creating the Web Module ............................................................................................................................... 21

Adding Dependencies to the Web Module ..................................................................................................... 21

Creating EngineInvoker.java ......................................................................................................................... 21

Creating the BuyerDemo.jsp ........................................................................................................................ 23

Creating the index.jsp ................................................................................................................................... 24

Creating the invoker.jsp ................................................................................................................................. 25

Creating the Enterprise Application ............................................................................................................. 26

Adding Dependencies to the Enterprise Application .................................................................................... 27

Creating the application.xml .......................................................................................................................... 27

Building and Deploying ............................................................................................................................... 28

Running the Web Module .............................................................................................................................. 28

Related Content .......................................................................................................................................... 29

Copyright ...................................................................................................................................................... 30
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 Development Component.
   For example choose 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.
In the *Project Explorer* view, you should see the *buyerrules* node as shown below:

![Project Explorer View](image)

**Adding the Classes**

**Creating the Web Module**

In the SAP NetWeaver Developer Studio, choose *File > New > Project.*

In the wizard that appears, expand the *Development Infrastructure* node and choose *Development Component.* Choose *Next.*

In the screen that appears, expand the *J2EE* node and choose *Web Module.* Choose *Next.*

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.*

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 as shown below:

![Project Explorer view showing buyer_wm node](image)

**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

1. Make sure you are in the Java EE perspective.
2. 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. In the dialog box that appears, choose Class.

1. In the screen that appears, enter com.sap.buyer in the Package field.
2. Enter Buyer in the Name field and choose Finish.
3. You should see the com.sap.buyer node under the Java Resources: source node.
4. Double-click Buyer.java under the com.sap.buyer node.
The Buyer.java window appears with the lines as shown below:

```
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;
    }
```

