Using dynamic domains for flexible rule authoring in WebSphere Operational Decision Management

Ensuring rule accuracy is an important part of rule development in WebSphere Operational Decision Management 7.5. This tutorial demonstrates how a dynamic domain populated from an Excel file can be used to reduce inaccuracies when authoring rules. Using domains also eliminates dependencies on IT support for code changes because business users can extend the set of values in the domain by simple edits to the spreadsheet in the Decision Center. Further, because adding a domain value does not result in changes to service contracts, redeployment of rule services is not necessary.

Introduction

This article demonstrates how to create and modify a dynamic domain when working with business rules in WebSphere Operational Decision Management V7.5. An example rule is defined to expect a value of type string. The rule is then converted so that the expected values are constrained by an enumerated domain. The domain is dynamically populated by an Excel file. This article provides steps for converting and testing the rule in Rule Designer, and shows how the domain can be modified or extended by a business user in the Decision Center.

This article assumes the following prerequisites:

- Familiarity with rule design and development in WebSphere Operational Decision Management V7.5
- Practical experience with Rule Designer
- A successful installation of IBM WebSphere Operational Decision management V7.5
- An installed copy of Microsoft Excel 2003 or 2007

What are dynamic domains?

WebSphere Operational Decision Management V7.5 supports the use of a variety of domains for rule development. A domain places a restriction on type elements in the Business Object Model (BOM). WebSphere Operational Decision Management supports static, dynamic, enumerated, and complex domains. For example, enumerated domains can be used in rule editing to present valid choices to the rule author. A dynamic domain is an enumerated domain that makes use of values in an Excel file or other data source. The domain is dynamically populated from the data source. This provides the flexibility to make on-the-fly changes and then synchronize the domain and the data source.

Why use dynamic domains?

Dynamic domains offer a number of advantages to both rule developers and business users:

- By constraining a business user's selection to a predefined set of values, rule developers can help prevent errors.
- Dynamic domains can be easily extended or modified by business users, whereas static enumerations require IT support for changes.
- Adding a value to a dynamic domain does not require a change to a rule service contract.

Preventing errors. An important best practice in rule design is creating rules that ensure accuracy and consistency when they are used. Enforcing consistency upfront can prevent downstream validation errors during rule execution. For example, suppose that one

clause of a rule expects the name of a language, specified as a string. The rule might state that if the language of a request is English, the response is to be transmitted in that language. Other language strings expected might be “German” or “Italian”, but unless the string is entered accurately the rule is invalid.

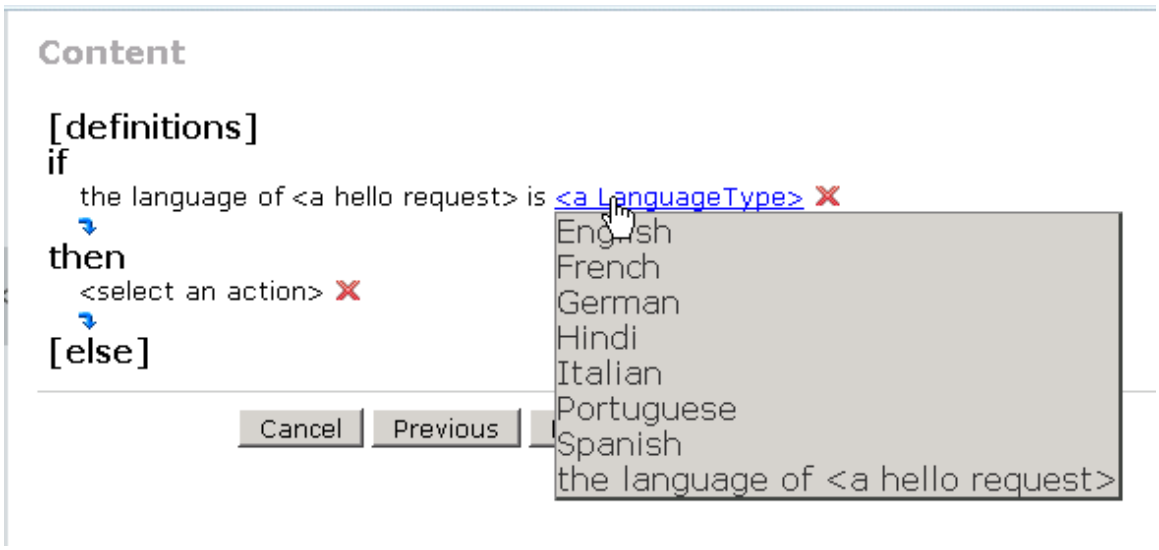
Figure 1. Specifying a string when defining a rule

During rule authoring, business users can accidentally misspell the string or use a mix of upper and lower cases. An exact match on the string is necessary for successful rule execution; user errors will result in the rule misfiring or not firing at all.

To tighten the constraints on a rule so that errors are less likely, a string can be replaced with an enumerated list. This improves rule accuracy but extensions to the list require additional coding. This can cause a delay while business users wait for the change to be implemented, tested, and released for use.

The recommended best practice is to represent the expected values in a dynamic domain. A dynamic domain is a predefined set of values that can be easily extended by business users without the need for an additional cycle of development and test time. In this example, a domain is used to constrain the choices to a predefined list of languages for selection.

Figure 2. Selecting a language from a predefined list



Freeing business users to extend valid parameters for rules. If an Excel file is the underlying data source of a domain, business users authoring rules in the Decision Center can easily add or remove values. A change can be introduced by making a simple edit to the spreadsheet and then updating the corresponding decision table. This flexibility removes dependencies on developers to make required code changes and can result in reduced time for changes to be deployed.

Eliminating the need for changes to service contracts. An important benefit of using domains instead of static enumerations, apart from reduced dependency on developers, is that adding a domain value does not result in changes to service contracts. Therefore developers do not need to redeploy the rule service or change the service client(s). Conversely, when using dynamic domains the service contract is less self-descriptive as it does not describe the possible domain values. The values would need to be communicated to the service client outside of the contract.

Part 1. Creating a dynamic domain

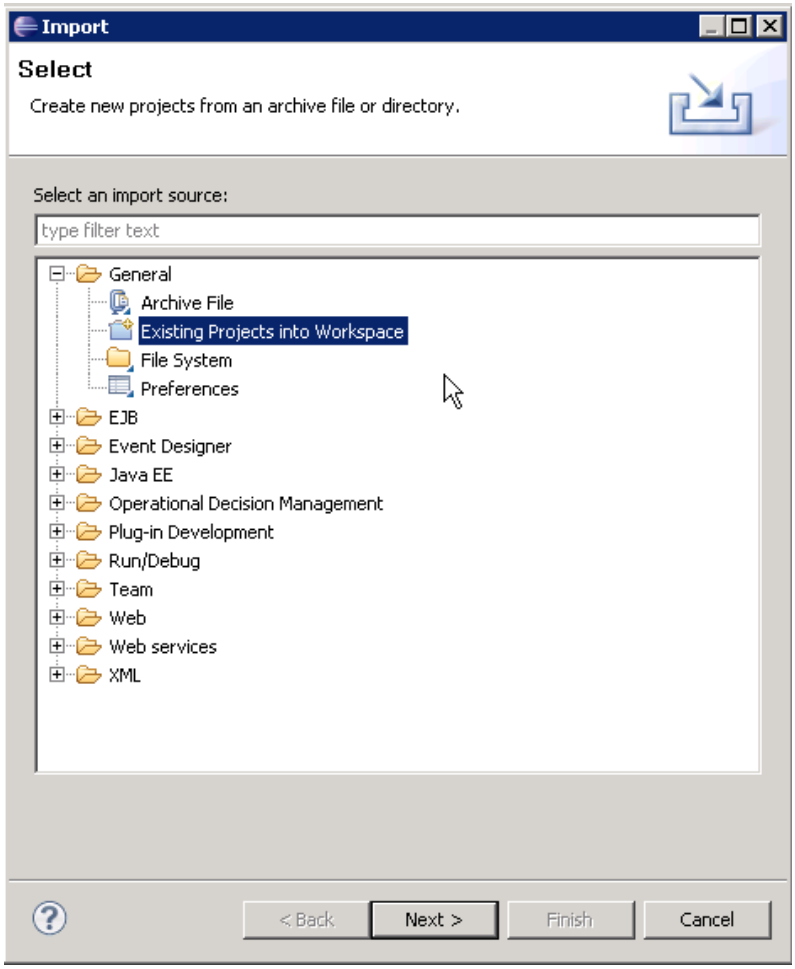
This section introduces the example rule project and describes the steps needed to convert a rule so that the expected parameter uses the enumerated type LanguageType instead of the type string.

Step 1: Import package and review rules

1. Open the Rule Designer. From the **File** menu, select **Import**…

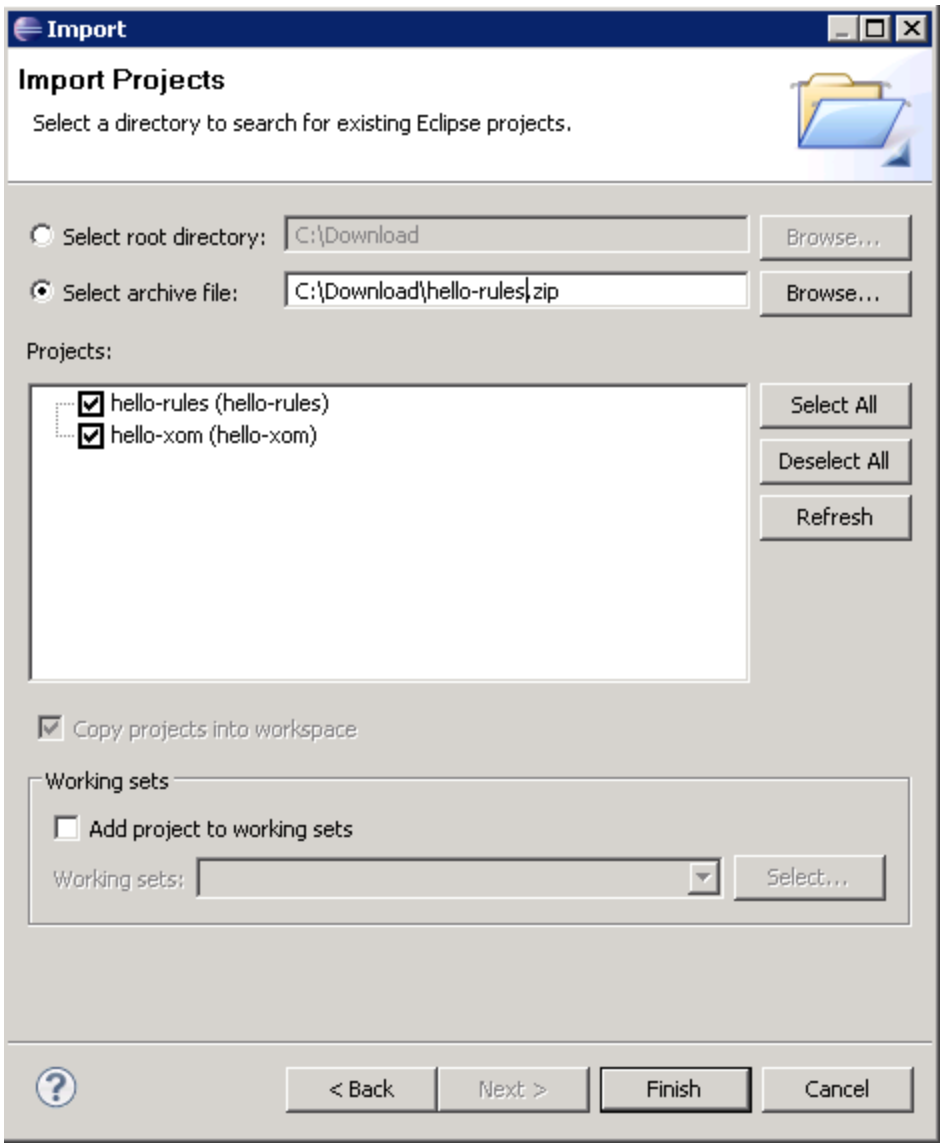
2. Select **Existing Projects into Workspace** and Click Next.

