OPENRULES®

Open Source Business Decision Management System

Release 6.2.3

External Rules

OpenRules, Inc.

www.openrules.com

January-2013

Table of Contents

INTRODUCTION

OpenRules® is as an open source Business Decision Management System (BDMS) with proven records of delivering and maintaining reliable decision support software. The detailed description of OpenRules® can be found in the User Manual.

The key component of OpenRules® is an enterprise-level Rule Repository that usually utilizes a popular spreadsheet mechanism to represent business rules placed in regular Excel files. However, along with rules repositories organized as a hierarchy of Excel files, OpenRules® allows you other ways to create and maintain your rule repositories:

- Standard relational databases, described at http://openrules.com/pdf/OpenRulesUserManual.DB.pdf
- External Rules, described at this document.

OpenRules® allows a user to create and maintain their rules outside of Excel-based rule tables. It provides a generic Java API for adding business rules from different external sources such as:

1. Database tables created and modified by the standard DB management tools
2. Live rule tables in memory dynamically modified by an external GUI
3. Java objects of the predefined type `RuleTable`
4. Problem-specific rule sources that implement a newly offered rules provider interface.

With external rules you may keep the business parts of your rules in any external source while the technical part (Java snippets) will remain in an Excel-based template, based on which actual rules will be created by the OpenRulesEngine. For example, you may keep your rules in a regular database table as long as its structure corresponds to the columns (conditions and actions)

of the proper Excel template. Thus, the standard DB management tools, or your own GUI that maintains these DB-based rule tables, de-facto become your own rules management environment.

The external rules may also support a preferred distribution of responsibilities between technical and business people. The business rules can be kept and maintained in a database or other external source by business analysts while developers can continue to use Excel and Eclipse to maintain rule templates and related software interfaces.

API FOR EXTERNAL RULES

OpenRules® provides a generic Java API for external rules. There is a special constructor,

```
OpenRulesEngine(String excelFileName, ExternalRules rules)
```

that has an additional parameter of the predefined Java type `ExternalRules`. You may create an object of this type such as,

```
ExternalRules externalRules = new ExternalRules();
```

and then add different rule tables using the method:

```
addRuleTable(String ruleTableName,
             String ruleTemplateName,
             Object[][] ruleGrid);
```

The complete API is described at OpenRules® API. This simple interface gives a developer the ability to bring rules from any external source and add them to OpenRulesEngine as regular Java objects. If the rules in the external source are changed, a developer may notify the `ExternalRules` object about this change by using the method,

```
externarRules.setModified(true);
```

Then during the next rule engine's run, all rules will be dynamically reloaded.

OpenRules® provides 5 sample projects that demonstrate how to use External Rules:

ExternalRulesFromJava:	shows how to define rules as Java objects
ExternalRulesFromDB:	shows how to define rules in MS Access using JDBC
ExternalRulesFromXML:	shows how to define rules in XML files
ExternalRulesFromExcel:	shows how to define rules as Excel Data tables
ExternalRulesFromGUI:	shows how to build a web application that allows a user to change and execute rules on the fly without a restart

These projects can be found in the complete OpenRules® installation under the section "External Rules". External rules can be invoked from regular rules described in Excel files. Because these external rules are not known until run-time, OpenRules® will produce warnings about these as yet unknown rules, but the OpenRulesEngine will still execute them without problems. To suppress the warnings and to keep track of all participating rules, you may fill out a newly introduced optional table of the type "ExternalRules" that lists names of all external rules along with their templates as in the following example:

ExternalRules	
greetingRules	defineGreeting
salutationRules	defineSalutation

The projects below will produce greetings like "Good Morning, Mrs. Robinson!" based on the current time and different customer's attributes such as gender and marital status. They are similar to the standard project "HelloJavaTemplates" but instead of using Excel-based rule tables they will use external rules.

The business logic for producing greetings and salutations is presented in the Excel file HelloTemplates.xls. The first template

Rules void **defineGreeting**(App app, int hour)		
C1	C2	A1
min <= hour	hour <= max	app.greeting = greeting;
int min	int max	String greeting
Hour From	**Hour To**	**Set Greeting**
		Unknown Greeting

specifies how to define different greetings (Good Morning, Good Afternoon, etc.) based on the hour of the day. If the parameter "hour" belongs to the interval [min;max] defined by a concrete rule, then the attribute "greeting", of the parameter "app" will be set to the proper greeting. If no rules are satisfied, this multi-hit table will use the default greeting "Unknown Greeting".

The second template

Rules void **defineSalutation**(App app, Customer c)			
C1	C2	C3	A1
c.gender.equals (gender)	c.maritalStatus. equals(status)	c.age < age	app.salutation = salutation;
String gender	String status	int age	String salutation
Gender	**Marital Status**	**Age Less Than**	**Set Salutation**
			Unknown Salutation

specifies how to define different salutations (Mr., Mrs., etc.) based on customer attributes Gender, Marital Status, and Age. If no rules are satisfied, this multi-hit table will use the default salutation "Unknown Salutation".

EXTERNAL RULES FROM JAVA

The OpenRulesEngine can be created with an additional parameter of the predefined type ExternalRules that allows for rule tables defined as Java objects. The project "ExternalRulesFromDJava" demonstrates different ways of defining external rules in Java.

Step 1. Defining Rule Tables in Java

All Java classes are typical for basic rule projects. In this project the main Java class `RunExternalRules` shows different ways for adding rule tables to the external rules. Here is the first rule table:

```
externalRules.addRuleTable(
    "ExternalSummerGreeting", //table name
    "defineGreeting",     //template name
    new String[][] {      //rules
          new String[] {"0","11","Good Morning Summer"},
          new String[] {"12","17","Good Afternoon
Summer"},
          new String[] {"18","21","Good Evening Summer"},
          new String[] {"22","24","Good Night Summer"}
    }
);
```

The first parameter specifies the rule table name. The second parameter specifies the template upon which this table will be based. The third parameter defines a grid that is a two-dimensional array, `Object[][]`, containing actual rules. This grid corresponds to the template "defineGreeting" - see above. The first rule in the grid states that IF Hour From is "0" AND Hour To is "11" THEN Set Greeting as "Good Morning Summer", etc.

The second rule table,

```
externalRules.addRuleTable(
    "ExternalWinterGreeting", //table name
    "defineGreeting",     //template name
    new String[][] {      //rules
          new String[] {"0","12","Good Morning Winter"},
          new String[] {"13","16","Good Afternoon
Winter"},
          new String[] {"17","22","Good Evening Winter"},
          new String[] {"23","24","Good Night Winter"}
    }
);
```

is very similar to the first one but defines greeting rules for a winter season.

The third rule table,

```
externalRules.addRuleTable(
    "ExternalGreetingHorizontal", //table name
    "defineGreetingHorizontal",   //template name
    new String[][] {          //rules
          new String[] {"0","11","Good Morning"},
          new String[] {"12","16","Good Afternoon"},
          new String[] {"17","22","Good Evening"},
          new String[] {"23","24","Good Night"}
    }
);
```

shows that you may use a horizontal template "ExternalGreetingHorizontal" and still use the same vertical structure of your rules.

