How to Guide for Creating, Building, Deploying and Running a BRM Application in CE 7.1

Applies to:

CE 7.1 SP1. For more information, visit the Business Rules Management homepage.

Summary

This article will help you to get start with BRM (Business Rules Management), includes how to invoke the rules, adding dependencies with UIs and how to deploy it to the server.

Author: Venkata Phaneendra Garimella

Company: Yash Technologies

Created on: 17 September 2009

Author Bio

Venkata Phaneendra working as an Associate Consultant, Process Integration, in Yash Technologies.

Table of Contents

BRM Introduction:

Before we move on to BRM, first we will talk about Business Rules – Business Rules describe the policies or constraints which define any business decision of an organization. To explain it further, business rules represent the core business logic of each organization; they guide and control the basic business processes that form the backbone of any business transaction.

BRM Definition:

"Business rules management (BRM) is a management discipline helps business organizations standardize and optimize their business rules in order to achieve greater visibility, centralization and consistency in key business decisions."

Software Requirements

- 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.

Step's Involved

- First of all we need to create a **Rules Composer** Development Component
- Next is to create a **Web module** to define the variable and UIs
- Finally we need to create an **EAR** project which we need to deploy to the server, which has reference to the Web Module.

Creating Rules Composer:

In order to go with BRM we have to define rules in the Rules Composer perspective. To open the Rules Composer perspective, go to **Window→ Open Perspective → Other** and then select the Rules Composer option in the opened dialog box.

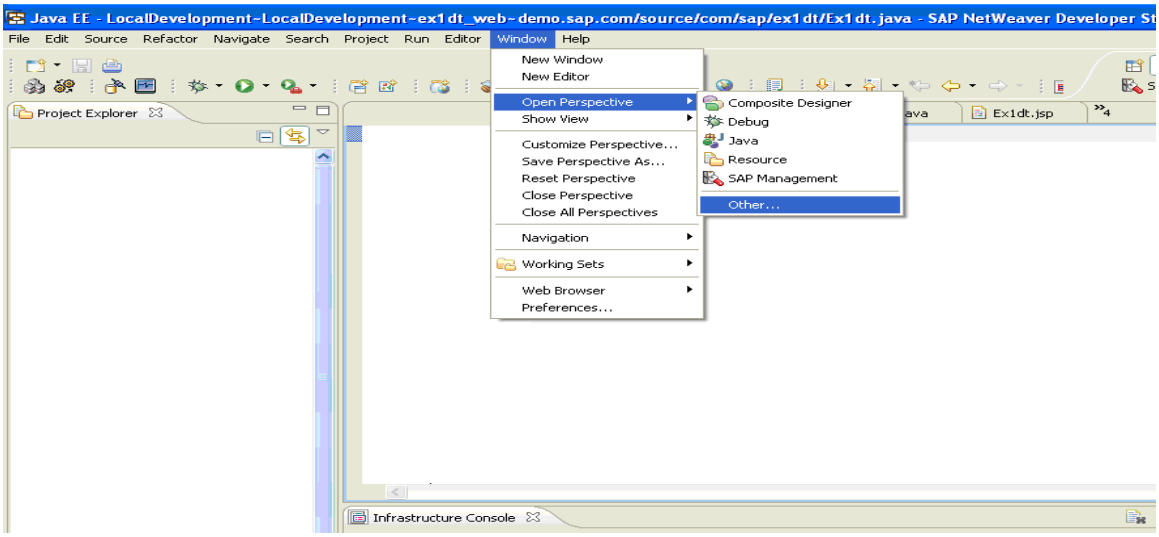

Figure 1.

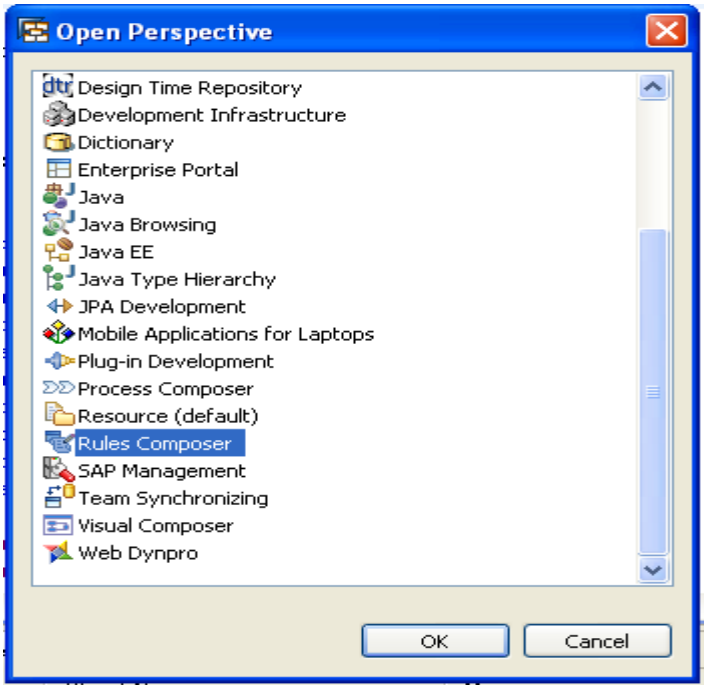

Figure 2.

Click on OK and the Rules Composer perspective will open.

Now we need to create a new project, for which we should go to **New→ Project,** there we need to select Rules Composer, under that Rules Composer Development Component.