Figure 3. Selecting an import source



3. On the Import Projects dialog, select “Select archive file”. Browse to the location of the downloaded zip package. Click **Finish**.

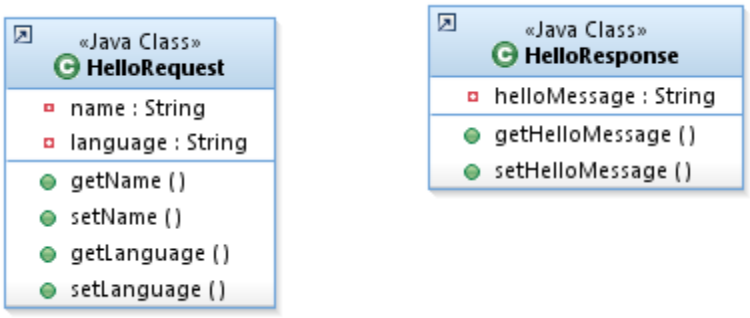
Figure 4. Selecting the archive file to import



The HelloRequest and HelloResponse classes

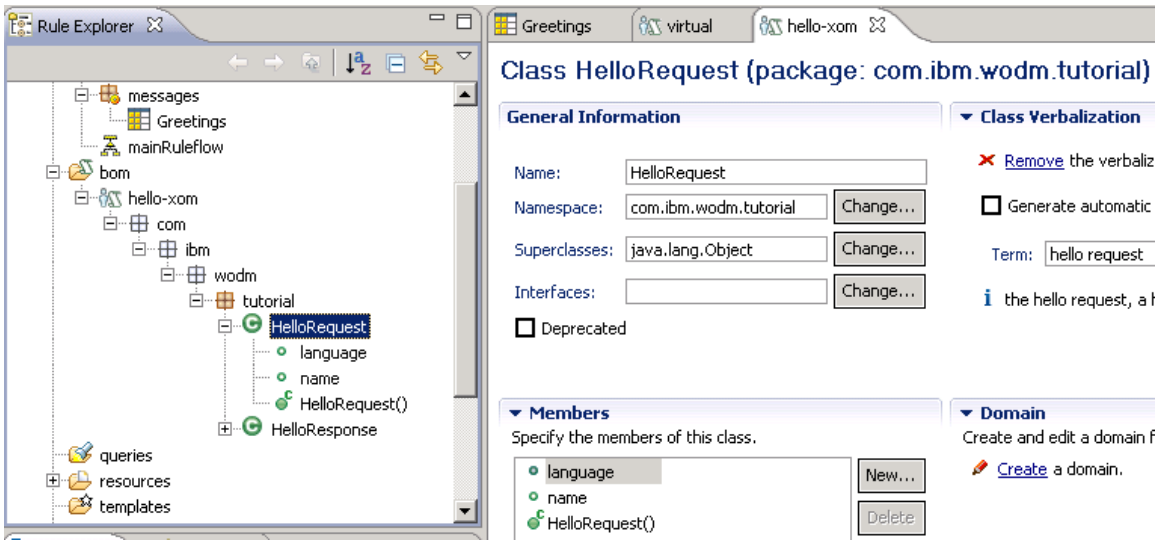
The Execution Object Model (XOM) in our example consists of 2 Java classes, a request and a response. The HelloRequest class takes a language (a string); the HelloResponse class returns a message in the language specified by the request.

Figure 5. The HelloRequest and Hello Response classes



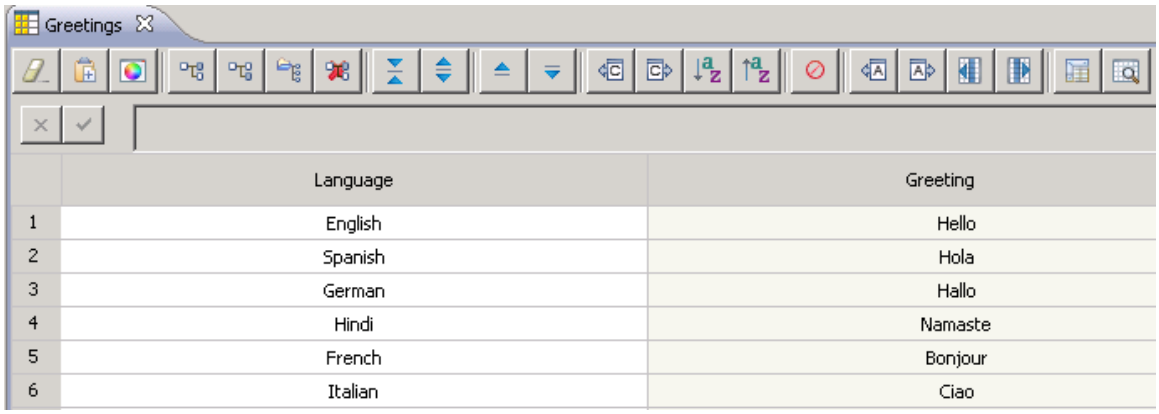
A Business Object Model (BOM) is generated from the XOM with these 2 classes.

Figure 6. The Business Object Model (BOM)



The Messages rule package contains a Greetings decision table as shown in Figure 7.

Figure 7. The Greetings decision table

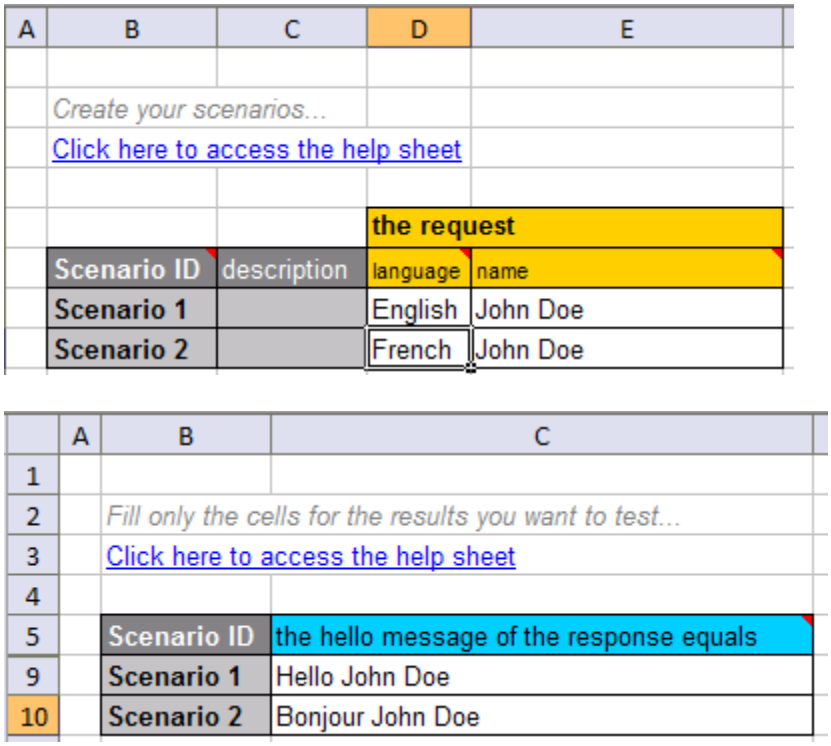


	Language	Greeting
1	English	Hello
2	Spanish	Hola
3	German	Hallo
4	Hindi	Namaste
5	French	Bonjour
6	Italian	Ciao

There are 6 rules in the rule set (each row corresponding to a rule). If the language specified in the request is English, the greeting returned is “Hello”. If the language requested is German, the greeting is “Hallo”. In this initial decision table, the language is of type string. Later in the tutorial, we will constrain the language to be a predefined set of values of type LanguageType.

The project includes a Decision Validation Services (DVS) test configuration with two scenarios and expected results (in hello-rules/testsuite.xlsx) as shown in Figure 8.

Figure 8. Decision Validation Services test scenarios

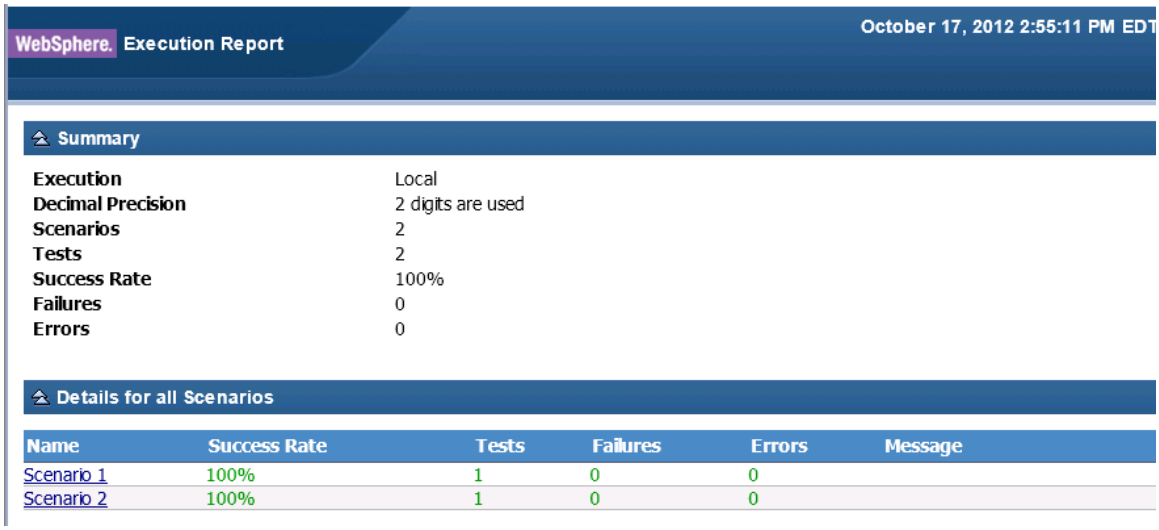


A	B	C	D	E
	Create your scenarios...			
	Click here to access the help sheet			
			the request	
	Scenario ID	description	language	name
	Scenario 1		English	John Doe
	Scenario 2		French	John Doe

	A	B	C
1			
2		*Fill only the cells for the results you want to test...*	
3		Click here to access the help sheet	
4			
5		**Scenario ID**	the hello message of the response equals
9		**Scenario 1**	Hello John Doe
10		**Scenario 2**	Bonjour John Doe

To execute the DVS test suite, right-click on DVS test.launch in hello-rules and run as ‘DVS test’. Figure 9 shows the result of the test suite execution:

Figure 9. Results of the DVS test execution



WebSphere. Execution Report

October 17, 2012 2:55:11 PM EDT

Summary

Execution	Local
Decimal Precision	2 digits are used
Scenarios	2
Tests	2
Success Rate	100%
Failures	0
Errors	0

Details for all Scenarios

Name	Success Rate	Tests	Failures	Errors	Message
Scenario 1	100%	1	0	0	
Scenario 2	100%	1	0	0	

Step 2: Create a domains.xlsx file defining the domain

Next, create an Excel spreadsheet for the dynamic domain. For this article, the file domains.xlsx has already been created under the resources folder with the following content:

Figure 10. The domains.xlsx file

A	B	C
Values	**BOM to XOM**	**Label**
ENGLISH	return "English";	English
FRENCH	return "French";	French
SPANISH	return "Spanish";	Spanish
GERMAN	return "German";	German
HINDI	return "Hindi";	Hindi
ITALIAN	return "Italian";	Italian

The Excel domain provider handles the link between the values in the Excel spreadsheet and the BOM. The Excel file must have one row for each value of the domain provider. The file must conform to this structure:

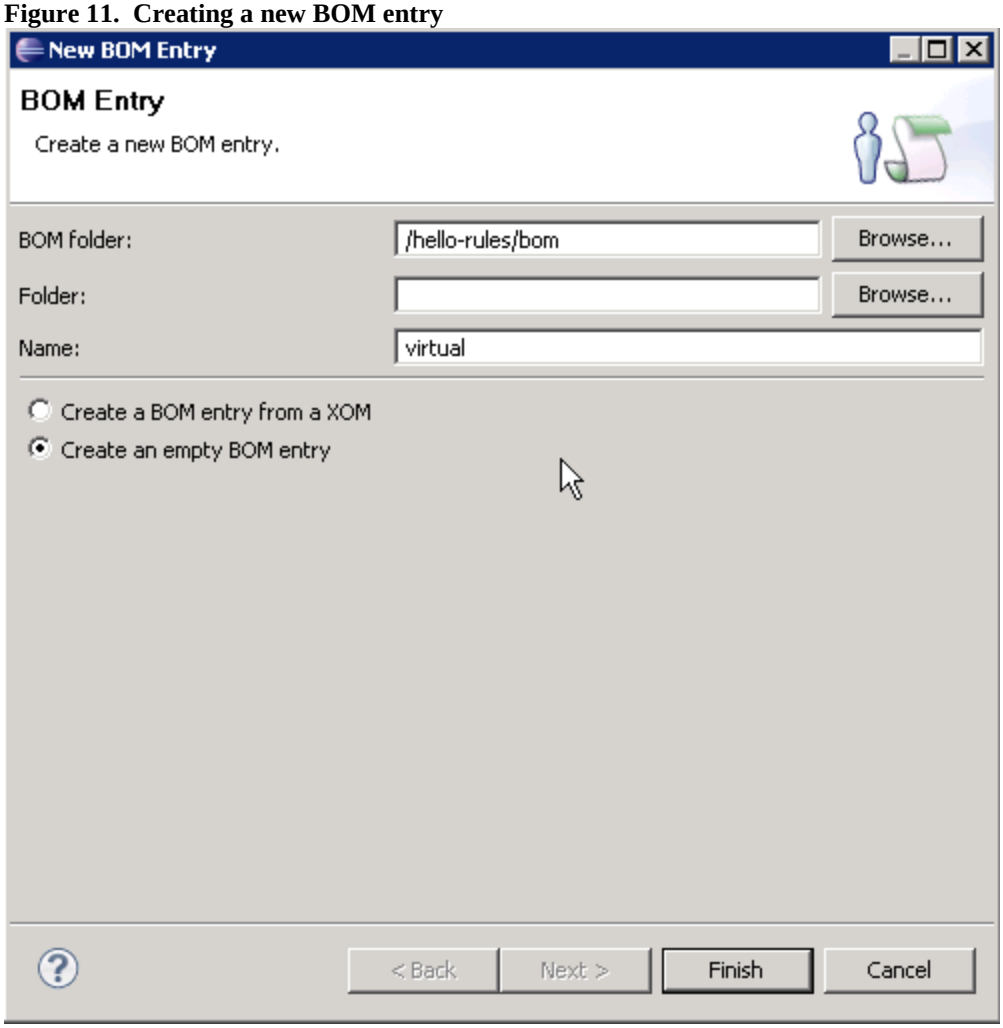
- The Values column contains the values of the domain provider.
- The BOM to XOM mapping column indicates the BOM to XOM mapping for each domain value.
- The Label column lists each language as it will be displayed when authoring a rule (the verbalization of the domain value).

Step 3: Create new virtual BOM entry

1. Right-click the bom directory and select **New -> BOM entry**.

2. In the BOM entry dialog, enter virtual for the Name field, and select the radio button "Create an empty BOM entry". Click **Finish**.

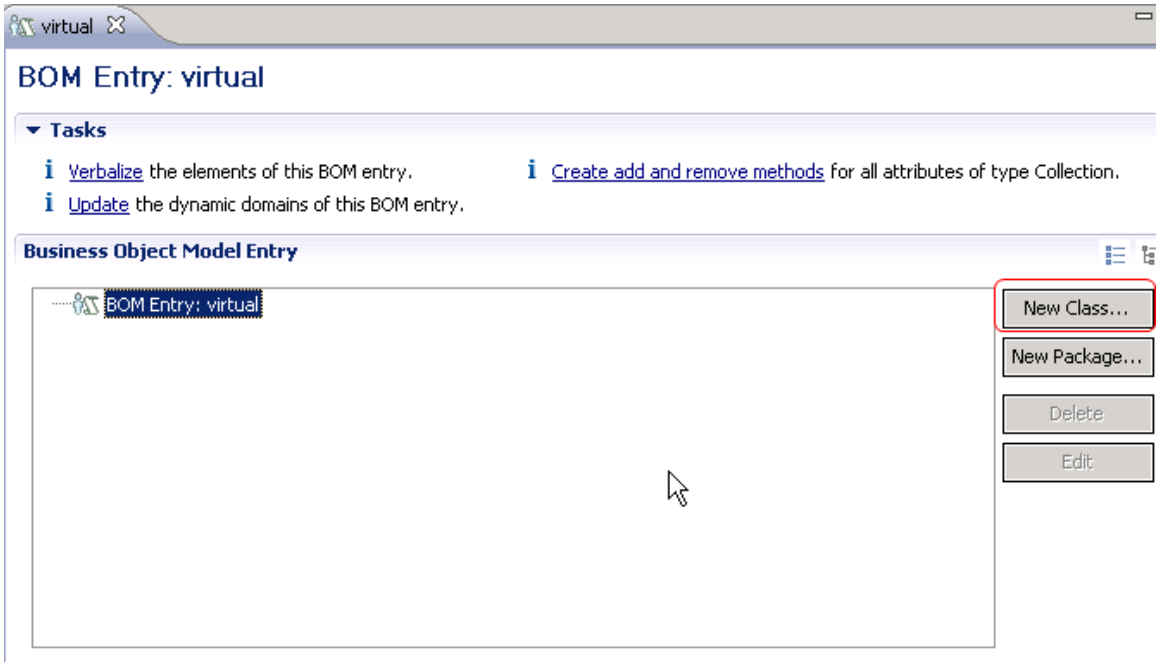
Figure 11. Creating a new BOM entry



Step 4: Create new BOM class called LanguageType class

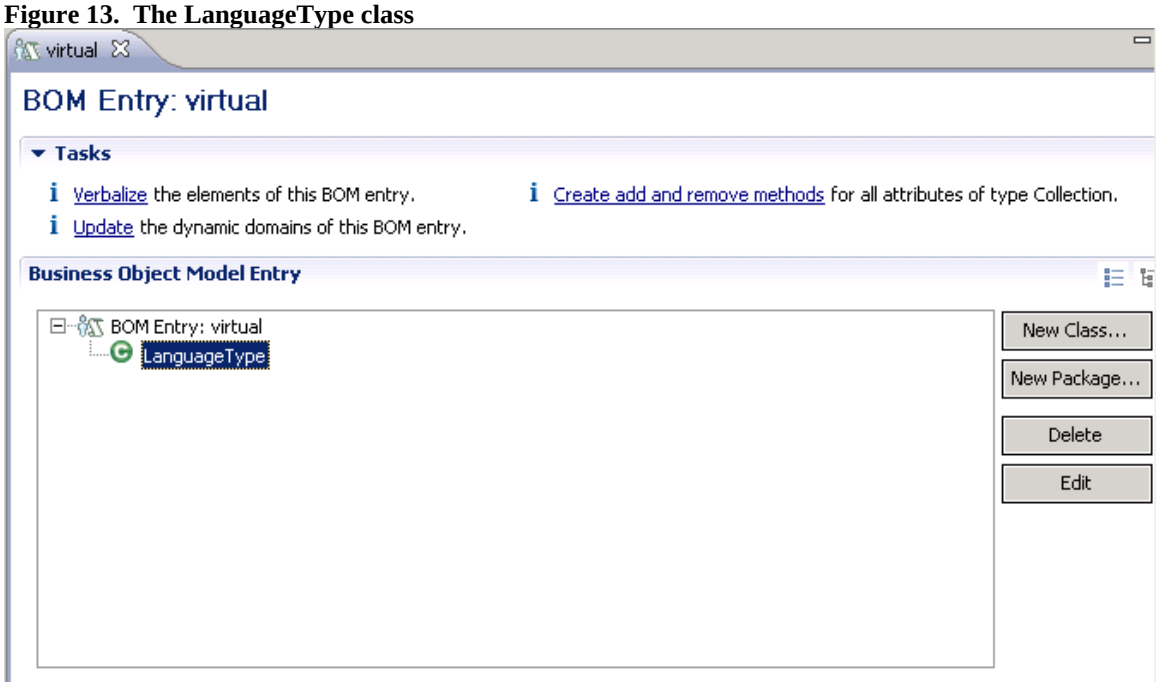
1. From the Rule Explorer, double-click the virtual BOM entry just created. Click New Class.

Figure 12. Creating the LanguageType class



2. Enter LanguageType as the name of the class, and click Finish. The result is shown in Figure 13.

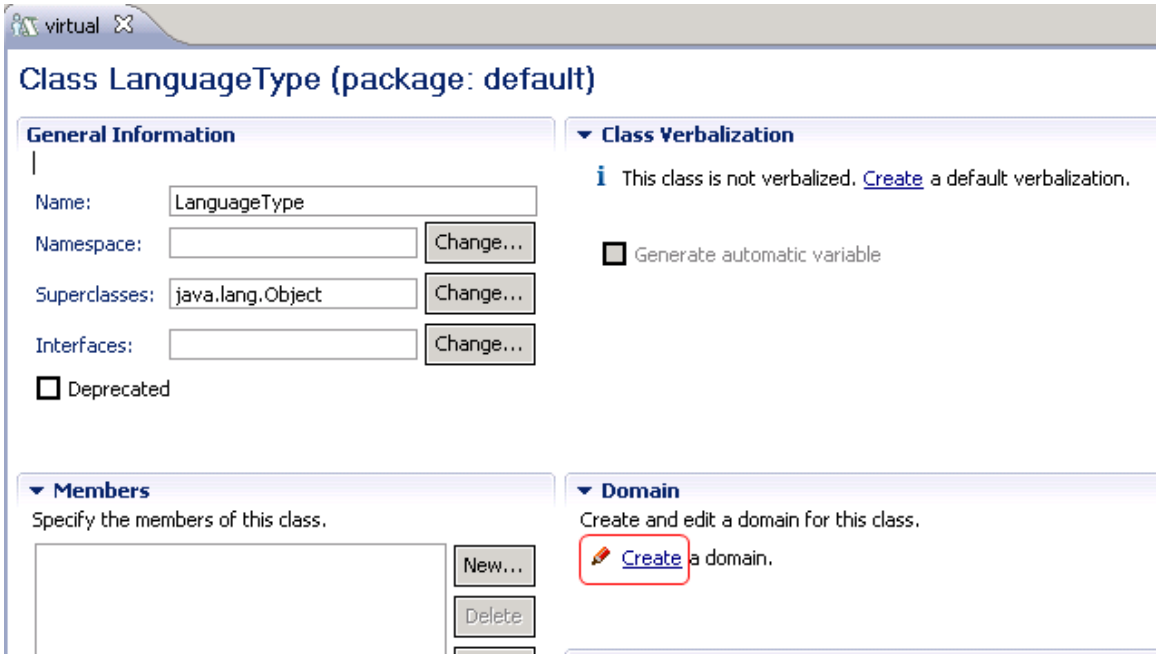
Figure 13. The LanguageType class



Step 5: Create the domain for the class

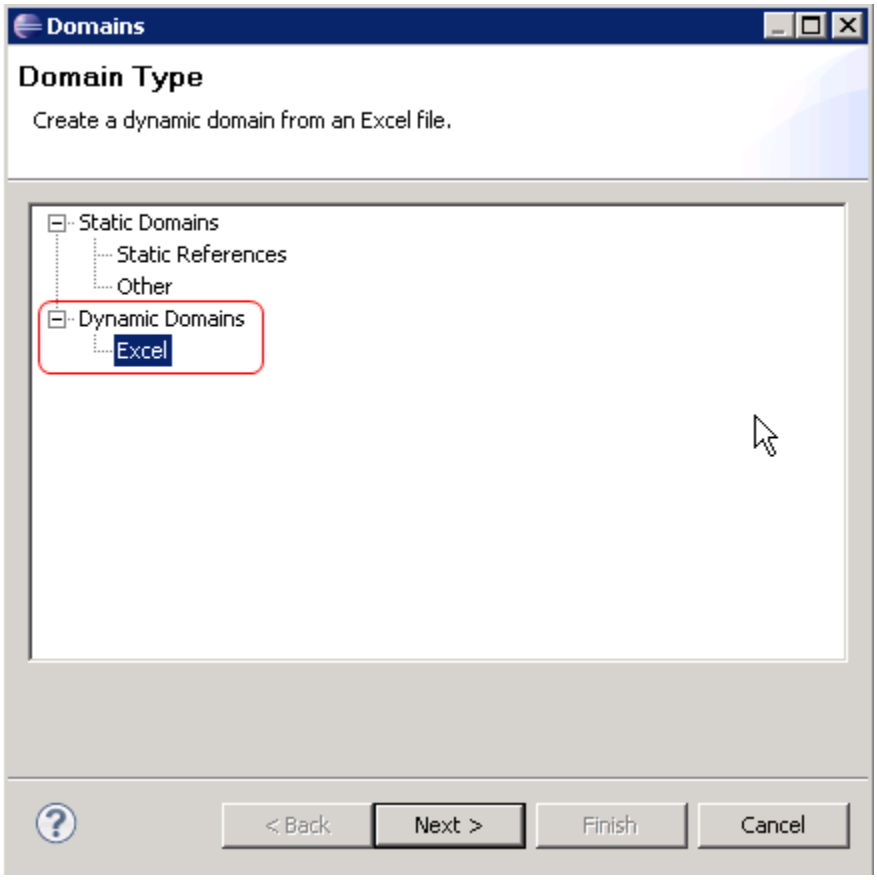
1. Double click Language Type. Select Create a domain.