The fourth rule table,

```
externalRules.addRuleTable(
  "ExternalGreetingReverseOrder", //table name
  "defineGreeting",           //template name
  new String[] { "A1","C1" }, //labels order differs from template
order
  new Object[][] { // not all cells contains strings but other objects
    new Object[] {"Good Morning", new Integer(0),new
Integer(11) },
    new Object[] {"Good Afternoon",new Integer(12),new
Integer(17)},
    new Object[] {"Good Evening",new Integer(18),new
Integer(21)},
    new Object[] {"Good Night",new Integer(22),new
Integer(24)}
  }
};
```

shows several additional options that could be added to the `ExternalRules` object. First of all, you can use all optional rules and conditions along with other features available for "normal" rules and templates - as described above. The array of Strings,

```
new String[] { "A1","C1" }
```

placed just before the grid informs OpenRulesEngine that this rule table starts with the action A1 followed by the condition C1, thus violating the default column order in the template. The grid `Object[][]` demonstrates the ability to specify not only String but any Java objects as long as they correspond to the types of parameters specified in the template.

If the type of objects inside the rule tables do not correspond to the templates, the proper error will be produced. For example, if you make a mistake in the first rule table by typing "O" instead of "0"

```
new String[] {"O","11","Good Morning Summer"}
```

you will receive a compilation error that will look like this:

```
org.openl.syntax.SyntaxErrorException: Error: For input string:
"O": java.lang.NumberFormatException
at ExternalRules#ExternalSummerGreeting?row=0&column=0&openl=
java.lang.NumberFormatException: For input string: "O"
```

The error message points you to the name of the invalid external table (*ExternalRules#ExternalSummerGreeting*) and to the coordinates of the invalid cells inside the grid (*row=0&column=0*).

Step 2. Executing External Rules from a Java Program

The main file HelloCustomer.xls defines the Environment of our rule project as follows:

Environment	
import.java	hello.*
include	../include/HelloTemplates.xls

This application uses two simple Java beans: "Request" with one integer attribute "hour" and "Response" with one String attribute "result".

The main Java class `RunExternalRules` creates and executes an OpenRulesEngine in the standard way:

```
String fileName = "file:rules/main/HelloCustomer.xls";
OpenRulesEngine engine =
         new OpenRulesEngine(fileName,externalRules);
Response response = new Response();
Request request = new Request();
request.setHour(Calendar.getInstance().get(Calendar.HOUR_
OF_DAY));
Object[] params = new Object[] { request, response };

for (int i = 0; i < externalRules.getRuleTables().size();
i++) {
  RuleTable rules =
(RuleTable)externalRules.getRuleTables().get(i);
  System.out.println(rules);
  engine.run(rules.getTableName(), params);
  System.out.println("Greeting generated by rules '" +
            rules.getTableName() +
          "' for hour " +request.hour +": " +
response.result );
  System.out.println();
}
```

To run the project, you may double-click on the file "run.bat". Here is an expected output:

```
INITIALIZE OPENRULES ENGINE 5.3.0 (build 03092009) for
[file:rules/main/HelloCustomer.xls]
External rules table: ExternalSummerGreeting
External rules table: ExternalWinterGreeting
External rules table: ExternalGreetingHorizontal
External rules table: ExternalGreetingReverseOrder
IMPORT.JAVA=hello.*
INCLUDE=../include/HelloTemplates.xls
[../include/HelloTemplates.xls] has been resolved to
[file:<...>/rules/include/HelloTemplates.xls]

ExternalRules ExternalSummerGreeting template defineGreeting
0 11 Good Morning Summer
12 17 Good Afternoon Summer
18 21 Good Evening Summer
22 24 Good Night Summer

Greeting generated by rules 'ExternalSummerGreeting' for hour 16: Good
Afternoon Summer

ExternalRules ExternalWinterGreeting template defineGreeting
0 12 Good Morning Winter
13 16 Good Afternoon Winter
17 22 Good Evening Winter
23 24 Good Night
```

Greeting generated by rules 'ExternalWinterGreeting' for hour 16: Good Afternoon Winter

```
ExternalRules ExternalGreetingHorizontal template
defineGreetingHorizontal
0 11 Good Morning
12 16 Good Afternoon
17 22 Good Evening
23 24 Good Night
```

Greeting generated by rules 'ExternalGreetingHorizontal' for hour 16: Good Afternoon

```
ExternalRules ExternalGreetingReverseOrder template defineGreeting
Good Morning 0 11
Good Afternoon 12 17
Good Evening 18 21
Good Night 22 24
```

Greeting generated by rules 'ExternalGreetingReverseOrder' for hour 16: Good Afternoon

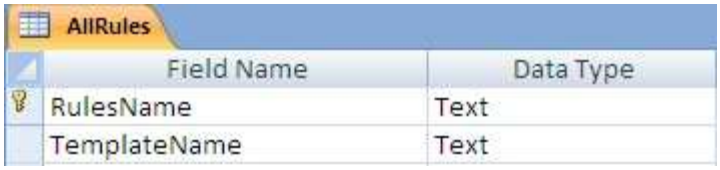
EXTERNAL RULES FROM DATABASE

OpenRules® allows you to keep your business rules in regular database tables whose structures correspond to the columns (conditions and actions) of Excel's templates based on which of the proper rule tables will be executed. This way the standard DB management tools can be used as your own rules management environments.

The project "ExternalRulesFromDB", demonstrates how to define rules in a MS Access database with regular tables (without Excel files saved as blobs). Because we are using a standard JDBC interface, this project should work similarly with other database management systems.

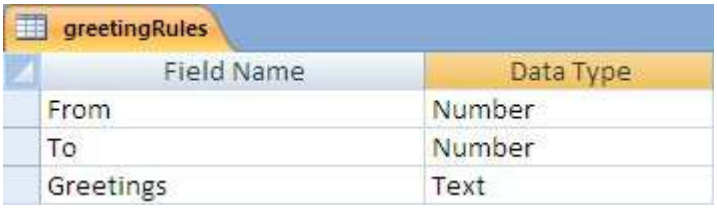
Step 1. Setting Up Database with Rule Tables

Use MS Access to create a new database, labeled "OpenRulesDB", and save it in the subdirectory "DB" of the directory ExternalRulesFromDB. Using MS Access, create the table "AllRules", which looks like this one:



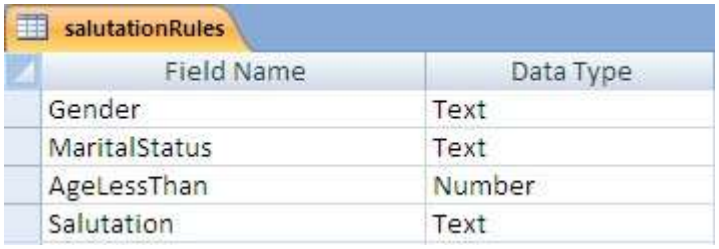
RulesName	TemplateName
greetingRules	defineGreeting
salutationRules	defineSalutation

This DB table has only two text columns "RulesName' and "TemplateName". Now we have to create a simple DB table, "greetingRules", with a structure that corresponds to our template "defineGreeting":



From	To	Greetings
0	11	Good Morning
12	15	Good Afternoon
16	21	Good Evening
22	24	Good Night

Similarly, we will create a table, "salutationRules", that corresponds to our template "defineSalutation":



Gender	MaritalStatus	AgeLessThan	Salutation
Male			Mr.
Male	Single	3	Little
Female	Single		Ms.
Female	Married		Mrs.