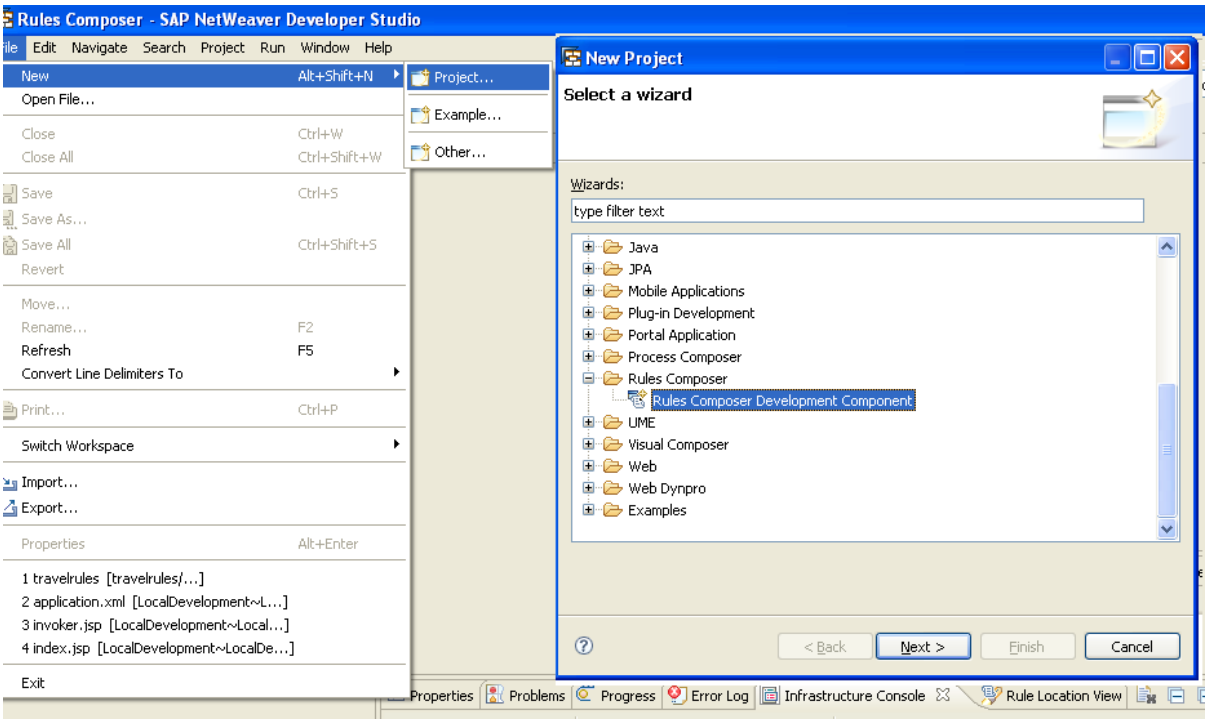

Figure 3.

Choose **Next** and the following screen will appear where we need select the software component where we want to create our Development Component. In our Example it is Local Development, under that My Components [demo.sap.com], choose **Next,** and in the appearing screen give the name to the project, in my example it **travelrules.**

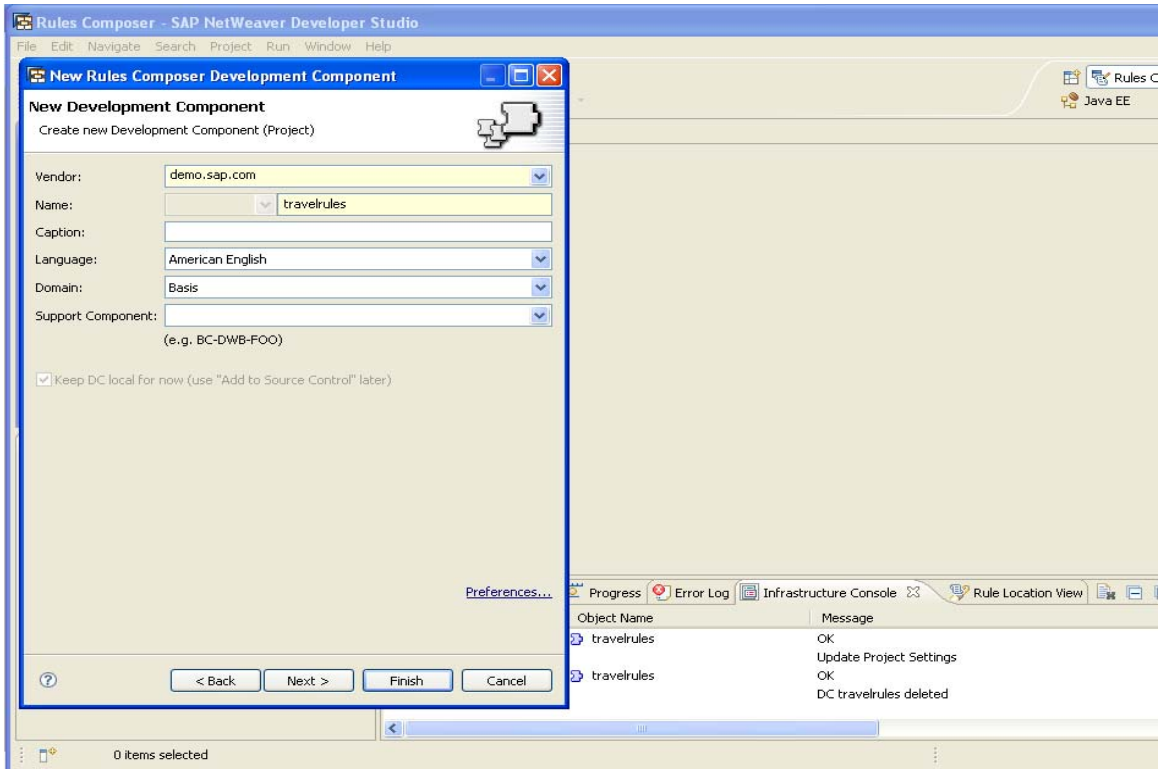

Figure 4.