5. Delete all existing lines and copy the following lines into the window:

**Syntax**

```
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;
}

6. Press Ctrl+Shift+F.

The result must be as follows:

7. Save the changes.
Defining the Public Parts

Make sure you are in the Development Infrastructure perspective.

1. In the Component Browser view, expand the Local Development node, MyComponents [demo.sap.com] node (the software component in which Development Component has been created), and double-click 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 as shown below:

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 as shown below:

![Entity administration of Public Part public](image)

6. Choose *Finish*.

**Adding Dependencies to the Rules Composer DC**

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

---

*SAP DEVELOPER NETWORK | sdn.sap.com  BUSINESS RULES EXPERT COMMUNITY | © 2008 SAP AG 9*
4. In the screen that appears, under **Dependency Details** section, select the **Design Time**, **Deploy Time** and **Run Time** checkboxes as shown below:

![Adding Dependencies](image)

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 tab.

3. In the dialog box that appears, expand the com.sap.buyer node and double-click Buyer as shown below:

5. Under the *Aliases Name* section, expand the *Buyer* node and select all the relevant classes as shown below:

6. Save the changes.
Creating the Ruleset

1. In the Project Explorer view, expand the buyerrules node and in the context menu of the Rules Modeling node, choose New Ruleset.

2. In the dialog box that appears, enter discontrules in the field. Choose OK.

In the Project Explorer view you should see the discontrules node under the Rules Modeling node as shown below:
Creating the Rules

1. In the Project Explorer view, expand the buyerrules node, the Rules Modeling node and in the context menu of the discountrules node, choose New Rule.

2. In the dialog box that appears, enter Any Buyer in the field and choose OK.

You should see the Any Buyer node in the Outline view.

Repeat the steps to create two more rules: Big Buyer and No Credit Buyer.

In the Outline view the result must be as follows:

3. Save the Changes.
Editing Any Buyer rule

1. In the Outline view, choose the Any Buyer node.
2. In the Rule Editor that appears, under If section, choose the Add icon.
   The default condition: Operation.isSuccessful Equals true appears.
3. Edit the default condition as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyersname as shown below:
     - Leave the comparator: Equals as it is.
• Choose the RHS value: Default Value and in the inline textbox enter * as shown below:

4. Add another condition and edit as follows:
   • Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyersspendinghabit.
   • Leave the comparator: Equals as it is.
   • Choose the RHS value: Default Value and in the inline textbox enter *.

5. Add one more condition and edit as follows:
   • Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyerscredit.
   • Leave the comparator: Equals as it is.
   • Choose the RHS value: Default Value and in the inline textbox enter *.

6. Under Then section, choose the Add icon and in the drop down menu that appears, choose Execute.
7. Choose <Action Method> and in the drop down menu, choose `Buyer.setSetdiscount` as shown below:

8. Choose `(double)` and in the inline text enter the value: 10

The result must be as shown below:
Editing Big Buyer rule

1. In the Outline view, choose the Big Buyer node.

2. In the Rule Editor that appears, under If section, choose the Add icon.
   The default condition: Operation.isSuccessful Equals true appears.

3. Edit the default condition as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyersname.
   - Leave the comparator: Equals as it is.
   - Choose the RHS value: Default Value and in the inline textbox enter ".

4. Add another condition and edit as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyersspendinghabit.
   - Leave the comparator: Equals as it is.
   - Choose the RHS value: Default Value and in the inline textbox enter good.

5. Add one more condition and edit as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyerscredit.
   - Leave the comparator: Equals as it is.
   - Choose the RHS value: 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, choose Execute.

7. Choose < Action Method > and in the drop down menu, choose Buyer.setSetdiscount ( {double} ).

8. Choose { (double) } and in the inline text enter the value: 20
Editing No Credit Buyer rule

1. In the Outline view, choose the No Credit Buyer node.
2. In the Rule Editor that appears, under If section, choose the Add icon.
   The default condition: Operation.isSuccessful Equals true appears.
3. Edit the default condition as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyersname.
   - Leave the comparator: Equals as it is.
   - Choose the RHS value: Default Value and in the inline textbox enter *.
4. Add another condition and edit as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyersspendinghabit.
   - Leave the comparator: Equals as it is.
   - Choose the RHS value: Default Value and in the inline textbox enter *.
5. Add one more condition and edit as follows:
   - Choose the LHS value: Operation.isSuccessful and in the drop down menu choose Buyer.getBuyerscredit.
   - Leave the comparator: Equals as it is.
   - Choose the RHS value: 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, choose Execute.
7. Choose < Action Method > and in the drop down menu, choose Buyer.setSetdiscount ( {double} ).
8. Choose \(\{\text{double}\}\) and in the inline text enter the value : 0

The result must be as shown below:

![Rules Composer screenshot](image)

**Deploying the Rules**

**Prerequisites**

You should have a running instance of SAP AS, and should have configured the SAP NetWeaver Developer Studio with this instance.

**Procedure**

1. In the *Project Explorer* view, in the context menu of the *buyerrules* node, choose *Development Component > Build*

2. In the dialog box that appears, make sure the *buyerrules* checkbox is selected and choose *OK*. To check if the build has happened successfully, check the *Infrastructure Console*.

   **Note:** If the *Infrastructure Console* is not open, choose *Window > Show View > Other* and in the dialog box that appears, expand the *Development Infrastructure* node and choose *Infrastructure Console* and then choose *OK*.

3. In the context menu of *buyerrules* node, choose *Development Component > Deploy*.

4. In the *Deploy DCs* dialog box that appears, select the *buyerrules* checkbox and choose *OK*.

   **Note:** Open the *Infrastructure Console*, to check if the deploy has happened successfully.

A dialog box saying *Deploy finished successfully* appears.
Executing the Rules

Creating the Web Module
We have already created a Web Module named *buyer_wm*.
More Information: Adding the Classes

Adding Dependencies 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 does 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 Local Development node, 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 the Dependencies tab.
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.

**Note:** In the context menu of the buyer_wm node, choose Sync/Create Project > Sync Used DCs. In the dialog box that appears, choose OK.

Creating EngineInvoker.java
Make sure that you are in the Java EE perspective.
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.
5. In the context menu of com.sap.helper, choose New > Other
6. In the wizard that appears, choose Class. Choose Next.
7. In the screen that appears, enter EngineInvoker in the Name field. Choose Finish.
   You should see the EngineInvoker.java window.
8. Delete all existing lines and copy the following lines into the window:

**Syntax**