To make this database accessible from a Java program we have to setup the proper data source. In Windows, we have to go to Control Panel, open Administrative Tools, and select Data Sources (ODBC). Add a new User Data Source with the following information:

Data Source Name: OpenRulesDB

Description: HelloExternalRulesFromDB

Click on the "Select" button and chose your just created OpenRulesDB.mdb file.

The above DB structure serves only as an example. You may organize your database with rule tables differently with additional information about rule tables including such attributes as "CreatedBy", "CreatedAt", "LastModifiedBy", "LastModifiedAt", "Category", and many more attributes that accommodate your particular needs.

Step 2. Defining a DB interface in Java

To inform an OpenRulesEngine about external rules, you need to create an object of the type ExternalRules and add to it all external RuleTables. Each instance of the class RuleTable consists of:

- rule table name (String)
- template name (String)
- a grid of objects that represent the content of a rule tables (Object[][])

In our case, to create an instance of the class External Rules we should:

1) read all rows from the DB table "AllRules"
2) for every pair (RuleName;TemplateName) find the proper DB table and create the required grid of the type Object[][] for all rows and all columns of the DB table.

To accomplish this, we have created the class OpenRulesDB.java that contains the method "readRuleTables()". This class is inherited from the standard JDBC interface supported by the class DbUtil included in the OpenRules® installation within the project "com.openrules.tools". Here is the code of this method with comments:

```
public synchronized ExternalRules readRuleTables() {

     String mainTable = "AllRules";
     String columnWithRuleNames = "RulesName";
     String columnWithTemplateNames = "TemplateName";
```

```
        ExternalRules externalRules = new ExternalRules();

        try {
          String mainSQL = "SELECT * FROM " + mainTable;
          ResultSet mainRS = executeQuery(mainSQL);
          // FOR ALL RULE TABLES
          while(mainRS.next()) {
                String ruleTableName = mainRS.getString(columnWithRuleNames);
                String ruleTemplateName =
                      mainRS.getString(columnWithTemplateNames);
                System.out.println("Rules " + ruleTableName + " template "
                          + ruleTemplateName);
                try {
                  int numberOfRows = count(ruleTableName);
                  System.out.println("Total " + numberOfRows + " rows");
                  String insideSQL = "SELECT * FROM " + ruleTableName;
                  ResultSet rs = executeQuery(insideSQL);
                  ResultSetMetaData md = rs.getMetaData();
                  int numberOfColumns = md.getColumnCount();
                  System.out.println("Total " + numberOfColumns + " columns");
                  Object[][] grid = new Object[numberOfRows][numberOfColumns];
                  int rowIndex = 0;
                  // FOR ALL TABLE's ROWS
                  while (rs.next()) {
                      Object[] row = new Object[numberOfColumns];
                      // FOR ALL TABLE's COLUMNS
                      for (int i=0; i< numberOfColumns; i++) {
                           // Add grid element
                           row[i] = rs.getObject(i+1);
                           if (row[i] == null)
                              row[i] = "";
                      }
                      grid[rowIndex++] = row;
                  }
                  rs.close();
                  // ADD RULE TABLE
                  externalRules.addRuleTable(ruleTableName,
                                             ruleTemplateName,grid);
                }
                catch(Exception e) {
                    System.err.println("ERROR in the DB table " +
                              ruleTableName + "\n" + e.toString());
                }
          }
          mainRS.close();
        }
        catch(Exception e) {
           System.err.println("ERROR in the DB table " +
                 mainTable + "\n" + e.toString());
                 }
           return externalRules;
        }
}
```

Step 3. Creating and Executing Rules from a Java Program

All other modules are typical for basic rule projects. The main Java file, RunExternalRulesFromDB.java, is used to test the above rules:

```
import com.openrules.ruleengine.ExternalRules;
import com.openrules.ruleengine.OpenRulesEngine;

public class RunExternalRulesFromDB {

  public static void main(String[] args) {
       // Read DB to create ExternalRules
       OpenRulesDB db = new OpenRulesDB();
       ExternalRules externalRules = db.readRuleTables();
       // Create OpenRulesEngine with external rules
       String fileName =
"file:rules/main/HelloCustomer.xls";
       OpenRulesEngine engine =
                  new
OpenRulesEngine(fileName,externalRules);
       // Print external rules
       for (int i = 0; i <
externalRules.getRuleTables().size(); i++)

       System.out.println(externalRules.getRuleTables().get(
i));

       // Create a test App with a test customer from
HelloData.xls
       App app = (App) engine.run("getDefaultApplication");
       // Run OpenRulesEngine
       engine.run("generateGreeting",app);
       System.out.println("\nGenerated Greeting: " +
app.getResult());
  }
}
```

Here we create an instance of OpenRulesEngine using the main Excel-file, HelloCustomer.xls, and external rules received from the DB. The main file HelloCustomer.xls defines the Environment as follows:

Environment	
import.java	hello.*
import.static	com.openrules.tools.Methods
include	../include/HelloTemplates.xls
include	../include/HelloData.xls

This application uses two simple Java beans:

```
Customer.java:
      String name;
      String maritalStatus;
      String gender;
      int age;
App.java:
      Customer customer;
      String greeting;
      String salutation;
      String result;
```

The proper instance of Customer and App are created based on the Excel file, HelloData.xls, using these data tables:

Data App apps			
customer.name	customer.maritalStatus	customer.gender	customer.age
Customer Name	**Marital Status**	**Gender**	**Age**
Robinson	Married	Female	24
Smith	Single	Male	19

Method App getDefaultApplication()
return apps[0];

And finally, the engine will execute rules by calling the method "run":

```
engine.run("generateGreeting",app);
```

The proper method, "generateGreeting", is described in the file, HelloCustomer.xl,s in the following table:

Method void generateGreeting(App app)

```
int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
greetingRules(app, hour);
salutationRules(app, app.customer);
app.result = app.greeting + ", " + app.salutation + " " +
app.customer.name + "!";
```

You may validate the entire rule project by double-clicking on the file "compile.bat". Because the actual external rule tables, "greetingRules" and "salutationRules", will become known only in run-time the proper OpenRules® validator may produce errors (warnings) about unknown rule tables. You may ignore these errors or you may explicitly inform OpenRules® about this fact by adding an optional table to the file HelloCustomer.xls:

ExternalRules	
greetingRules	defineGreeting
salutationRules	defineSalutation

To run the project you may double-click on the file "run.bat". Here is an expected output:

```
ExternalRules greetingRules template defineGreeting
0 11 Good Morning
12 15 Good Afternoon
16 21 Good Evening
22 24 Good Night

ExternalRules salutationRules template
defineSalutation
Male   Mr.
Male Single 3 Little
Female Single  Ms.
Female Married  Mrs.

Generated Greeting: Good Morning, Mrs. Robinson!
```

EXTERNAL RULES FROM XML

OpenRules® allows you to keep your business rules in XML files which correspond to the columns (conditions and actions) of Excel's templates based upon which the proper rule tables will be executed. While you may use any XML processing software, this sample project demonstrates how to use a simple XML interface provided by OpenRules®.