Choose **Finish.**

In the **Project Explorer** the created project is shown as below:

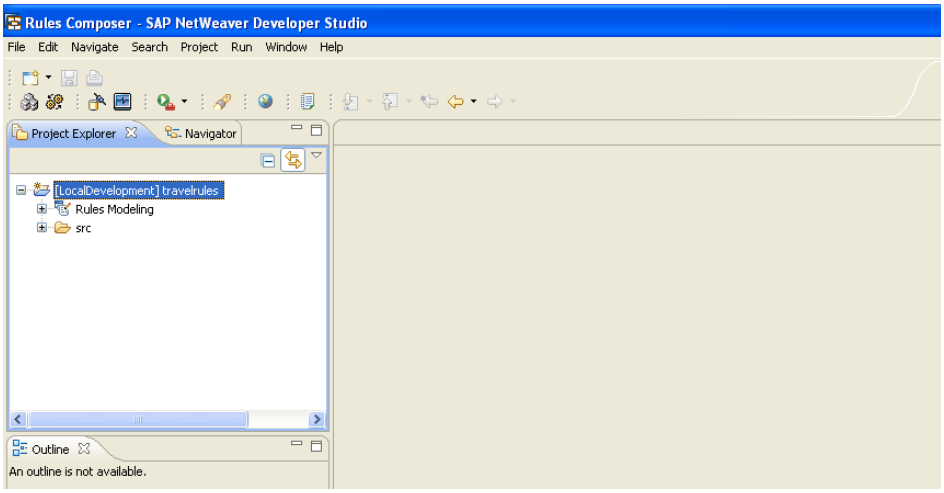

Figure 5.

Creating Web Module:

Before going for the rules definition we have to define the variables, which we are going to use while defining the rules, for this we need to create a web module under which we have to define the classes and JSPs.

For this we have to create a new project as:

File →New → Project, and select Development Component under Development Infrastructure node, choose next in the screen that appears

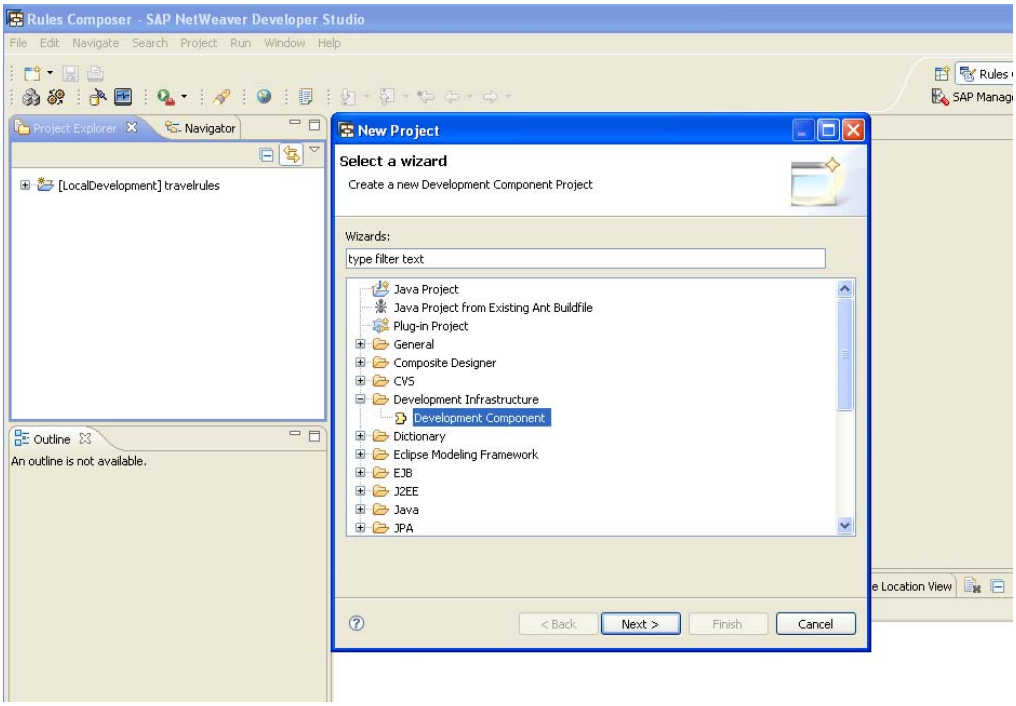

Figure 6.

In the next page that appears, expand J2EE node under that select Web module and choose Next and save it under **Local Development → My Components [demo.sap.com]**.

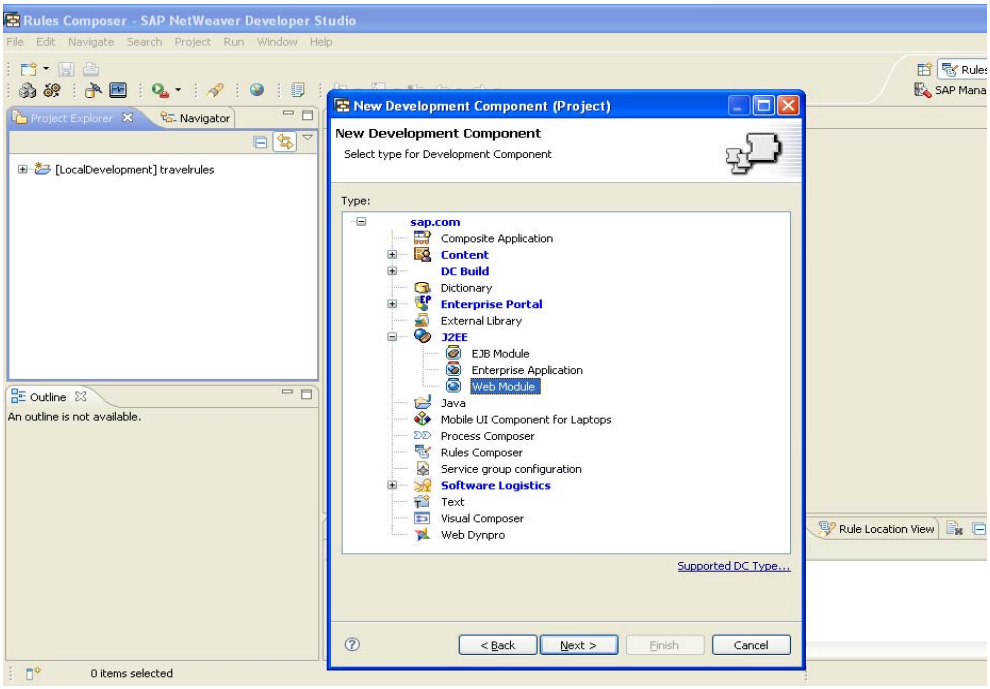

Figure 7.