Figure 14. Creating a domain for the class



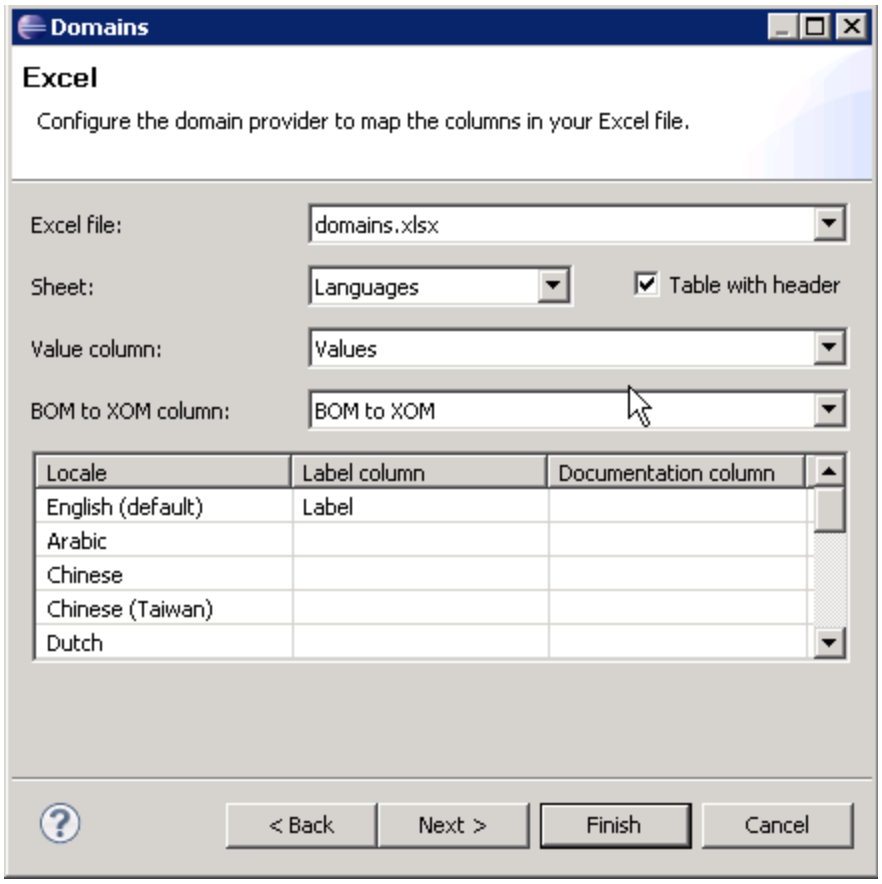
2. Under Dynamic Domains, choose Excel and click Next.

Figure 15. Selecting the domain type (Excel)



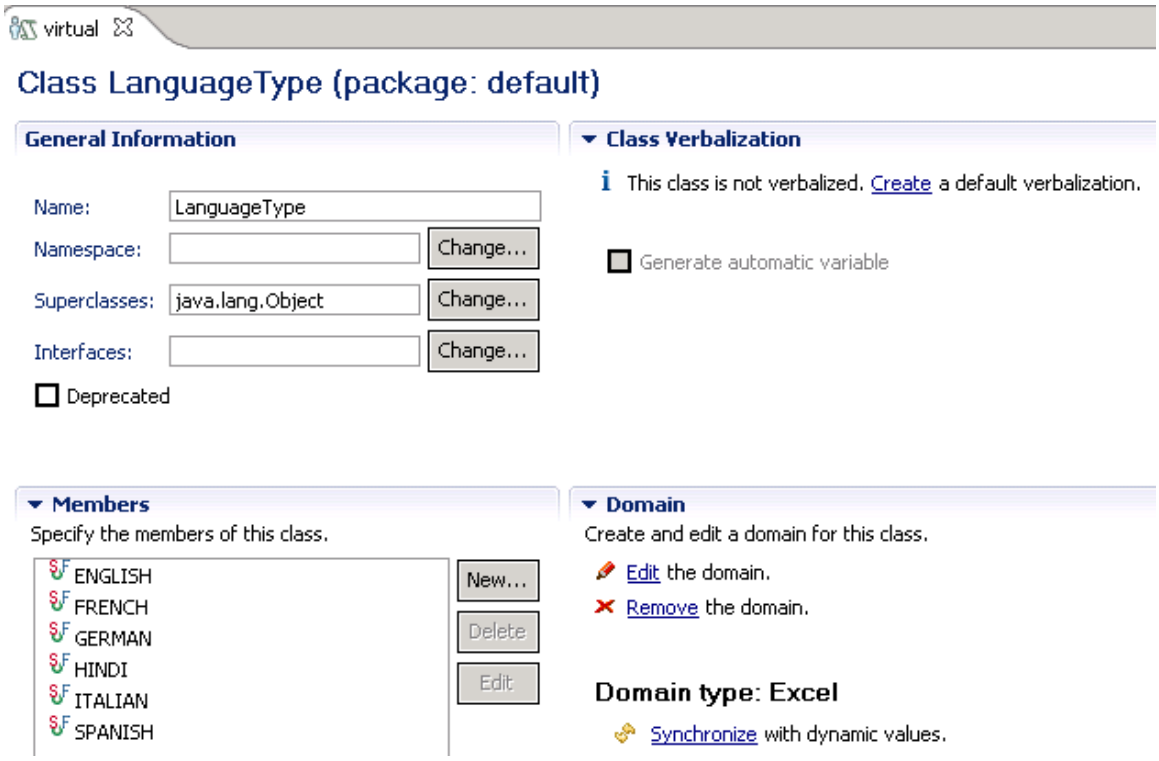
3. Enter the name of the Excel file that defines the domain. Indicate Languages as the name of the sheet. ***Be sure to select the "Table with header" checkbox***. Enter Values for the Value column. Enter BOM to XOM for the BOM to XOM column. Select Label as the Label column.
4. Click **Finish**.

Figure 16. Mapping the columns of the Excel file



The domain values now appear as members of the class.

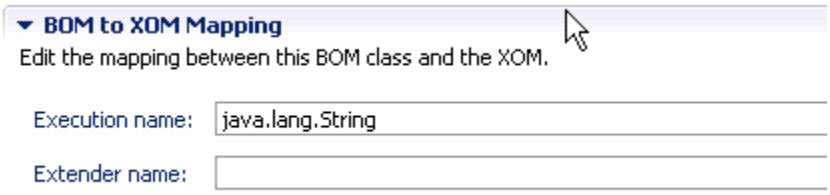
Figure 17. The domain values



5. Expand the BOM to XOM Mapping section. You may need to maximize the tab and scroll down to see this section.

6. Enter java.lang.String as the Execution name for the BOM to XOM Mapping.

Figure 18. Mapping the BOM class to the XOM

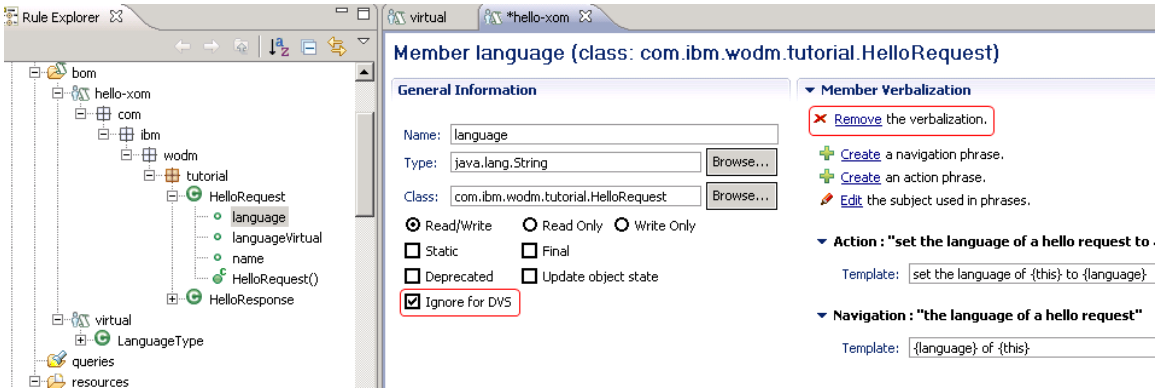


7. Save the project.

Step 6: Create a new member of HelloRequest class

1. From the bom directory, navigate to the HelloRequest class, double click language, and remove the verbalization. Select the checkbox “Ignore for DVS” so that language is not used during testing.

Figure 19. Removing the original verbalization for the HelloRequest class

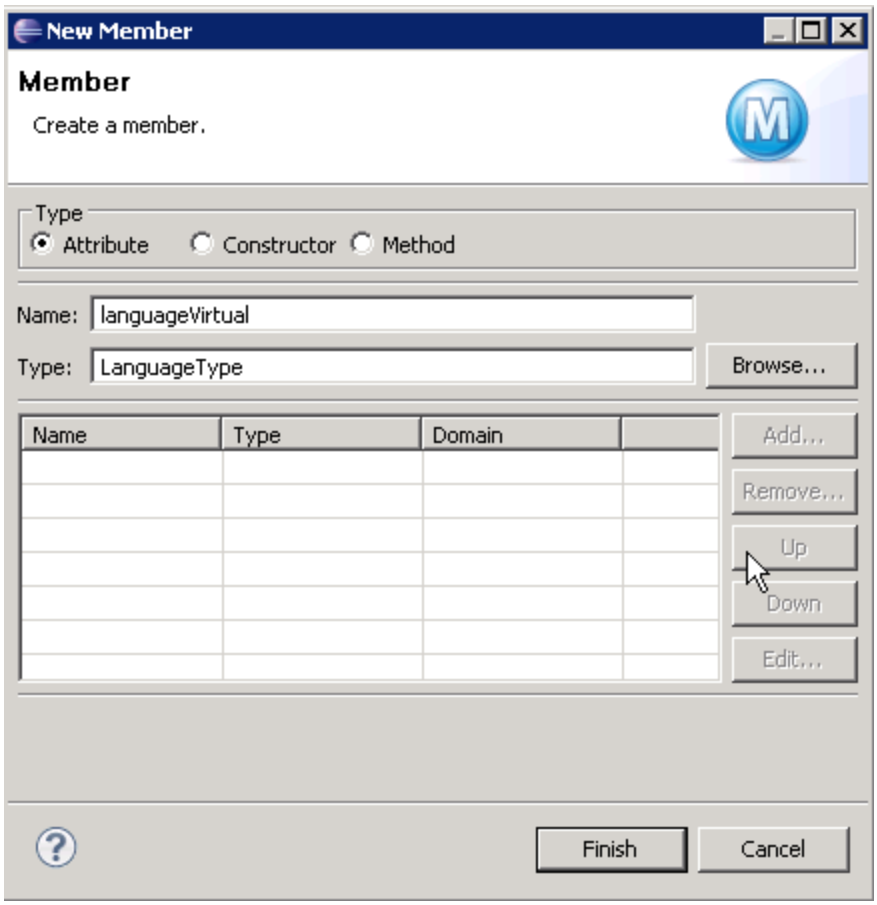


2. Save your work.

TIP: You will see errors in the Rule Explorer view. However, these errors will be resolved when the remaining steps are completed.