Step 1. Defining Rule Tables in XML

You may create a subdirectory "xml" in the directory "rules" and place different xml-files into it. The first file, "GreetingRules.xml", defines a rule table with the name "greetingRules" that will be based on the template with the name "defineGreeting":

```
<?xml version="1.0" encoding="UTF-8"?>

<GreetingRules ruleTableName="greetingRules"
templateName="defineGreeting" type="Array of Rule(s)">

  <Rule from="0" to="11" greeting="Good Morning" />
  <Rule from="12" to="16" greeting="Good Afternoon" />
  <Rule from="17" to="21" greeting="Good Evening" />
  <Rule from="22" to="24" greeting="Good Night" />
</GreetingRules>
```

Similarly, we create the second file, "SalutationRules.xml":

```
<?xml version="1.0" encoding="UTF-8"?>
<SalutationRules ruleTableName="salutationRules"
templateName="defineSalutation">
   <Rule
       gender="Female"
       maritalStatus="Married"
       salutation="Mrs."
   />
   <Rule
       gender="Female"
       maritalStatus="Single"
       salutation="Ms."
   />
```

```
    <Rule
       gender="Male"
       maritalStatus=""
       salutation="Mr."
    />
    <Rule
       gender="Male"
       maritalStatus="Single"
       maxAge="5"
       salutation="Little"
    />
</SalutationRules>
```

Please note that the last rule contains an extra attribute, "maxAge". OpenRules® does not require any specification of the XML document and will dynamically recognize its structure.

Step 2. Reading XML rules in Java

To inform an OpenRulesEngine about external rules, you need to create an object of the type ExternalRules and add to it all external RuleTables. Each instance of the class RuleTable consists of:

- rule table name (String)
- template name (String)
- a grid of objects that represent the contents of a rule table (Object[][])

In this project, we will create an instance of the class External Rules directly in the Excel method "getExternalRules":

Method ExternalRules **createExternalRules**()

ExternalRules externalRules = new ExternalRules();
addGreetingRules(externalRules);
addSalutationRules(externalRules);
return externalRules;

This method will execute two other methods, "addGreetingRules" and "addSalutationRules", that will read the above xml-files and will add the proper rule tables to the newly created ExternalRules object.

Before reading the xml files, we have to specify the proper xml schemas in the Environment table placed in the main Excel file HelloXMLRules.xls:

Environment	
import.java	com.openrules.table.external.Objects
import.schema	file:rules/xml/GreetingRules.xml
	file:rules/xml/SalutationRules.xml

OpenRules® dynamically defines the Java classes, GreetingRules and SalutationRules, that will be used to read the proper XML files.

Now we may specify the method "addGreetingRules":

```
Method ExternalRules addGreetingRules(ExternalRules externalRules)
GreetingRules greetings =
GreetingRules.load("file:rules/xml/GreetingRules.xml");
Objects[] grid = new Objects[greetings.Rule.length];
for(int i = 0; i < greetings.Rule.length; ++i) {
   GreetingRules.Rule r = greetings.Rule[i];
   Objects row = new Objects(3);
   row.set(0,r.from); row.set(1,r.to); row.set(2,r.greeting);
   grid[i] = row;
}
externalRules.addRuleTable(greetings.ruleTableName,
greetings.templateName, grid );
```

First we load the rules from the xml-file defining its relative path using the standard OpenRules® URL notation:

```
file:rules/xml/GreetingRules.xml
```

All objects specified in the file GreetingRules.xml becomes available to the Java code through the object "greetings" of the dynamically defined type GreetingRules. In particular, the object "greetings.Rule" points to the array of objects of the dynamic type "Rule" as it was defined in the xml-file.

Next, we create a "grid" as an array of the predefined type Objects, which is used by OpenRules® to simplify the handling of the multi-dimensional array. Looping through all elements of the array greetings.Rules, we add new rows to the object "grid". Data elements inside each rule are available through their names as defined in the xml-file: `r.from`, `r.to`, and `r.greeting`.

Similarly, we specify the method "addSalutationRules":

```
Method ExternalRules addSalutationRules(ExternalRules externalRules)
SalutationRules salutations =
SalutationRules.load("file:rules/xml/SalutationRules.xml");
Objects[] grid = new Objects[salutations.Rule.length];
for(int i = 0; i < salutations.Rule.length; ++i) {
   SalutationRules.Rule r = salutations.Rule[i];
   Objects row = new Objects(4);
   row.set(0,r.gender); row.set(1,r.maritalStatus); row.set(2,r.maxAge);
row.set(3,r.salutation);
   grid[i] = row;
}
externalRules.addRuleTable(salutations.ruleTableName,
salutations.templateName, grid );
```

Step 3. Creating and Executing Rules from a Java Program

All other modules are typical for basic rule projects. The main Java file, RunExternalRulesFromXML.java, is used to test the above rules:

```
import com.openrules.ruleengine.ExternalRules;
import com.openrules.ruleengine.OpenRulesEngine;

public class RunExternalRulesFromXML {

    public static void main(String[] args){
      // The first engine reads XML-based rules described at HelloXMLRules.xls
      String xlsMainData =
"file:rules/main/HelloXMLRules.xls";
      OpenRulesEngine engine1 = new
OpenRulesEngine(xlsMainData);
      ExternalRules externalRules =
            (ExternalRules)
engine1.run("createExternalRules");
      // Print External Rules
```

```
      for (int i = 0; i <
externalRules.getRuleTables().size(); i++)
System.out.println(externalRules.getRuleTables().get(i));
       // The second engine reads test data and execute external rules
      // created by the first engine
      String fileName =
"file:rules/main/HelloCustomer.xls";
      OpenRulesEngine engine2 =
                  new
OpenRulesEngine(fileName,externalRules);
      App app = (App)
engine2.run("getDefaultApplication");
      engine2.run("generateGreeting",app);
      System.out.println("\nGenerated Greeting:");
      System.out.println(app.getResult());

    }
}
```

The first instance, “engine1", of the class OpenRulesEngine is based on the main Excel-file, HelloXMLRules.xls. We execute the method, "createExternalRules", to create external rules from the xml files. The second instance "engine2" of the OpenRulesEngine uses the main Excel-file, HelloCustomer.xls, and the newly created external rules.

The main file, HelloCustomer.xls, defines the Environment as follows:

Environment	
import.java	hello.*
import.static	com.openrules.tools.Methods
include	../include/HelloTemplates.xls
include	../include/HelloData.xls

This application uses two simple Java beans:

Customer.java:

```
      String name;
      String maritalStatus;
      String gender;
      int age;
```

App.java:

```
Customer customer;
String greeting;
String salutation;
String result;
```

The proper instance of Customer and App are created based on the Excel file, HelloData.xls, using these data tables:

Data App apps			
customer.name	customer.maritalStatus	customer.gender	customer.age
Customer Name	**Marital Status**	**Gender**	**Age**
Robinson	Married	Female	24
Smith	Single	Male	19

Method App getDefaultApplication()
return apps[0];

And finally, `engine2` will execute the rules by calling the method "run":

```
engine2.run("generateGreeting",app);
```

The proper method, "generateGreeting", is described in the file, HelloCustomer.xls. in the following table:

Method void generateGreeting(App app)

int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
greetingRules(app, hour);
salutationRules(app, app.customer);
app.result = app.greeting + ", " + app.salutation + " " +
app.customer.name + "!";