After choosing the web module as our new Development Component, choose Next and in the following screen give the name as travelrules_web (to identify it as web module) and choose Next/Finish.

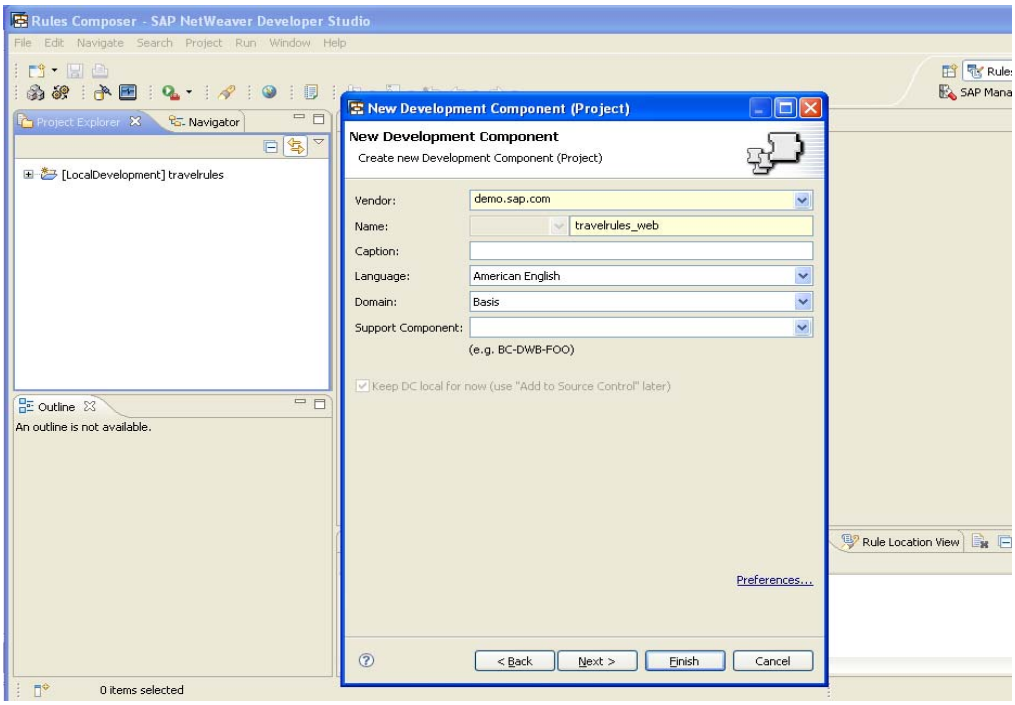

Figure 8.

The created Web Module will look like the following.

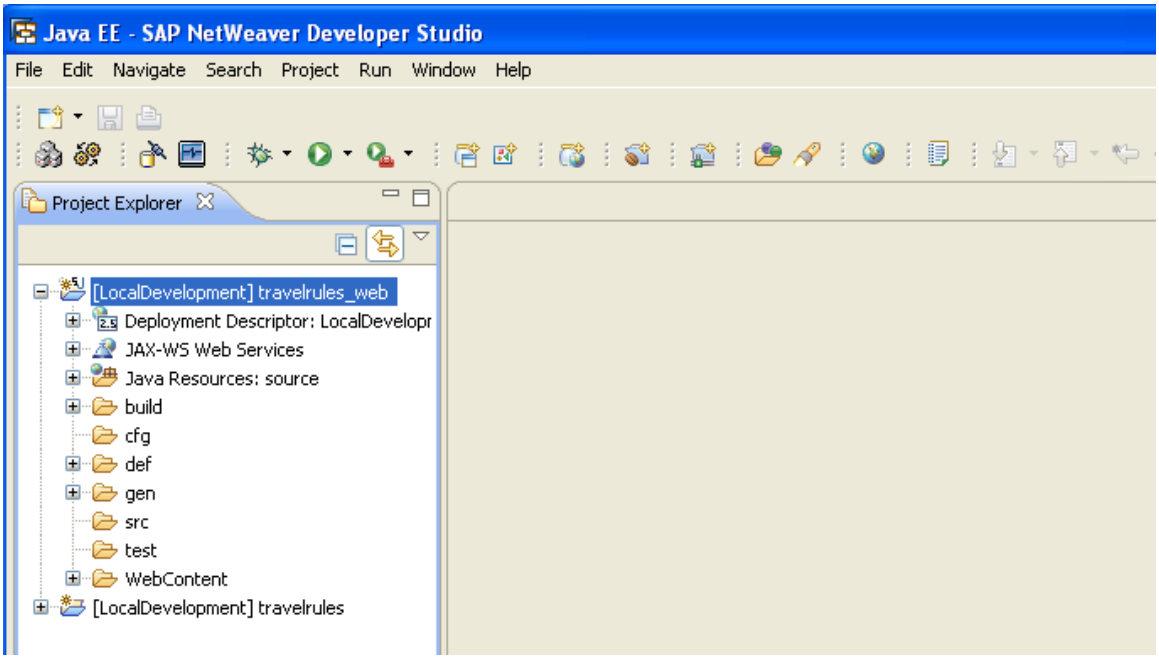

Figure 9.

We need to create a java class under the web module which we have created, in order for the creating, expand travelrules_web node and in the context menu of Java Resources: source node, choose New > Class.

Assign a package for this class named as “com.sap.travelrules” and name our class as “Travelrules”. Add the contents to the “Travelrules.java”.

In this class we have to write setter and getter methods for input and output fields.

In our example, as we are dealing with Travel Rules which can defined using the grade of the Employee and Trip location.

Hence Employee name, Grade and Trip type will be the input parameters. And all the allowances like airfare, daily allowances, food allowances, and accommodation etc., will be our output parameters.