```java
package com.sap.helper;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import com.sap.brms.qrules.ejb.RuleEngineHome;
import com.sap.brms.qrules.engine.RuleEngine;
import com.sap.brms.qrules.engine.RulesetContext;

public class EngineInvoker
{
    private static String jndiName = "com.sap.brms.RuleEngine";
    private static String payloadSeparator;
    private static String ret_payloadSeparator;
    private static final String PROPS_FILE = "engine.properties";
    public EngineInvoker()
    {
    }
    public static RuleEngine getRuleEngine()
            throws Exception
    {
        InitialContext context = new InitialContext();
        Object obj = context.lookup(jndiName);
        RuleEngineHome home = (RuleEngineHome)PortableRemoteObject.narrow(obj,
            RuleEngineHome.class);
        return (RuleEngine)home.create();
    }
    public static List invokeRuleset(String projectName, String rsName, List input)
    {
        List output = new ArrayList();
        RuleEngine ruleEngine;
        if(projectName == null || rsName == null || input == null)
        {
            output
                .add("Project Name or Ruleset Name or Payload should not be NULL");
        }
        try
        {
            ruleEngine = getRuleEngine();
            output = ruleEngine.invokeRuleset(projectName, rsName, input);
        }
        catch(Exception e)
        {
            output.add(e.getMessage());
        }
```

SAP DEVELOPER NETWORK | sdn.sap.com
© 2008 SAP AG 22
return output;
}
public static void main(String args[]){
}
}

Creating the BuyerDemo.jsp
1. In the Project Explorer view, expand the web module: buyer_wm node and in the context menu of the WebContent node, choose New > Other
2. In the dialog box that appears, expand the Web node and choose JSP. Choose Next
3. In the screen that appears, enter BuyerDemo in the File name field.
   You should see the BuyerDemo.jsp window with the following lines:

Syntax
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
</body>
</html>

5. Replace <title>Insert title here</title> with <title>Buyer Demo</title>.
6. Copy the following lines after <body>

Syntax
<form name="ui" method="POST" action="invoker.jsp">
<h3>BRMS Invocation Client</h3>
<% String buyersname = (String) request.getAttribute("BUYERS_NAME");
String buyersspendinghabit = (String) request.getAttribute("SPENDING_HABIT");
String buyerscredit = (String) request.getAttribute("BUYERS_CREDIT");
String setdiscount = (String) request.getAttribute("DISCOUNT_SET");

if(buyersname == null){
    buyersname = "";
}
if(buyersspendinghabit == null){
    buyersspendinghabit = "";
}
if(buyerscredit == null){
    buyerscredit = "";
}
if(setdiscount == null){
    setdiscount = "";
}%>
<table cellpadding="0" cellspacing="0" border="0">
<tr>
<td>Name of the Buyer:</td>
<td><input type="text" name="BUYERS_NAME" value="<%=buyersname %>"></td>
</tr>
<tr>
<td>Spending Habit of the Buyer:</td>
<td><input name="SPENDING_HABIT" type="text" value="<%=buyersspendinghabit %>"></td>
</tr>
<tr>
<td>Buyer's Credit:</td>
<td><input name="BUYERS_CREDIT" type="text" value="<%=buyerscredit %>"></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><input name="INVOKE" type="submit" value="Submit"></td>
</tr>
</table>
<br/>
<strong>Discount Set:</strong><br/>
<textarea name="DISCOUNT_SET" style="width: 503px; height: 50px;"textarea><%=setdiscount %></textarea>
</form>

7. Save the changes.

Creating the index.jsp

1. Expand the web module: buyer_wm node and in the context menu of the Web Content node, choose New > Other
2. In the dialog box that appears, expand the Web node and choose JSP. Choose Next
3. In the screen that appears, enter index in the File name field. Choose Next.

You should see the index.jsp window with the following lines:

Syntax