You may validate the entire rule project by double-clicking on the file "compile.bat". Because the actual external rule tables, "greetingRules" and "salutationRules", will become known only at run-time the proper OpenRules® validator may produce errors (warnings) about unknown rule tables. You may

ignore these errors or you may explicitly inform OpenRules® about this fact by adding an optional table to the file, HelloCustomer.xls:

ExternalRules	
greetingRules	defineGreeting
salutationRules	defineSalutation

To run the project you may double-click on the file "run.bat". Here is an expected output:

```
INITIALIZE OPENRULES ENGINE 5.3.0 (build 03092009) for
[file:rules/main/HelloXMLRules.xls]
IMPORT.JAVA=com.openrules.table.external.Objects
IMPORT.SCHEMA=file:rules/xml/GreetingRules.xml
IMPORT.SCHEMA=file:rules/xml/SalutationRules.xml
ExternalRules greetingRules template defineGreeting
0 11 Good Morning
12 16 Good Afternoon
17 21 Good Evening
22 24 Good Night

ExternalRules salutationRules template defineSalutation
Female Married null Mrs.
Female Single null Ms.
Male  null Mr.
Male Single 5 Little

INITIALIZE OPENRULES ENGINE 5.3.0 (build 03092009) for
[file:rules/main/HelloCustomer.xls]
External rules table: greetingRules
External rules table: salutationRules
IMPORT.JAVA=hello.*
IMPORT.JAVA=com.openrules.tools.Operator
IMPORT.STATIC=com.openrules.tools.Methods
INCLUDE=../include/HelloTemplates.xls
[../include/HelloTemplates.xls] has been resolved to
[file:<..>/ExternalRulesFromXML/rules/include/HelloTempla
tes.xls]
INCLUDE=../include/HelloData.xls
[../include/HelloData.xls] has been resolved to
[file:<..>/ExternalRulesFromXML/rules/include/HelloData.x
ls]

Generated Greeting:
Good Afternoon, Mrs. Robinson!
```

EXTERNAL RULES FROM EXCEL

OpenRules® allows you to keep your business rules in Excel data tables that correspond to the columns (conditions and actions) of Excel's templates based upon which the proper rule tables will be executed.

Step 1. Defining Rule Tables in Excel Data Tables

We will create the main xls-file HelloRules.xls in the subdirectory "rules/mainl". The first Data table defines "greetingRules" which will be based on the template with the name "defineGreeting":

Data GreetingRule greetingRules		
from	to	greeting
From	**To**	**Greeting**
0	11	Good Morning
12	17	Good Afternoon
18	22	Good Evening
23	24	Good Night

To access this table from java we define the following method:

Method GreetingRule[] getDefaultGreetingRules()
return greetingRules;

This data table uses the datatype, GreetingRules, which is specified in the proper Java class:

```
public class GreetingRule {
  int from;
  int to;
  String greeting;

  public int getFrom() {
        return from;
```

```
    }
    public void setFrom(int from) {
          this.from = from;
    }
    public int getTo() {
          return to;
    }
    public void setTo(int to) {
          this.to = to;
    }
    public String getGreeting() {
          return greeting;
    }
    public void setGreeting(String greeting) {
          this.greeting = greeting;
    }
  }
```

Similarly, we create the second Data table "salutationRules":

Data SalutationRule salutationRules			
gender	maritalStatus	maxAge	salutation
Gender	**Marital Status**	**Age Less Than**	**Set Salutation**
Female	Married		Mrs.
Female	Single		Ms.
Male			Mr.
Male	Single	10	Little

and the proper method:

Method SalutationRule[] getDefaultSalutationRules()
return salutationRules;

This data table uses the datatype, SalutationRules, which is specified in the proper Java class:

```
public class SalutationRule {

    String       gender;
    String  maritalStatus;
    String       maxAge;
    String salutation;
```

```
    public String getGender() {
          return gender;
    }
    public void setGender(String gender) {
          this.gender = gender;
    }

    public String getMaritalStatus() {
          return maritalStatus;
    }
    public void setMaritalStatus(String maritalStatus) {
          this.maritalStatus = maritalStatus;
    }

    public String getMaxAge() {
          return maxAge;
    }
    public void setMaxAge(String maxAge) {
          this.maxAge = maxAge;
    }

    public String getSalutation() {
          return salutation;
    }
    public void setSalutation(String salutation) {
          this.salutation = salutation;
    }

}
```

Step 2. Creating and Executing External Rules from a Java Program

All other modules are typical for basic rule projects. The main Java file RunExternalRulesFromXML.java is used to test the above rules:

```
import com.openrules.ruleengine.ExternalRules;
import com.openrules.ruleengine.OpenRulesEngine;

public class RunExternalRulesFromExcel {

  public static void main(String[] args)  {
    // The first engine
    String xlsMainRules =
"file:rules/main/HelloRules.xls";
```

```
    OpenRulesEngine engine1 = new
OpenRulesEngine(xlsMainRules);
    GreetingRule[] greetingRules =

(GreetingRule[])engine1.run("getDefaultGreetingRules");
    String[][] greetingGrid = new
String[greetingRules.length][3];
    for (int i = 0; i < greetingRules.length; i++) {
      GreetingRule rule = greetingRules[i];
      greetingGrid[i] = new String[] {
         Integer.toString(rule.from),
         Integer.toString(rule.to),
         rule.greeting
       };
    }

    SalutationRule[] salutationRules =

(SalutationRule[])engine1.run("getDefaultSalutationRules"
);
    String[][] salutationGrid =
                      new
String[salutationRules.length][4];
    for (int i = 0; i < salutationRules.length; i++) {
      SalutationRule rule = salutationRules[i];
      salutationGrid[i] = new String[] {
           rule.gender,
           rule.maritalStatus,
           rule.maxAge,
           rule.salutation
       };
    }
    // create external rules
    ExternalRules externalRules = new ExternalRules();
    externalRules.addRuleTable(
          "greetingRules",   //table name
          "defineGreeting",  //template name
          greetingGrid
    );
    externalRules.addRuleTable(
          "salutationRules", //table name
          "defineSalutation", //template name
          salutationGrid
    );
    // Display external rules
    for (int i = 0; i <
externalRules.getRuleTables().size(); i++)
```

```
System.out.println(externalRules.getRuleTables().get(i));

    // The second engine
    String fileName =
"file:rules/main/HelloCustomer.xls";
    OpenRulesEngine engine2 =
                    new
OpenRulesEngine(fileName,externalRules);
    App app = (App) engine2.run("getDefaultApplication");
    engine2.run("generateGreeting",app);
    System.out.println("\nGenerated Greeting:");
    System.out.println(app.getResult());

  }
}
```

The first instance, “engine1", of the class OpenRulesEngine, is based on the main Excel-file, HelloRules.xls. We create the array, greetingRules, by executing the method, "createExternalRules", to generate external rules from the xml files:

```
GreetingRule[] greetingRules =

      (GreetingRule[])engine1.run("getDefaultGreetingRules");
```

Then we convert this array into a simple "greetingGrid" of the type `String[][]`. Similarly, we create the grid, "salutationRules".