As well in this class we will provide setter and getter methods for the all the I/O parameters.

The class will look like the following:

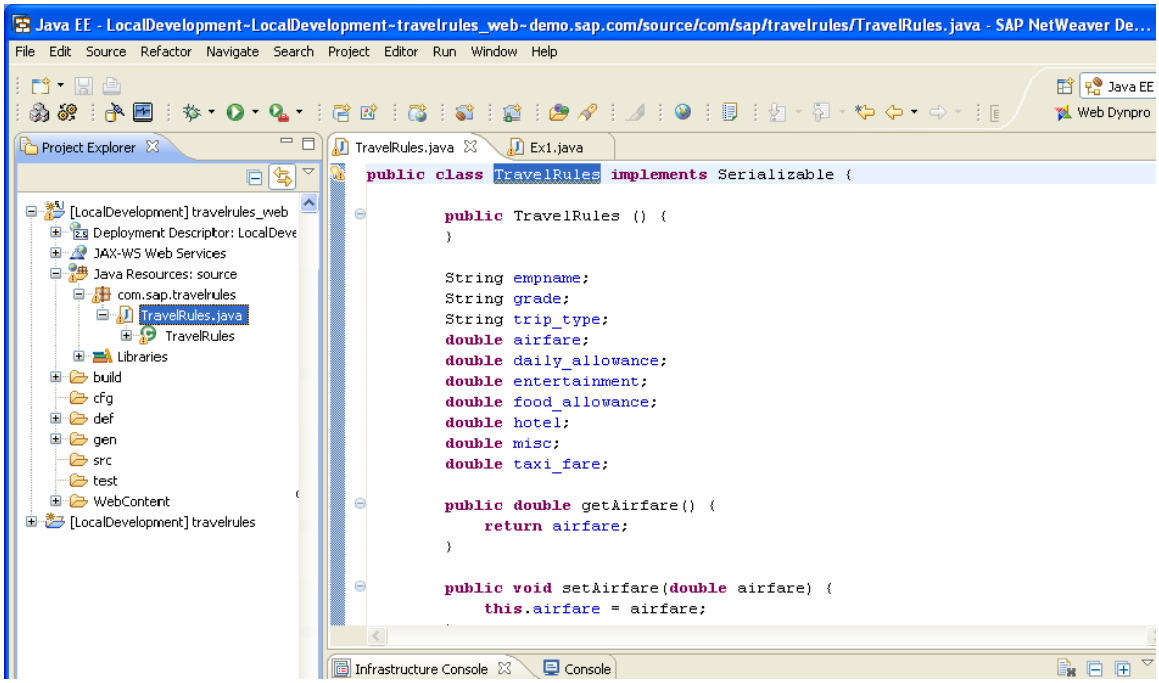

Figure 10.

After creating the class, go to Development Infrastructure perspective, and add public node to it, by choosing Add in the Public Parts tab. The steps for doing this is

- In the Component Browser view, expand the Local Development node, MyComponents [demo.sap.com] node, and select the travelrules_web node.
- In the Component Properties view, choose the Public Parts tab and in the page that appears, choose Add.
- In the screen that appears, enter public in the Name field. Choose Finish.

After adding the public node, it will appear as the Figure 11.
Next in the context menu of the public node, choose Manage Entities and

in the screen that appears, under the Entities section, expand the Java Class, com, sap and TravelRules nodes, select the TravelRules java class checkbox and choose Finish. It will look like as Figure 12.

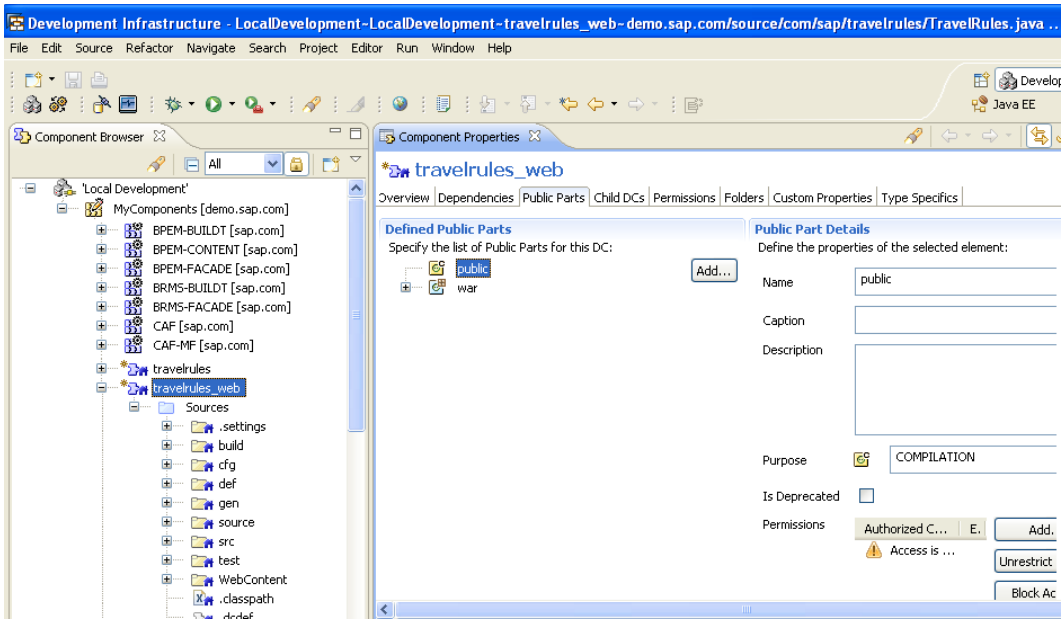

Figure 11.