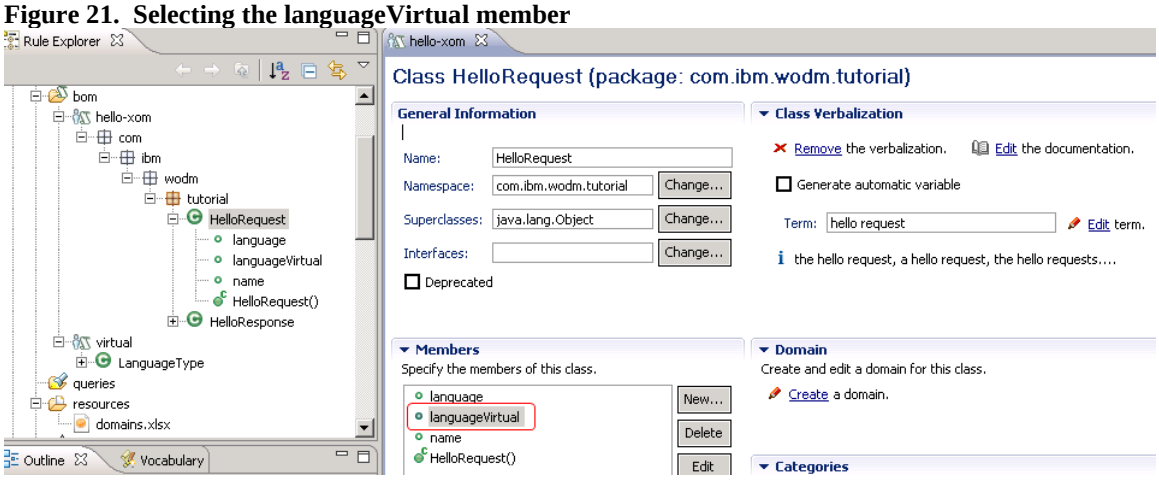
3. Create a new member of the HelloRequest class called languageVirtual. Double-click the HelloRequest class. Under the Members section, click New. It is of type LanguageType (instead of string). Click **Finish**.
4. Save your work.

Figure 20. Creating the languageVirtual member



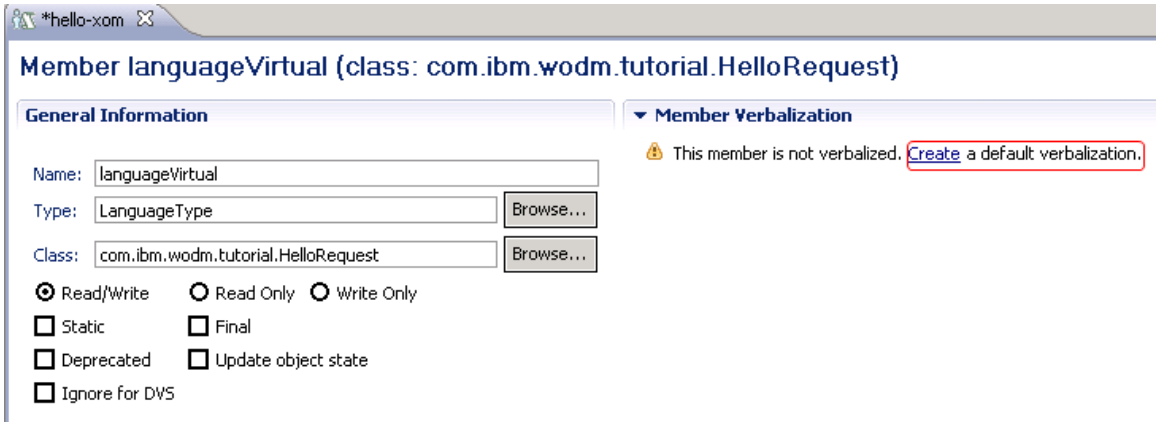
5. Double-click on the member languageVirtual.

Figure 21. Selecting the languageVirtual member



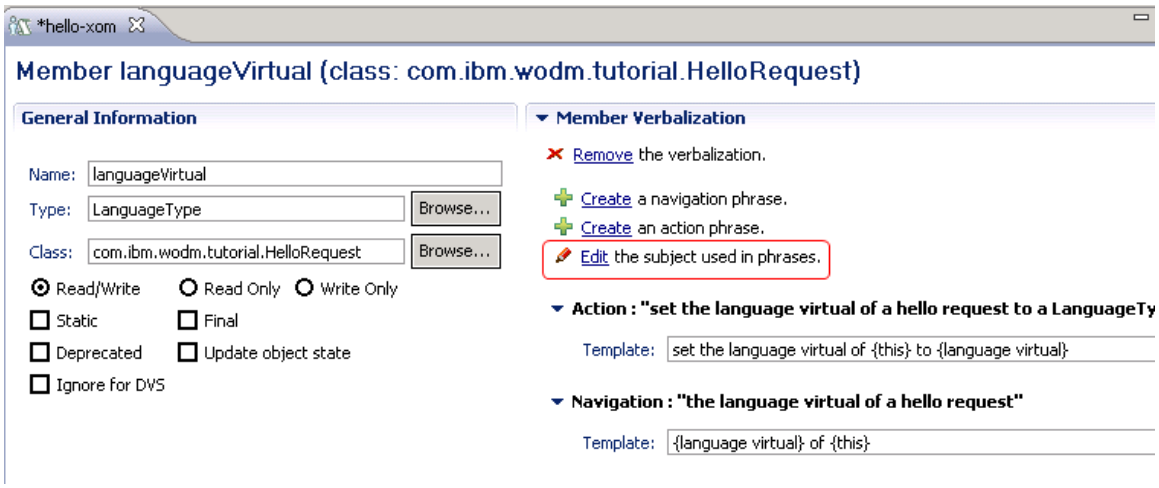
6. Under the Member Verbalization section, click Create to create a default verbalization.

Figure 22. Creating a default verbalization for languageVirtual



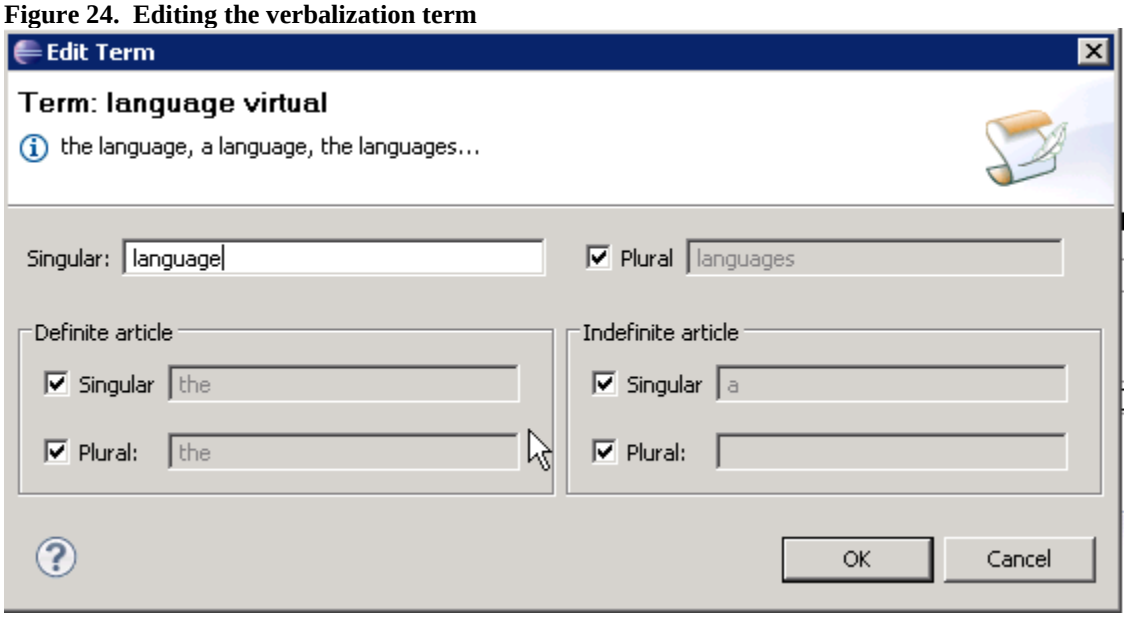
7. Modify the verbalization by clicking "Edit the subject used in phrases."

Figure 23. Editing the verbalization



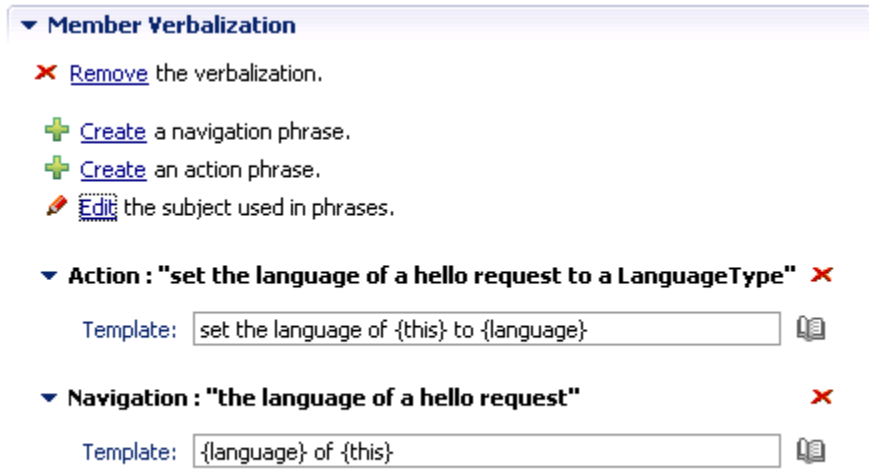
8. In the dialog box, change languageVirtual to language. Click OK.

Figure 24. Editing the verbalization term



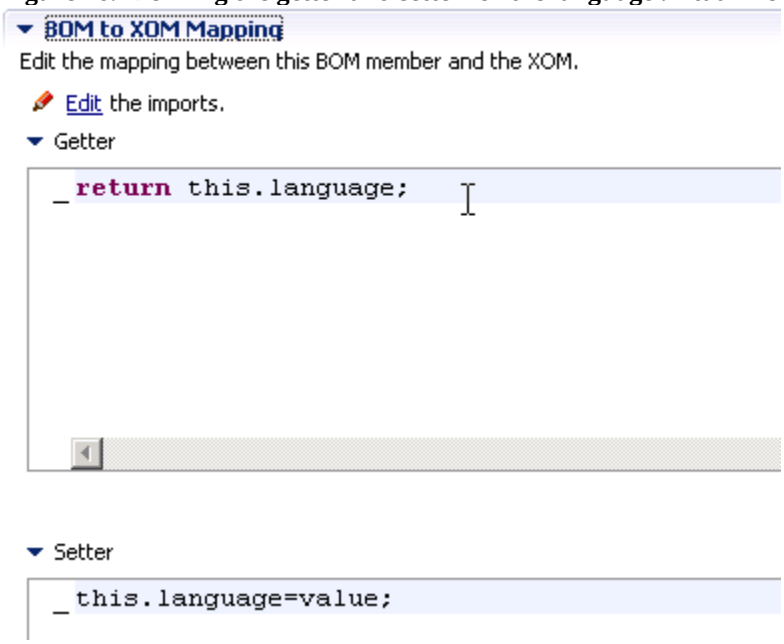
9. Remove the word virtual from the Action clause so that the final verbalization matches Figure 25.