Next, we create an instance of ExternalRules and add two rule tables into it:

```
ExternalRules externalRules = new ExternalRules();
externalRules.addRuleTable(
   "greetingRules", //table name
   "defineGreeting", //template name
   greetingGrid
);
externalRules.addRuleTable(
   "salutationRules", //table name
   "defineSalutation", //template name
   salutationGrid
);
```

The second instance, "engine2," of the OpenRulesEngine uses the main Excel-file, HelloCustomer.xl,s and the newly created external rules:

```
OpenRulesEngine engine2 =
```

```
new OpenRulesEngine(fileName,externalRules);
```

The main file, HelloCustomer.xls, defines the Environment as follows:

Environment	
import.java	hello.*
import.static	com.openrules.tools.Methods
include	../include/HelloTemplates.xls
include	../include/HelloData.xls

This application uses two simple Java beans:

Customer.java:

```
String name;

String maritalStatus;

String gender;

int age;
```

App.java:

```
Customer customer;

String greeting;

String salutation;

String result;
```

The proper instances of Customer and App are created based on the Excel file, HelloData.xls, using these data tables:

Data App apps			
customer.name	customer.maritalStatus	customer.gender	customer.age
Customer Name	**Marital Status**	**Gender**	**Age**
Robinson	Married	Female	24
Smith	Single	Male	19

Method App getDefaultApplication()

```
return apps[0];
```

And finally, `engine2` will execute rules by calling the method "run":

```
engine2.run("generateGreeting",app);
```

The proper method, "generateGreeting", is described in the file, HelloCustomer.xls, in the following table:

Method void generateGreeting(App app)

int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
greetingRules(app, hour);
salutationRules(app, app.customer);
app.result = app.greeting + ", " + app.salutation + " " +
app.customer.name + "!";

You may validate the entire rule project by double-clicking on the file "compile.bat". Because the actual external rule tables, "greetingRules" and "salutationRules", will become known only at run-time the proper OpenRules® Validator may produce errors (warnings) about unknown rule tables. You may ignore these errors or you may explicitly inform OpenRules® about this fact by adding an optional table to the file HelloCustomer.xls:

ExternalRules	
greetingRules	defineGreeting
salutationRules	defineSalutation

To run the project you may double-click on the file "run.bat". Here is an expected output:

```
INITIALIZE OPENRULES ENGINE 5.3.0 (build 03092009) for
[file:rules/main/HelloRules.xls]
IMPORT.JAVA=hello.*
ExternalRules greetingRules template defineGreeting
0 11 Good Morning
12 17 Good Afternoon
18 22 Good Evening
23 24 Good Night
```

```
ExternalRules salutationRules template defineSalutation
Female Married null Mrs.
Female Single null Ms.
Male null null Mr.
Male Single 10 Little

INITIALIZE OPENRULES ENGINE 5.3.0 (build 03092009) for
[file:rules/main/HelloCustomer.xls]
External rules table: greetingRules
External rules table: salutationRules
IMPORT.JAVA=hello.*
IMPORT.JAVA=com.openrules.tools.Operator
IMPORT.STATIC=com.openrules.tools.Methods
INCLUDE=../include/HelloTemplates.xls
[../include/HelloTemplates.xls] has been resolved to
[file:<..>/ExternalRulesFromExcel/rules/include/HelloTemp
lates.xls]
INCLUDE=../include/HelloData.xls
[../include/HelloData.xls] has been resolved to
[file:<..>/ExternalRulesFromExcel/rules/include/HelloData
.xls]

Generated Greeting:
Good Afternoon, Mrs. Robinson!
```

EXTERNAL RULES FROM GUI

OpenRules® allows you to keep your business rules in Excel data tables that correspond to the columns (conditions and actions) of Excel's templates based on which the proper rule tables will be executed.

Step 1. Defining A Graphical User Interface

This project illustrates how to create a web application that will consist of two parts:

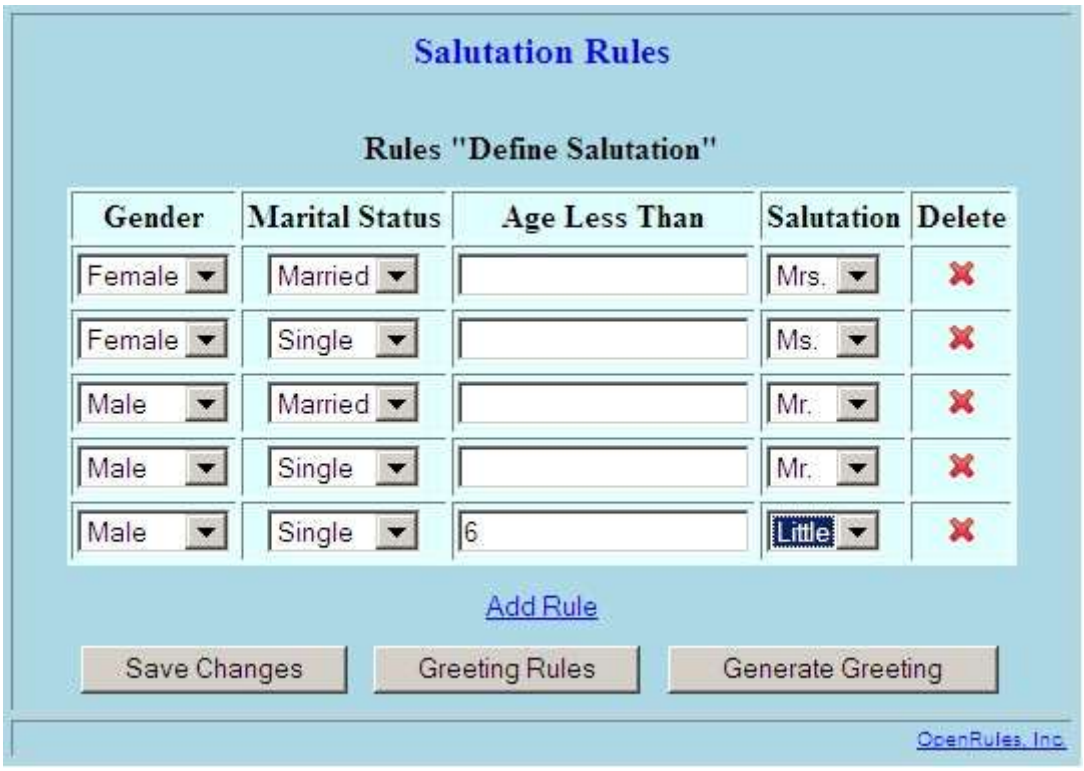
1) Data input and Rule Engine Execution
2) Online Rules Editing

The view "Generate Customer Greeting" will allow a user to enter basic information about a customer and will generate a greeting like "Good Morning,

Mrs. Robinson!" based on the current time. Here is an example of the proper view:

By clicking on the button "Generate Greeting" a user could produce a new greeting in accordance with the latest greeting and salutation rules. By clicking on the button, "Greeting Rules", a user will be taken to a web-based rule editor to modify the Greeting Rules:

By clicking on the button, "Salutation Rules", a user will be taken to a web-based rule editor to modify the Salutation Rules:



This editor shows how to make changes in the rule attributes; it also allows a user to add rules by clicking on the hyperlink "Add Rule", or to delete rules by clicking on the red cross.