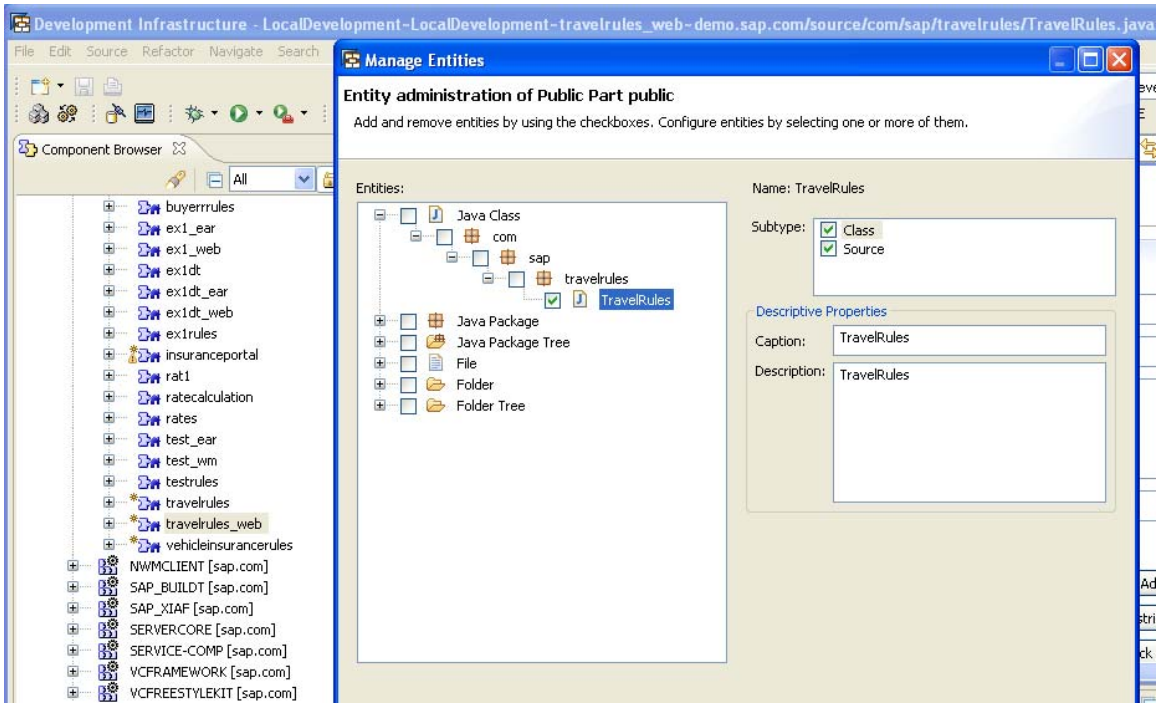

Figure 12.

After this we have to add Dependencies between our Development components. For this,

- In the Component Browser view, expand the Local Development node, MyComponents [demo.sap.com] node and double-click the TravelRules node.
- In the Component Properties view, choose the Dependencies tab and in the page that appears, choose Add.
- In the dialog box that appears, expand the MyComponents node and select the travelrules_web checkbox. Choose Next.
- In the screen that appears, under Dependency Details section, select the Design Time, Deploy Time and Run Time checkboxes.
- Choose Finish.
- In the Dependencies tab page, expand the travelrules_web node and choose the war node. Choose Remove and in the dialog box that appears choose Yes.
- In the Component Browser view, expand the MyComponents [demo.sap.com] node and in the context menu of the travelrules node, choose Build. Choose OK.

Adding Classes to the Rules Composer DC

Get into Rules Composer perspective.

- In the Project Explorer view, expand the travelrules node, the Rules Modeling node and double-click the Aliases node.
- In the Project Aliases Editor that appears, choose the Class Aliases tab and in the tab page that appears, choose the Add Classes tab.
- In the dialog box that appears, expand the com.sap.travelrules node and double-click travelrules as shown in the Figure 13.
- Choose Finish.
- Under the Aliases Name section, expand the TravelRules node and select all the relevant classes.
- Save the changes.

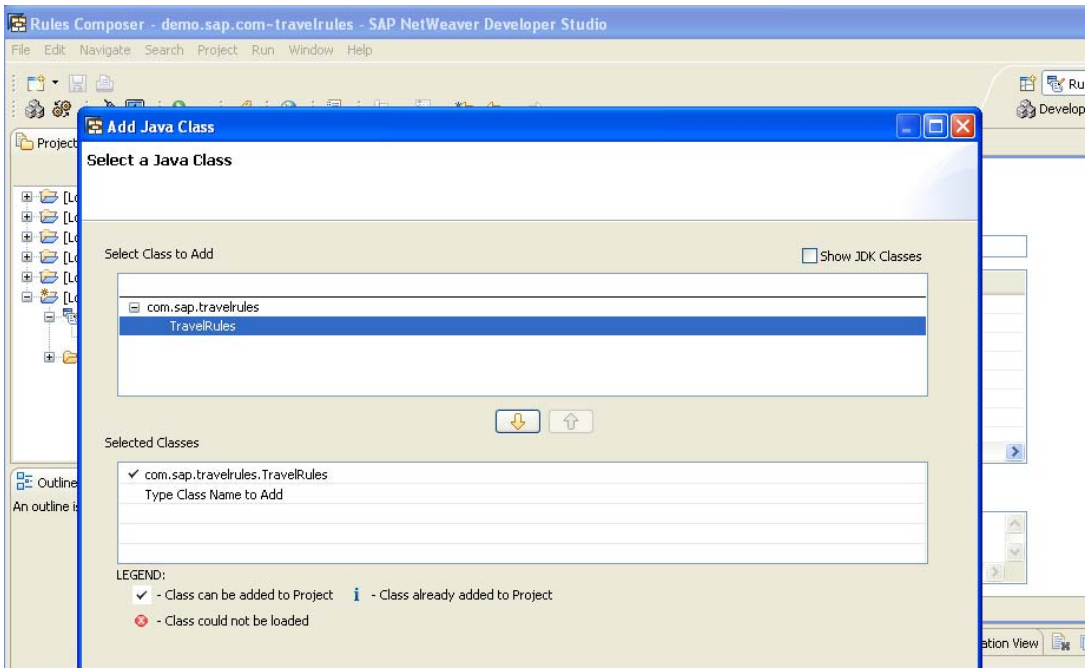

Figure 13.