Figure 25. The final verbalization



10. Under the BOM to XOM Mapping section, define the getter and setter for the new member as shown in Figure 26. You may need to maximize the tab and scroll down to see this section.

Figure 26. Defining the getter and setter for the languageVirtual member



11. Save your work.

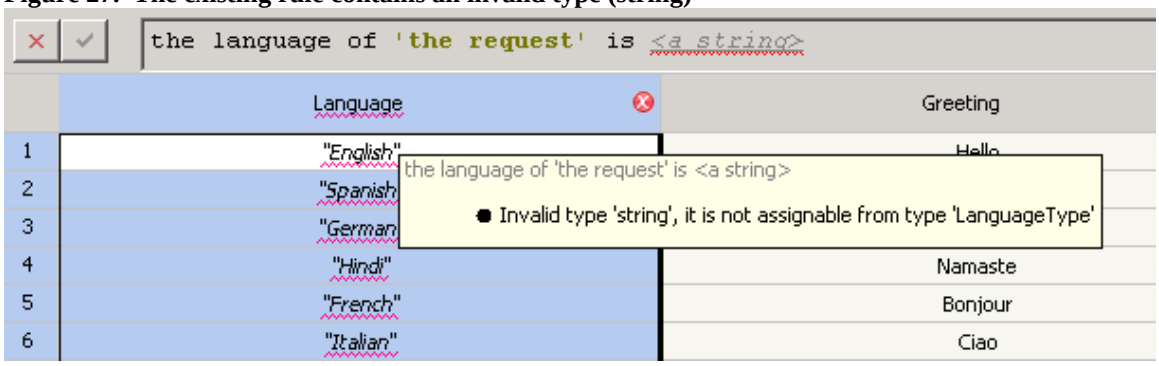
Thus far, you created a new attribute that represents the underlying "language", defined getters and setters for it, verbalized it, and made it of type LanguageType. You also removed the verbalization from the original "language".

Step 7: Modify the corresponding rules

The original rules are expecting a string. Therefore the rules must be adjusted to now expect input of type languageType.

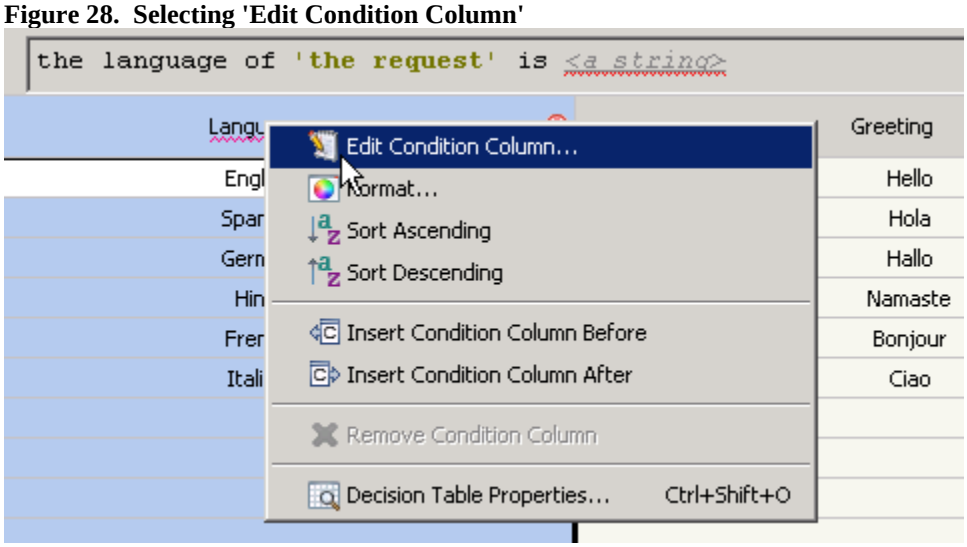
1. Under the rules folder, double click the Greetings rule package. String is now an invalid type.

Figure 27. The existing rule contains an invalid type (string)



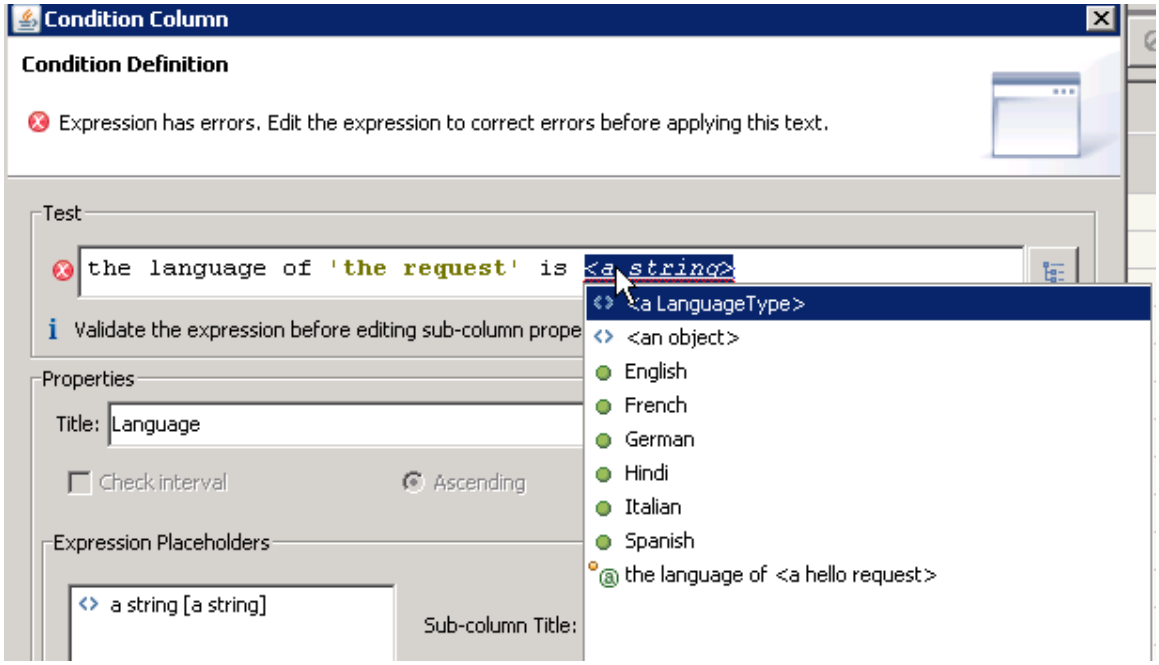
2. To correct this, right-click on the column header Language and select Edit Condition Column…

Figure 28. Selecting 'Edit Condition Column'



3. Edit the Condition Column and indicate LanguageType instead of string.

Figure 29. Replacing string with LanguageType



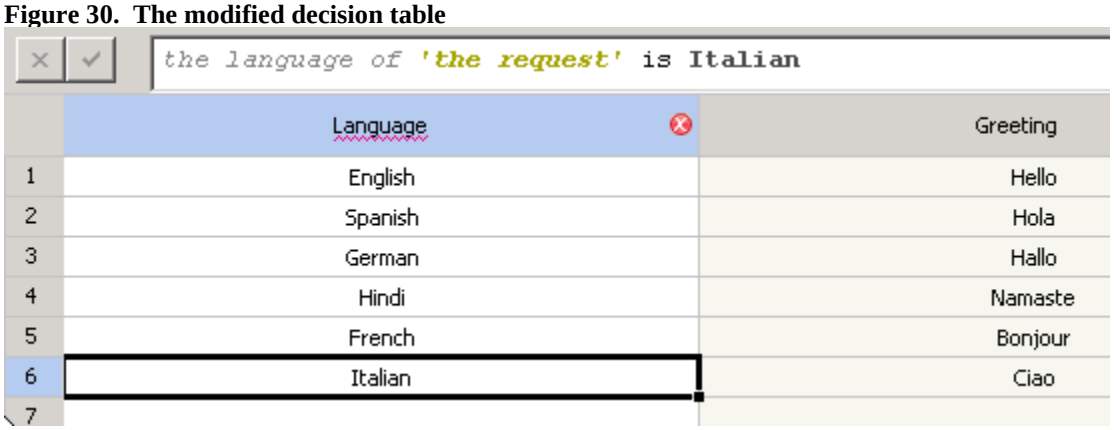
4. Click Apply and then click OK.

5. Double click on each string in the Language column and choose the appropriate value from the dropdown (English for "English", and so on).

6. Save your work.

The resulting Decision table should look like Figure 30.

Figure 30. The modified decision table



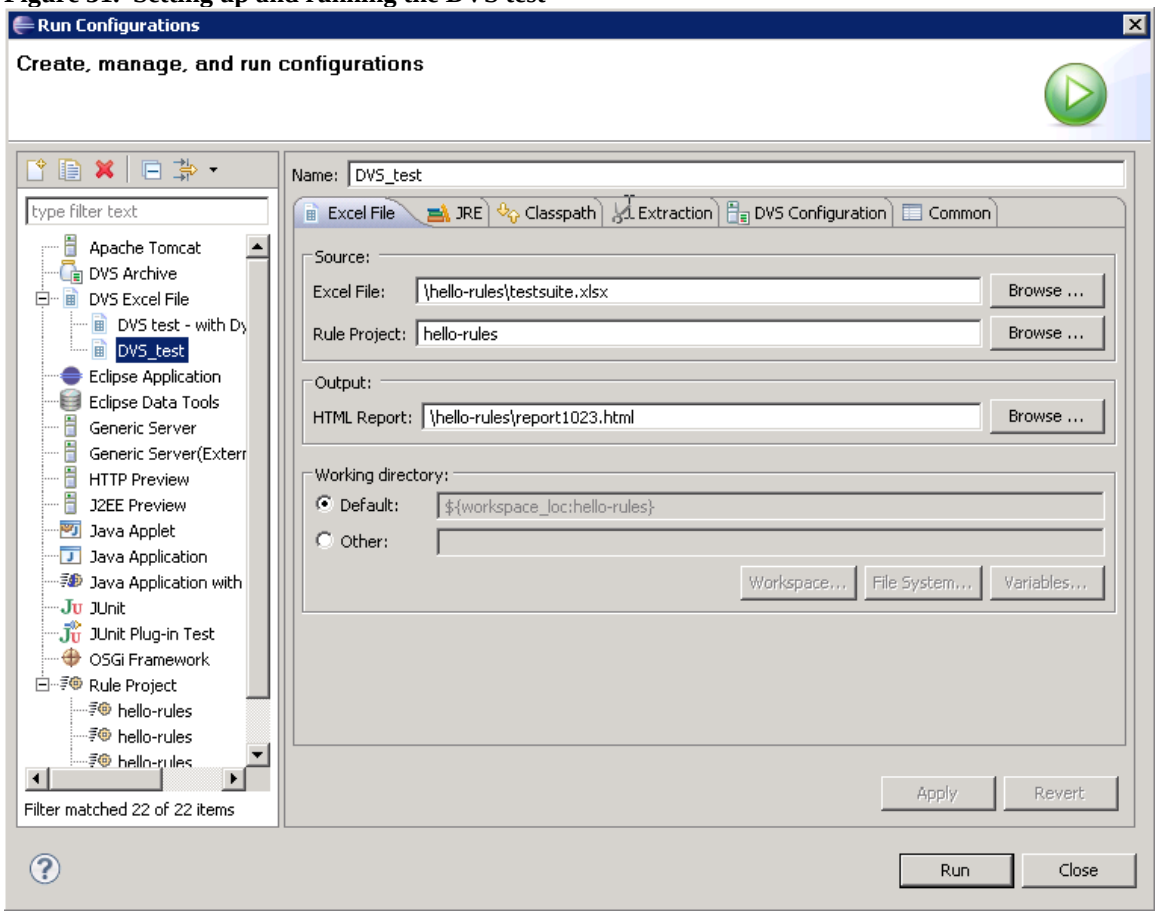
	Language	Greeting
1	English	Hello
2	Spanish	Hola
3	German	Hallo
4	Hindi	Namaste
5	French	Bonjour
6	Italian	Ciao
7		

Step 8: Rerun the test

In this step, you rerun the DVS test as a regression test to ensure that the change to the dynamic domain does not impact the expected results.