Step 2. Defining Implementation Approach

We will build this web application using OpenRules® Forms by defining 3 Excel-based layouts for each of the above views and using navigation logic described as processing flow rules. We will deploy our application on the Tomcat server. As usual, we will create the following files:

File	Directory	Purpose
HelloExternalRulesFromGUI.xls	*./war/rules/main*	Describes the Environment table and the main method that will be executed during every interaction with a web client
HelloForms.xls	*./war/rules/gui*	Describes all screen layouts and processing flow rules
Dialog.xls	*./war/rules/gui*	The standard OpenRules® file borrowed from the project openrules.forms.lib
HelloData.xls	*./war/rules/data*	Rule templates
index.jsp	*./*	The entry point to this JSP-based web application

What makes this application special is the need to reinitialize the rule engine that generates a greeting each time the greetings and/or salutations have been modified. However, it is not necessary to reinitialize a rule engine associated with an already opened `OpenRulesSession` with all layouts and related rule tables. So, we need to carefully distributes greeting generation information and GUI information between two different rule engines while making sure that re-initialization of the first engine is done very quickly.

When we start an application for the first time, we want to display the default rules (defined in an Excel file) and we also want to use the default data about a customer (defined in another Excel file).

In this implementation, we will define a special Java class `HelloManager` whose responsibilities will include these and other data management tasks. The manager will support two rule engines:

1. A rule engine that reads the default greeting and salutation rules from the file, `war/rules/main/HelloDefaultRules.xls`. Only this engine will deal with greeting rules and rule templates presented in the file, `war/rules/logic/HelloTemplates.xls`.

2. A rule engine associated with the `OpenRulesSession` that will handle all GUI problems and will also read the default data about a customer from the Excel file, "HelloData.xls".

Thus, the entry point to our web application "`index.jsp`" will look as follows:

```
<%@ page import="com.openrules.forms.gui.jsp.*" %>
<%@ page import="com.openrules.forms.*" %>
<%@ page import="hello.rules.*" %>
<%@ page import="com.openrules.ruleengine.*" %>

<%
String s_attr = "openrules_session";
OpenRulesSession openrules_session = (OpenRulesSession)
session.getAttribute(s_attr);
if (openrules_session == null ) {
    // Create manager using data from HelloDefaultRules.xls
    String xlsMainRules =
"file:./webapps/HelloExternaRulesFromGUI/rules/main/HelloDefaultRule
s.xls";
    HelloManager man = new HelloManager(xlsMainRules);
    // Create OpenRulesSession using HelloExternaRulesFromGUI.xls
    String xlsMainForms =
"file:./webapps/HelloExternaRulesFromGUI/rules/main/HelloExternaRule
sFromGUI.xls";
    openrules_session = new OpenRulesSession(xlsMainForms);
    session.setAttribute( s_attr, openrules_session);
    System.out.println("NEW SESSION based on " + xlsMainForms);
```

```
    man.setFormsEngine(openrules_session.getOpenRulesEngine());
    // Read default rules and data from Excel files
    man.getDefaults();
    Dialog dialog = openrules_session.getDialog();
    dialog.put("manager",man);
}
%>
<HTML><HEAD><TITLE>OpenRules</TITLE></HEAD>
<body>
<%
     System.out.println("PROCESS REQUEST");
     openrules_session.processRequest(session, request, out);
%>
</body>

</HTML>
```

The first rule engine will be created by the constructor HelloManager(xlsMainRules). The second rule engine, automatically created by the `OpenRulesSession`, will be set for HelloManager by the statement:

```
man.setFormsEngine(openrules_session.getOpenRulesEngine());
```

The Environment table for the first rule engine is located in the file HelloDefaultRules.xls:

Environment	
import.java	hello.rules.*
include	../logic/HelloTemplates.xls

The Environment table for the second rule engine is located in the file HelloExternaRulesFromGUI.xls:

Environment	
import.static	com.openrules.tools.Methods
import.java	hello.rules.*
include	../gui/Dialog.xls
	../data/HelloData.xls

	../gui/HelloForms.xls

The main execution loop is implemented by the following method:

```
Method TableLayout main(Dialog dialog)
HelloManager man = (HelloManager) dialog().get("manager");
if (man == null)
   return fatalErrorLayout("HelloManager is not defined if
index.jsp");
defineNextProcessingStep(man);
if (dialog().errors == 0)
{
    processingFlowRules(man);
    defineNextProcessingStep(man);
}
return mainLayout();
```

Step 3. Creating Supporting Java Classes

We define a Java package,, "hello.rules" with the following classes:

Customer (Customer.java)

- String name
- String gender
- String maritalStatus
- int age

App (App.java)

- Customer customer
- String greeting
- String salutation

GreetingRule (GreetingRule.java)

- int from
- int to
- String greeting

These classes are basic Java beans used inside rules and forms. To demonstrate the use of a more complex rule editor, we will implement the rule table for salutation rules as an OpenRules® dynamic table. To do this, we will define two classes:

SalutationRule implements Checkable (SalutationRule.java)

- String gender
- String maritalStatus
- String maxAge
- String salutation
- HelloManager manager

and the class, **SalutationRules,** that extends `DynamicTable` (see SalutationRules.java) by defining two methods:

```
public String getHeaderLayoutName() {
        return "salutationsTableHeader";
}
public String getRowLayoutName() {
        return "salutationsTableRow";
}
```

The main Java class is a placeholder for all other objects:

HelloManager (HelloManager.java)

- *OpenRulesEngine ruleEngine*
- *OpenRulesEngine formsEngine*
- *GreetingRule[] greetingRules*
- *SalutationRule[] defaultSalutationRules*
- *SalutationRules salutationRules*
- *App app*
- *ExternalRules externalRules*

The object, "ruleEngine", is defined in the constructor for the object, "formsEngine", defined in the index.jsp. When the application is initialized the manager executes the method "getDefaults":

```
public void getDefaults() {

    greetingRules =

(GreetingRule[])ruleEngine.run("getDefaultGreetingRules")
;

    defaultSalutationRules =

(SalutationRule[])ruleEngine.run("getDefaultSalutationRul
es");
    salutationRules = new SalutationRules(formsEngine);
    for (int i = 0; i < defaultSalutationRules.length;
i++) {
          SalutationRule rule =
defaultSalutationRules[i];
          rule.setManager(this);
          salutationRules.addNewRow(rule);
    }

    createExternalRules();
    externalRules.setModified(true);
    ruleEngine.log("There is " +

getExternalRules().getRuleTables().size()
                    + " external tables");

    Customer customer =
             (Customer)
formsEngine.run("getDefaultCustomer");
    app = new App();
    app.setCustomer(customer);
}
```