Creating the Rule set

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

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

In the Project Explorer view you should see the travel_rules node under the Rules Modeling node as shown in the Figure 14.

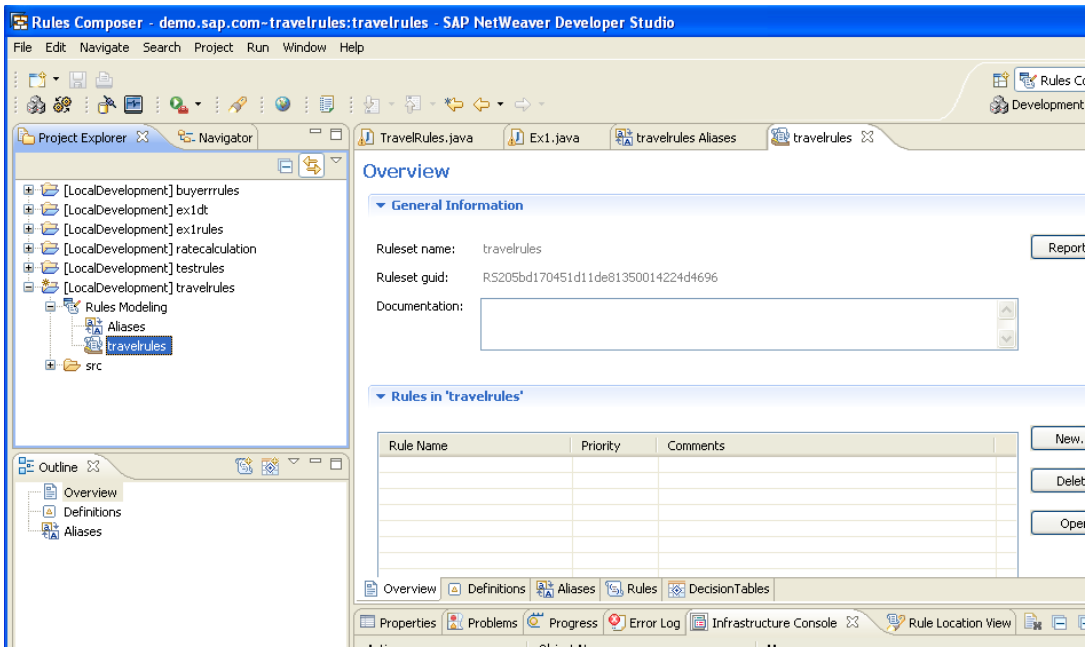

Figure 14.

Creating the Rules

In the Project Explorer view, expand the travelrules node, the Rules Modeling node and in the context menu of the travel_rules node, choose New Rule.

In the dialog box that appears, enter M1 in the field and choose OK.

You should see the M1 node in the Outline view.

Repeat the steps to create more rules, depending upon the requirement (in our case for M2 and M3).

The rules will be displayed, as shown in the Figure 15.

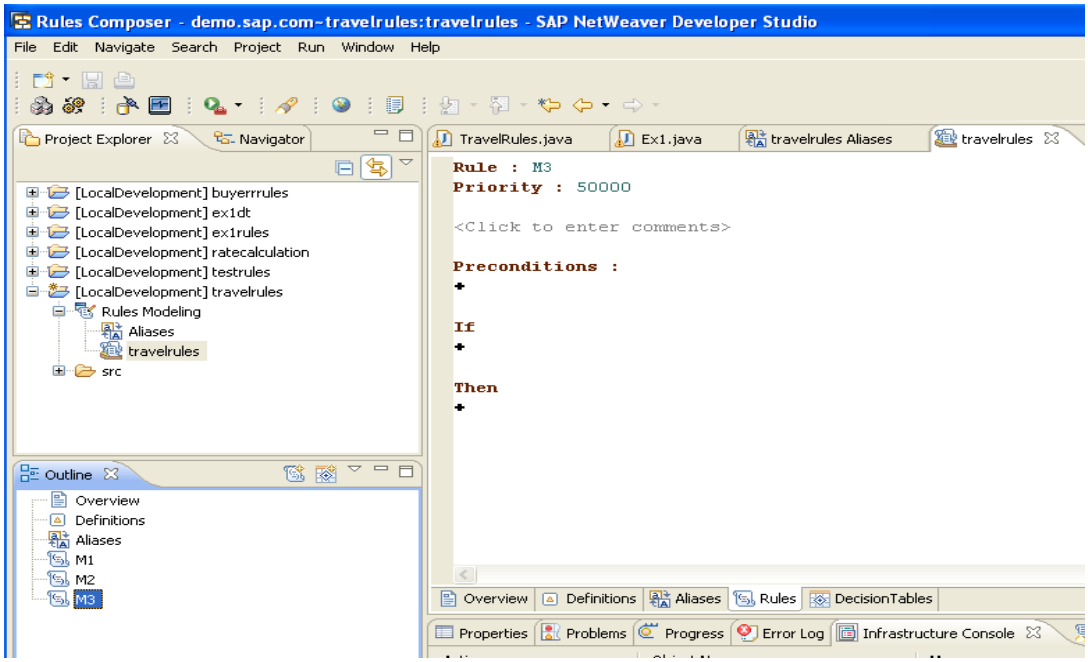

Figure 15.

Now we have to edit all the rules, as per the given requirements.

In our case the rule for M1 is like the following.