```html
%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
1DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert the title here</title>
</head>
<body>
```
5. Replace `<title>Insert title here</title>` with `<title>Buyer Demo</title>.

6. Copy the following line after `<body>`:

   **Syntax**
   ```java
   <jsp:forward page="BuyerDemo.jsp"/>
   ```

7. Save the changes.

**Creating the invoker.jsp**

1. Expand the web module: `buyer_wm` node and in the context menu of the Web Content node, choose New > Other

2. In the dialog box that appears, expand the Web node and choose JSP. Choose Next

3. In the screen that appears, enter `invoker` in the File name field. Choose Next.


   You should see the `invoker.jsp` window with the following lines:

   **Syntax**
   ```java
   <%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Insert the title here</title>
   </head>
   <body>
   </body>
   </html>
   ```

5. Replace `<title>Insert title here</title>` with `<title>Buyer Demo</title>.

6. Copy the following lines before `<html>`

   **Syntax**
   ```java
   <%@ page import="com.sap.helper.*"%>
   <%@ page import="com.sap.brms.qrules.engine.*" %>
   <%@ page import="java.util.*"%>
   <%@ page import="com.sap.buyer.*"%>
   ```

7. Copy the following line after `<body>`:

   **Syntax**
   ```java
   Buyer buyerObj = new Buyer();
   String buyersname = (String) request.getParameter("BUYERS_NAME");
   String buyersspendinghabit = (String) request.getParameter("SPENDING_HABIT");
   String buyerscredit = (String) request.getParameter("BUYERS_CREDIT");
   String setdiscount = (String) request.getParameter("DISCOUNT_SET");
   ```
buyerObj.setBuyerscredit(buyerscredit);
buyerObj.setBuyersname(buyersname);
buyerObj.setBuyersspendinghabit(buyersspendinghabit);

// set project name and ruleset name.
String projectName = "demo.sap.com-buyerrules";
String rulesetName = "discountrules";

// invoking the rule engine
List inputList = new ArrayList();
inputList.add(buyerObj);
Object obj = null;
if (inputList.size() != 0) {
    List output = EngineInvoker.invokeRuleset(projectName, rulesetName, inputList);
    if (output.size() == 0) {
        obj = "no output returned. input =" + inputList.size();
    } else {
        obj = output.get(0);
    }
} else {
    obj = "input has not been set";
}
//Object obj = output.get(0);
//obj = "Test";
if (obj instanceof String) {
    request.setAttribute("DISCOUNT_SET", obj);
} else if (obj instanceof Buyer) {
    Buyer buyer = (Buyer) obj;
    request.setAttribute("BUYERS_NAME", buyer.getBuyersname());
    request.setAttribute("DISCOUNT_SET", "" + buyer.getSetdiscount());
    request.setAttribute("SPENDING_HABIT", buyer.getBuyersspendinghabit());
    request.setAttribute("BUYERS_CREDIT", buyer.getBuyerscredit());
}%>
<jsp:forward page="BuyerDemo.jsp"/>

8. Save the changes.

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 Components [demo.sap.com]. Choose Next.
5. In the screen that appears, enter buyer_ear in the Name field. Choose Next.
6. Choose Next.
7. In the screen that appears, 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 Dependencies to the Enterprise Application

Make sure that you are in the Development Infrastructure perspective.

1. In the Component Browser view, expand the Local Development node, 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_ear node, choose Sync/Create Project > Sync Used DCs
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

Note: Expand the enterprise application: buyer_ear node, META-INF node double click the application.xml node.

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

Syntax

```xml
<?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>
```


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 BuyerDemo.
Building and Deploying

Make sure you are in the Development Infrastructure perspective.

1. In the Component Browser view, expand the Local Development, MyComponents[demo.sap.com] nodes and in the context menu of the buyer_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 node, 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 buyer_wm and buyer_ear in the Java EE perspective.

6. In the Project Explorer view, in the context menu of the buyer_wm and buyer_ear nodes, choose Development Component > Build.

7. In the dialog box that appears, choose OK.

8. In the context menu of the buyer_ear node, choose Development Component > Build.

9. 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 number and the application name: BuyerDemo.

2. Enter relevant data in all available fields and choose Submit.

The rules gets executed and you should see the discount set based on the data you entered.

Example

i. Enter the following data in the respective fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>User Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the Buyer</td>
<td>John</td>
</tr>
<tr>
<td>Spending Habit of the Buyer</td>
<td>good</td>
</tr>
<tr>
<td>Buyer's Credit</td>
<td>good</td>
</tr>
</tbody>
</table>

ii. Choose Submit.

You should get the Discount Set as 20 because in the BigBuyer rule, if the Name of the Buyer is *, Spending Habit of the Buyer is good and Buyer's Credit is good then the discount set is 20.
Here is the snapshot of the web module:

**BRMS Invocation Client**

Name of the Buyer: John  
Spending Habit of the Buyer: good  
Buyer's Credit: good  

Discount Set:  
20.0

Related Content  
For more information, visit the [Business Rules Management homepage](http://www.sap.com).