1. From the main menu, select **Run -> Run Configurations…**
2. Select DVS_test under **DVS Excel file**.
3. Verify the path of the Excel file containing the test suite and indicate the desired location of the HTML output report.

Figure 31. Setting up and running the DVS test



4. Verify the test results by viewing the HTML report with a browser.

Step 9: Add new values to the spreadsheet

1. First open the spreadsheet domains.xlsx and create a new row for each new value. In this example, a row has been added for the Alsatian and Portuguese languages.

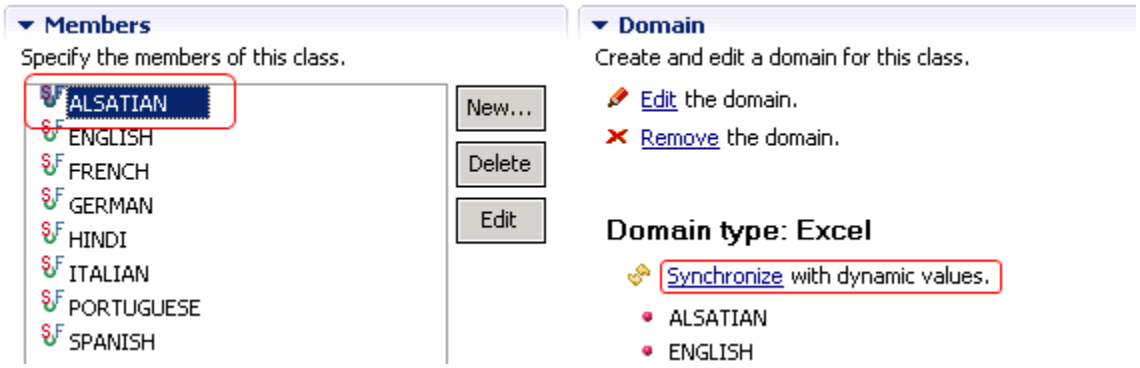
Figure 32. Adding new values in the Excel file

A	B	C
Values	**BOM to XOM**	**Label**
ENGLISH	return "English"	English
FRENCH	return "French";	French
SPANISH	return "Spanish";	Spanish
GERMAN	return "German";	German
HINDI	return "Hindi";	Hindi
ITALIAN	return "Italian";	Italian
PORTUGUESE	return "Portuguese";	Portuguese
ALSATIAN	return "Alsatian";	Alsatian

2. Save the file and refresh the resources folder.

3. Under the folder bom -> virtual, double-click the LanguageType class. Under Domain type: Excel, click Synchronize. The new values appear in the list of class members.

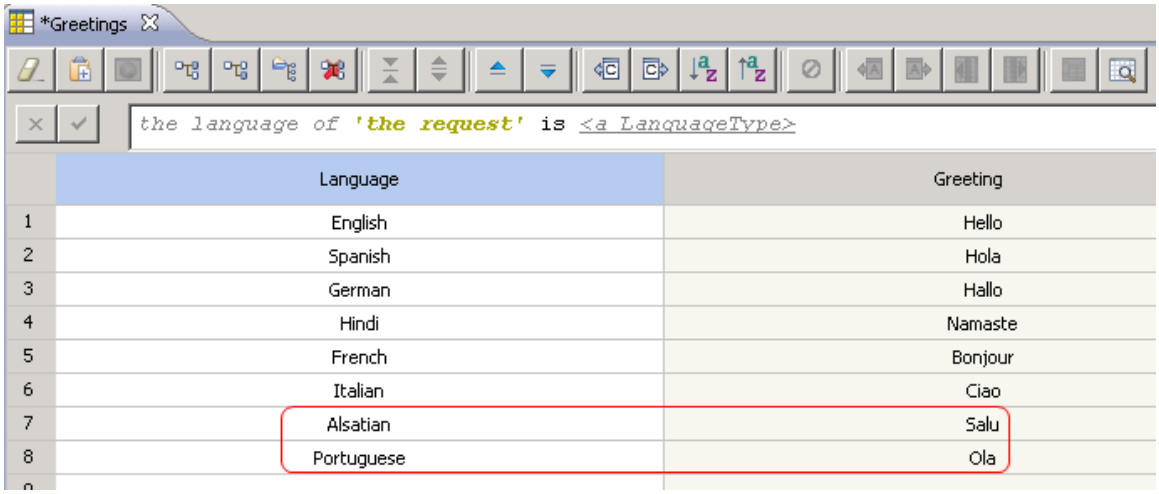
Figure 33. Synchronizing new domain values with the class



TIP: After any updates to a domain, refresh the resources folder containing the Excel sheet, and use the Synchronize link to synchronize the updates with the class.

4. Return to the Greetings decision table in the rule package. Use the dropdown menu to select the new languages and then type the expected response in the Greeting column.

Figure 34. Updating the decision table

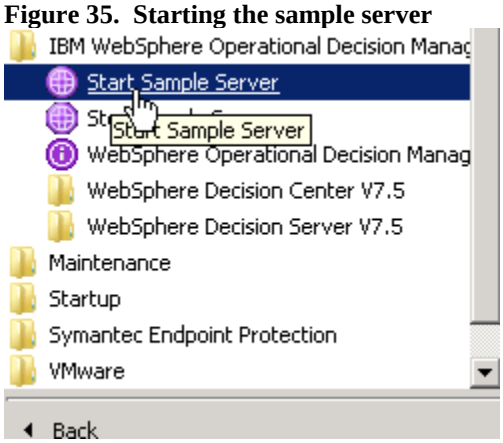


You can then extend your test suite by adding scenarios to cover the newly added languages.

Step 10: Publish to Decision Center

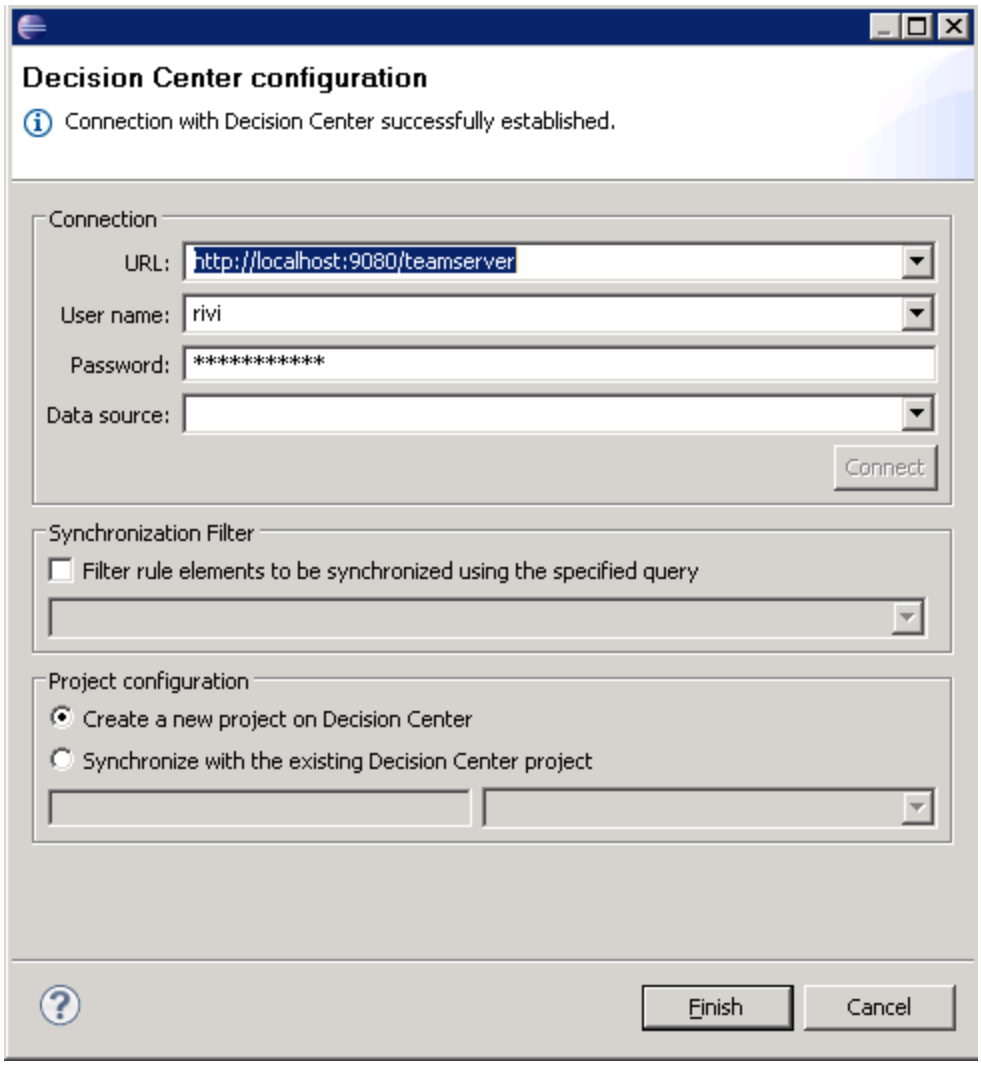
1. From the Start menu, start the Sample server if it is not already started.

Figure 35. Starting the sample server



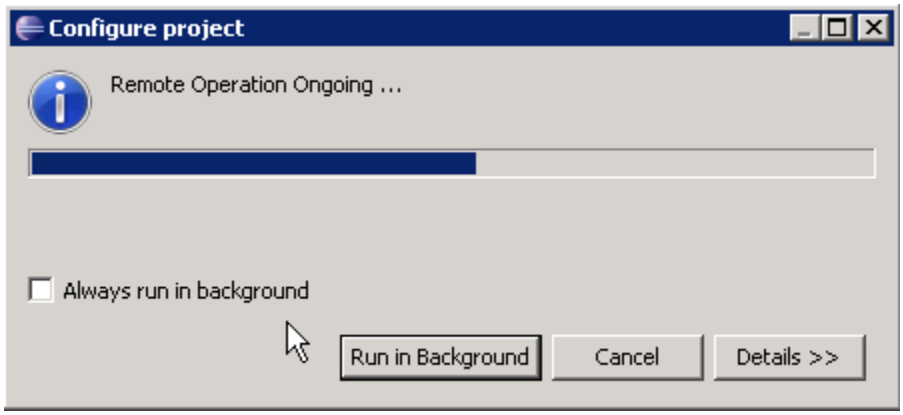
2. Right-click the project and select **Decision Center -> Connect**. Enter the URL used to access the Decision Center and supply your authentication information. Click the Connect button.

Figure 36. Configuring the connection to the Decision Center



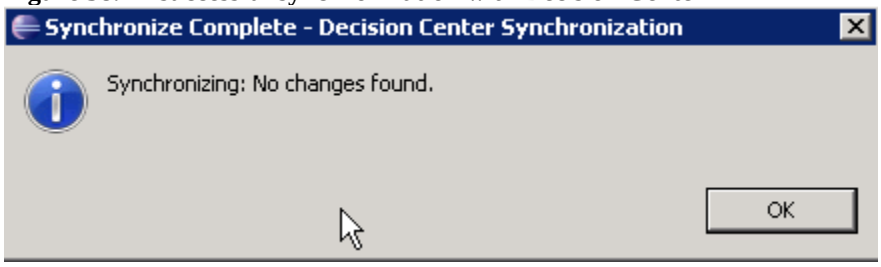
3. When the Remote Operation is complete, click Finish.

Figure 37. Connecting to the Decision Center



4. When synchronization is complete click OK.

Figure 38. A successful synchronization with Decision Center



Part 2. Modifying dynamic domains in the Decision Center

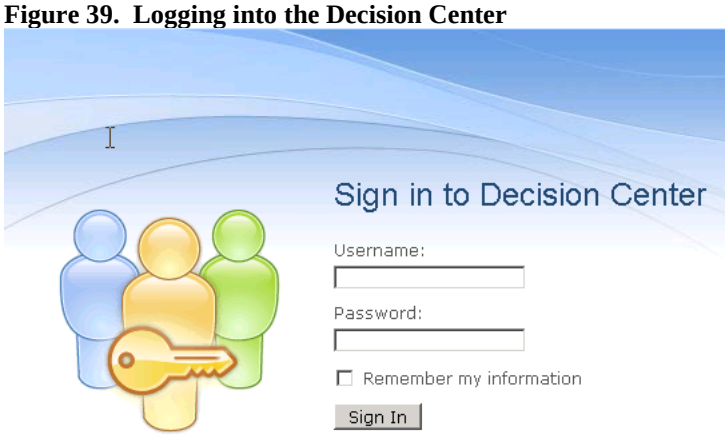
Using dynamic domains allows business users to implement and deploy a change to a business rule quickly without the aid of IT support. Rules developed with enumerations would require code modifications to accommodate new values; with dynamic domains a business user can make simple edits to the underlying Excel spreadsheet that defines the domain. The new values are immediately available for testing and deployment, and eventual synchronization with the Rule Designer.