This method receives the *greetingRules* from the file *HelloDefaultRules.xls* using the method "getDefaultGreetingRules". It receives the *defaultSalutationRules* using the method "getDefaultSalutationRules" and then creates *salutationRules* to support the proper dynamic graphical table. It then creates an instance of the type ExternalRules, using this method:

```
public void createExternalRules() {

    String[][] greetingGrid = new
String[greetingRules.length][3];
    for (int i = 0; i < greetingRules.length; i++) {
          GreetingRule rule = greetingRules[i];
          greetingGrid[i] = new String[] {
                    Integer.toString(rule.from),
                    Integer.toString(rule.to),
                    rule.greeting
            };
    }

    String[][] salutationGrid =
                      new
String[salutationRules.getRows().size()][4];
    for (int i = 0; i < salutationRules.getRows().size();
i++) {
      SalutationRule rule =

(SalutationRule)salutationRules.getRows().get(i);
      salutationGrid[i] = new String[] {
                rule.gender,
                rule.maritalStatus,
                rule.maxAge,
                rule.salutation
       };
    }

    externalRules = new ExternalRules();
    externalRules.addRuleTable(
          "greetingRules",    //table name
          "defineGreeting",   //template name
          greetingGrid);

    externalRules.addRuleTable(
          "salutationRules", //table name
          "defineSalutation", //template name
          salutationGrid);

    externalRules.setModified(false);
    ruleEngine.setExternalRules(externalRules);

}
```

And finally, the manager creates the default application, "app", with a customer received from the file *HelloData.xls*:

Data Customer customers			
customer.name	customer.maritalStatus	customer.gender	customer.age
Customer Name	**Marital Status**	**Gender**	**Age**
Robinson	Married	Female	24
Smith	Single	Male	19

Method Customer getCustomer()
return customers[0];

Step 4. Creating Graphical Layouts in Excel

All GUI realted forms and rules are described in the file HelloForms.xls. The "mainLayout" specifies a general layout for all three layouts:

Layout TableLayout mainLayout()		
properties	width	100%
	cellspacing	4
	cellpadding	2
	border	1
	style	background-color:lightblue
dialog().nextLayout		
  OpenRules, Inc. 		

The layout "GenerateGreeting":

Layout TableLayout generateGreetingLayout(App app, Customer c)	
Generate Customer Greeting	
currentTime()	
"Name:"	[c.name]
"Age:"	[c.age]

"Gender:"	[c.gender]["Male,Female"]	
"Marital Status:"	<F type="radio" >[c.maritalStatus]["Single,Married"] </F>	
actionButton("Generate Greeting");	actionButton("Greeting Rules");	actionButton("Salutation Rules");
"Generated Greeting:"	<C> app.result </C>	

There are two layouts to support "GreetingRules":

Layout TableLayout greetingRulesLayout(HelloManager man)		
Greeting Rules		
Rules "Define Greeting"		
greetingRulesTable(man);		
actionButton("Save Changes");	actionButton("Salutation Rules");	actionButton("Generate Greeting");

Layout TableLayout greetingRulesTable(HelloManager man)		
Hour From	Hour To	Greeting
[man.greetingRules[0].from]	[man.greetingRules[0].to]	[man.greetingRules[0].greeting][getPossibleGreetings()]
[man.greetingRules[1].from]	[man.greetingRules[1].to]	[man.greetingRules[1].greeting][getPossibleGreetings()]
[man.greetingRules[2].from]	[man.greetingRules[2].to]	[man.greetingRules[2].greeting][getPossibleGreetings()]
[man.greetingRules[3].from]	[man.greetingRules[3].to]	[man.greetingRules[3].greeting][getPossibleGreetings()]

This form has a fixed number of rules (rows), so a user may only change the values of rules attributes. The layout, "SalutationRules", represents a dynamic table:

Layout TableLayout salutationRulesLayout(HelloManager man)		
Salutation Rules		
Rules "Define Salutation"		
man.salutationRules.createTable();		
actionHyperlink("Add Rule");		
actionButton("Save Changes");	actionButton("Greeting Rules");	actionButton("Generate Greeting");

Layout TableLayout salutationsTableHeader()				
Gender	Marital Status	Age Less Than	Salutation	Delete

Layout TableLayout salutationsTableRow(SalutationRule rule)				
[rule.gender] ["Male,Female"]	[rule.maritalStatus] ["Married,Single"]	[rule.maxAge]	[rule.salutation] [getPossibleSalutations()]	deleteRuleButton(rule);

Layout TableLayout deleteRuleButton(SalutationRule rule)
<F type="image" src="../openrules.forms.lib/images/delete.png"> [][] [rule.manager.salutationRules.deleteRow(rule); dialog().setLastAction("Delete Rule")] </F>

Here is the rule table that specifies processing flow:

Rules void processingFlowRules(HelloManager man)			
IF Current Step is	**AND Action is**	**THEN Execute Code**	**AND Go To The Step**
	Init		GenerateGreeting
GenerateGreeting		{ man.cleanUp(); }	GenerateGreeting
GenerateGreeting	Generate Greeting	{ man.generateGreeting(); }	GenerateGreeting
GenerateGreeting	Greeting Rules		GreetingRules
GenerateGreeting	Salutation Rules		SalutationRules
GreetingRules	Save Changes	{ man.updateRules(); }	GreetingRules
GreetingRules	Salutation Rules		SalutationRules
GreetingRules	Generate Greeting		GenerateGreeting
SalutationRules	Save Changes	{ man.updateRules(); }	SalutationRules
SalutationRules	Add Rule	{ man.addSalutationRule(); }	SalutationRules
SalutationRules	Delete Rule		SalutationRules
SalutationRules	Greeting Rules		GreetingRules
SalutationRules	Generate Greeting		GenerateGreeting

As you can see, the action "Save Changes" leads to the execution of the manager's method "updateRules":

```
public void updateRules() {
        createExternalRules();
        getExternalRules().setModified(true);
        showRules();

}
```

This method will create a new instance of the external rules, (based on the latest changes introduced by the rule editor), and it will mark the external rules as "modified", which will force a rule engine to reinitialize itself before the next run of the method "generateGreeting":

```
public void generateGreeting() {
          ruleEngine.run("greetingRules",app);
          ruleEngine.run("salutationRules",app);

}
```

Step 5. Deploying and Executing the Web Application

To deploy this web application on the Tomcat server specified in the file *build.properties*, it is enough to double-click on *deploy.bat*. To start the application, make sure that your Tomcat is up and running and double-click on *run.bat*.

EXTERNAL RULES FOR DECISIONS

When you want to use ExternalRules with decisions you rely on the rule templates predefined in the file "DecisionTableExecuteTemplates.xls" (inside the standard folder "openrules.congig"). While the names of the templates are predefined ("`DecisionTableTemplate`", "`DecisionTable1Template`", or "`DecisionTable2Template`") you need an ability to also specify:

- Labels for all selected columns (like `"If"` or `"Then")`

- Names of decision variables used by these columnsallows for rule tables defined as Java objects.

To do that, you may use an extended API for the External Rules similar to the one used in the following example:

```
ExternalRules externalRules = new ExternalRules();
externalRules.addRuleTable(
     "DefineGreeting",    //table name
     "DecisionTable1Template", //template name
     new String[] { "If", "If", "Then" }, // labels
     new String[] { "Current Hour", "Current Hour", "Result" },
                                          // descriptions/variables
     new String[][] {     //rules
         new String[] {">=0","<=11","Good Morning"},
         new String[] {">=12","<=17","Good Afternoon"},
         new String[] {">=18","<=21","Good Evening"},
         new String[] {">=22","<=24","Good Night"}
     }
);
```

This way you may dynamically create decision table from your database or other sources. A complete example that demonstrates this interface is provided in the standard project "DecisionExternalRules".

TECHNICAL SUPPORT

Direct all your technical questions to support@openrules.com or to this Discussion Group.