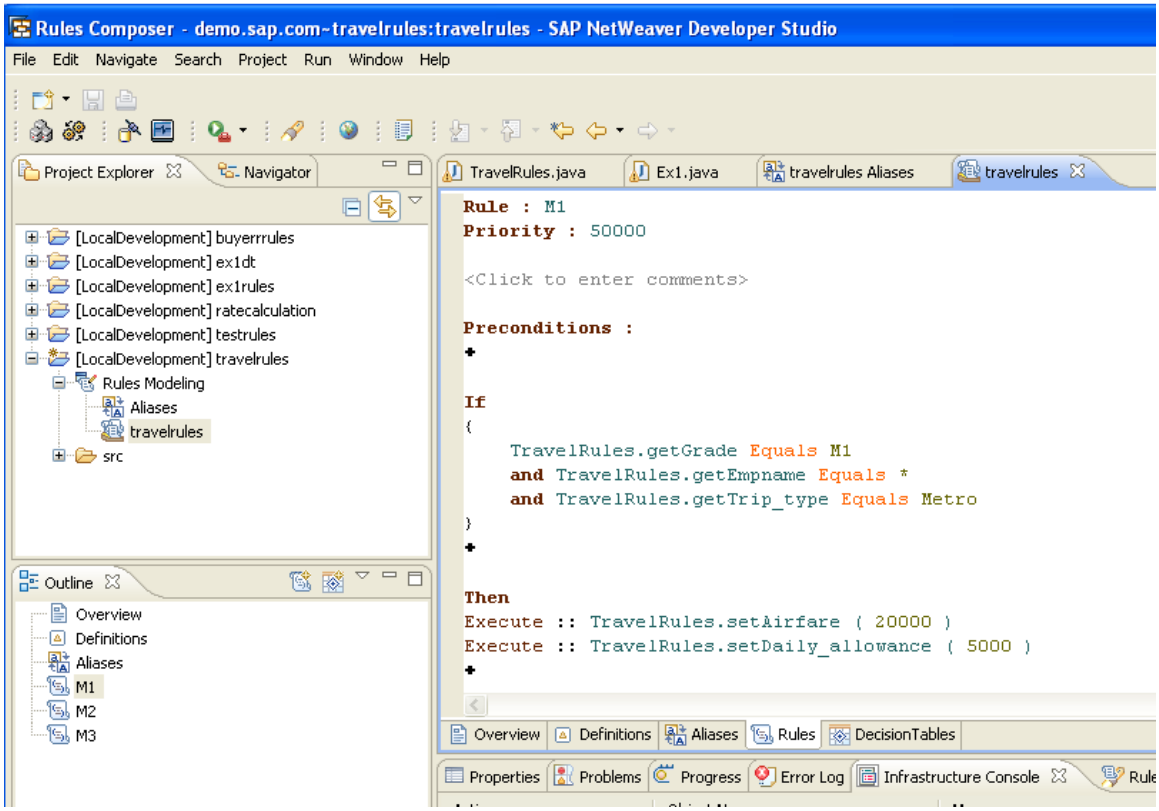

Figure 16.

Just we need to repeat the above step in order to edit the remaining rules.

After editing all the rules we need to execute the Rules.

Executing the Rules

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

Adding Dependencies to the Web Module

- Open *Development Infrastructure*.
- In the Component Browser view, expand the Local Development node, MyComponents [demo.sap.com] node and choose the travelrules_web node.
- In the Component Properties view, choose the Dependencies tab.
- 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.
- In the screen that appears, select the Design Time, Deploy Time, Run Time checkboxes. Choose Finish.

Creating EngineInvoker.java

Open *Java EE* perspective.
Expand the web module: travelrules_web node and in the context menu of the Java Resources: source node, choose New → Package.

- Choose Next.
- In the screen that appears, enter com.sap.helper in the Name field.
- Choose Finish.
- In the context menu of com.sap.helper, choose New > Class
- Choose Next.
- In the screen that appears, enter EngineInvoker in the Name field. Choose Finish.
- The code for EngineInvoker is in the following attachment.

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());
      }
   return output;
      }
      public static void main(String args[]){
      }
      }
```

After this we have to create “index.jsp”, travelrules.jsp and invoker.jsp under the we module travelrules_web.

Creating the Enterprise Application

- 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 *Enterprise Application*. 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 travelrules_ear in the *Name* field. Choose *Next*.
- Choose *Next*.
- In the screen that appears, select the LocalDevelopment~ LocalDevelopment~buyer-wm.demo.sap.com check box.
- Choose **Finish.**

Adding Dependencies to the Enterprise Application

- Open Development Infrastructure perspective.
- In the Component Browser view, expand the Local Development node, MyComponents[demo.sap.com] node and choose the travelrules_ear node.
- In the Component Properties view, choose the Dependencies tab.
- 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.
- In the screen that appears, select the Design Time, Deploy Time, Run Time checkboxes. Choose Finish.

Creating the application.xml

Open *Java EE* perspective.

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

The default code for the application.xml is

```
<?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~ex1_ear~demo.sap.com
   </display-name>
   <module>
          <web>
                 <web-uri>demo.sap.com~ex1_web.war</web-uri>
          <context-root>
LocalDevelopment~LocalDevelopment~buyer_wm~demo.sap.com
</context-root>
          </web>
   </module>
</application>
```

In the above file replace the value of <context-root> tag with a meaningful name, which is relevant to our project.

Like in the above case I have replaced the content with “**TravelRules**”.

Building and Deploying

- Open *Development Infrastructure* perspective or Java EE perspective.
- In the *Component Browser* view, expand the *Local Development*, *MyComponents[demo.sap.com]* nodes and in the context menu of the *travelrules_web* and travelrules_*ear* nodes, choose *Build*
- In the dialog box that appears, choose *OK*.
- In the context menu of the *travelrules_ear* node, choose *Deploy*.
- In the dialog box that appears, choose *OK*.
- Open the *Infrastructure Console*, to check if the build and deploy actions have happened successfully.

After successful deployment, we can Run our application using the web browser.

Testing

Open the browser and enter the Application Server Address followed by the port number and the application name: TravelRules.

Ex: http://localhost/TravelRules

Related Content

https://www.sdn.sap.com/irj/sdn/nw-rules-management?rid=/webcontent/uuid/f066ec08-474b-2b10-4a97-b66d605de037#section9

For more information, visit the Business Rules Management homepage

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.