The following steps demonstrate how a business user can add a new value to a predefined list of languages. The steps assume that the project "practice-hello-rules" has been synchronized with the Decision Center and is ready to use.

Step 1: Log into the Decision Center

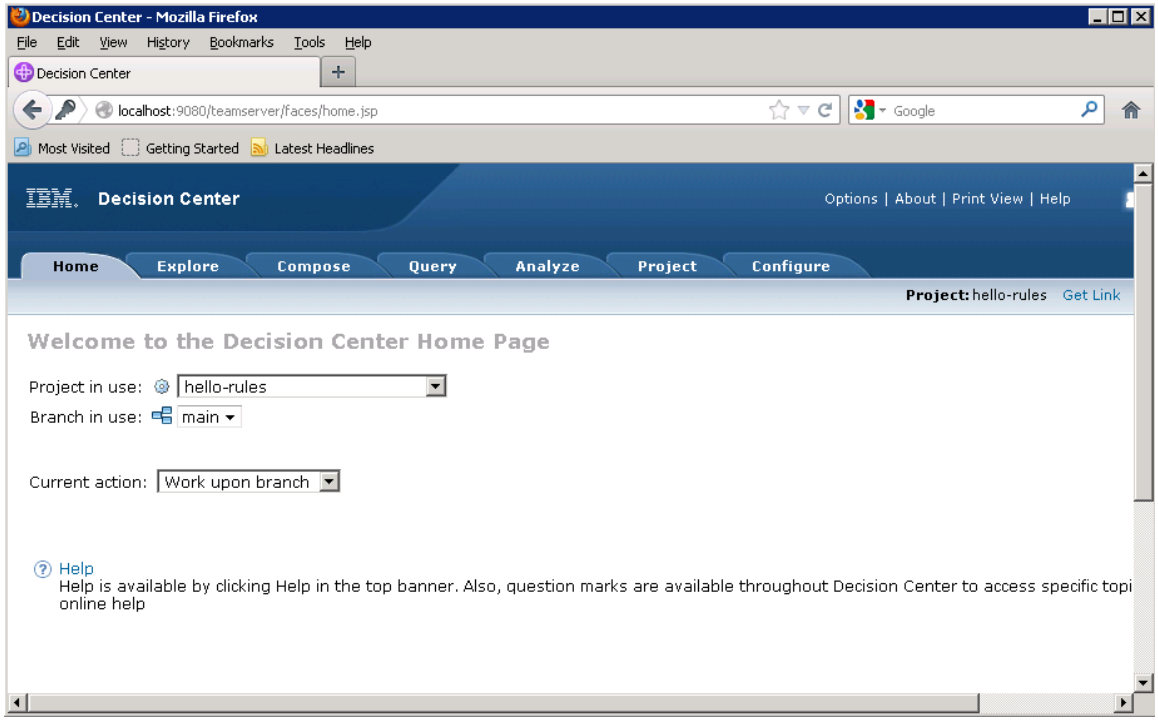
1. Enter your username and password on the Decision Center sign-in page.

Figure 39. Logging into the Decision Center



2. On the Home tab, select your project from the Project in use menu.

Figure 40. Selecting a project in Decision Center



3. On the Explore tab, under Business Rules, review the values of the Greetings decision table imported from Rule Designer. Click the Greetings link.

Figure 41. Accessing the Greetings decision table in Rule Designer

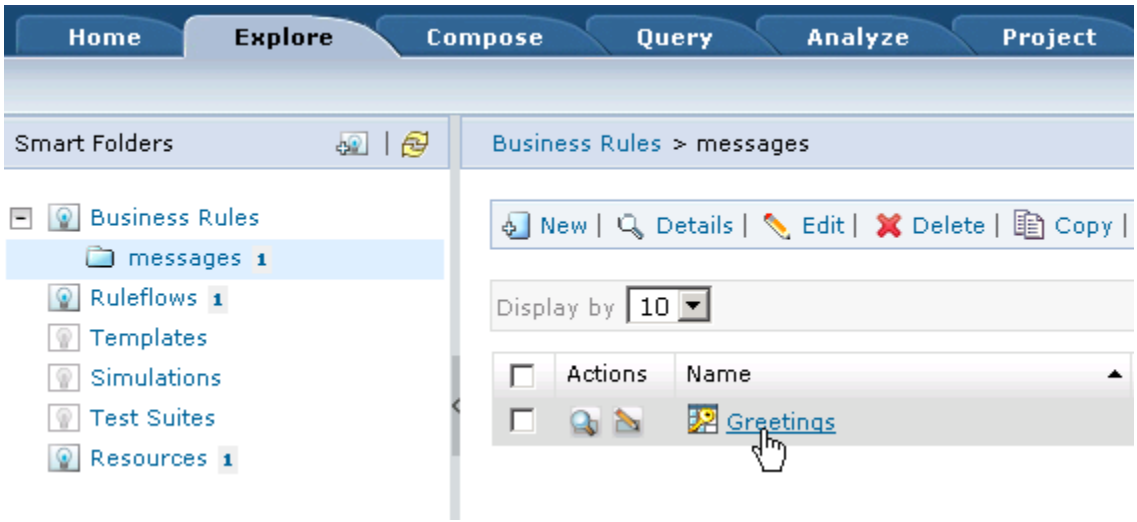
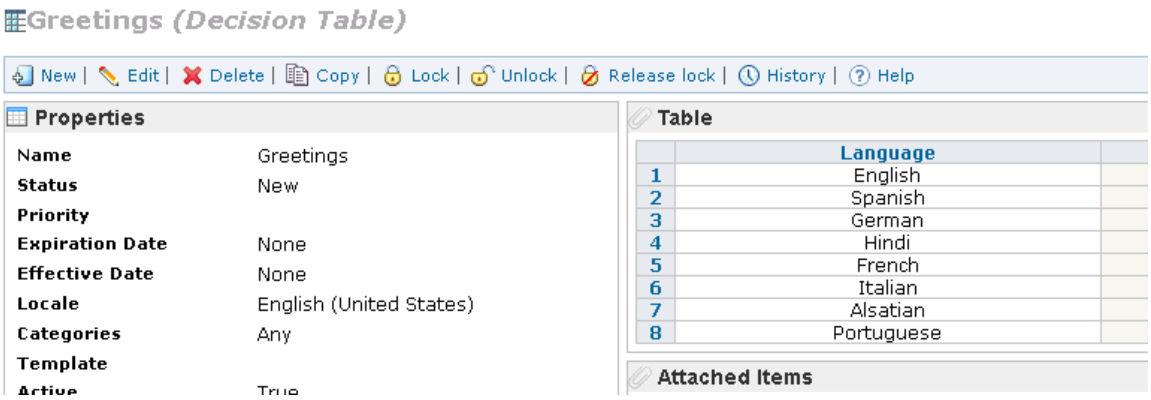


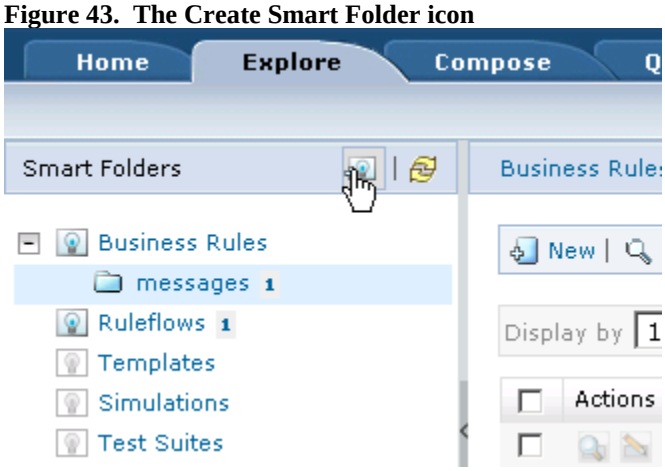
Figure 42. The Greetings decision table



Step 2: Create Resources Smart folder

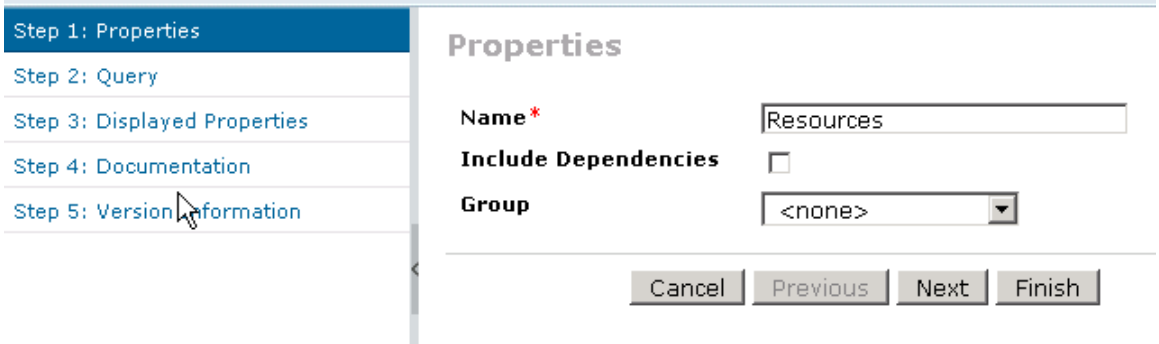
1. On the Explore tab, click the Create Smart Folder icon. Create a new smart folder called Resources.

Figure 43. The Create Smart Folder icon



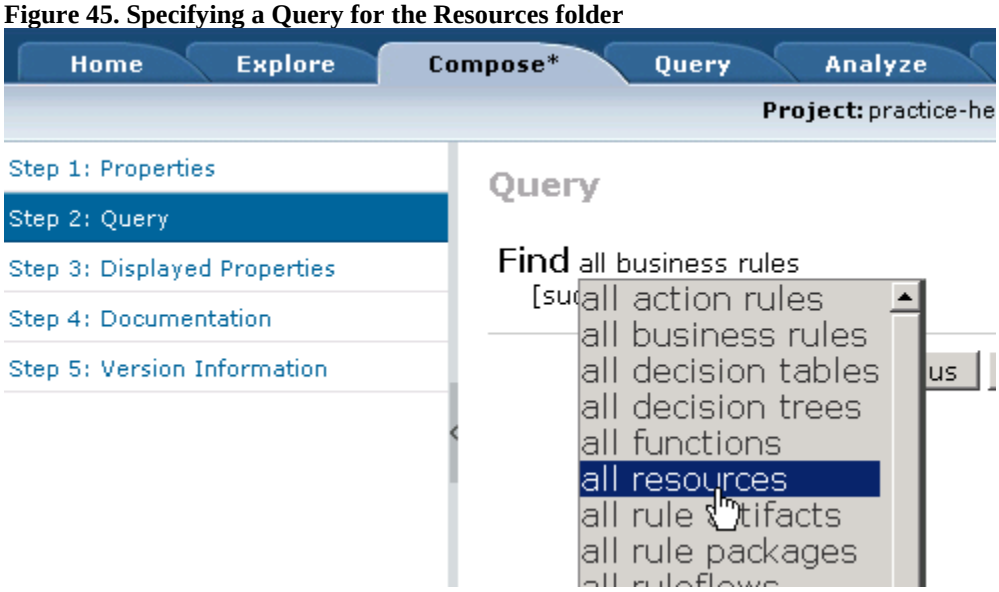
2. For Properties, enter Resources as the folder name. Click Next.

Figure 44. Creating a Resources folder



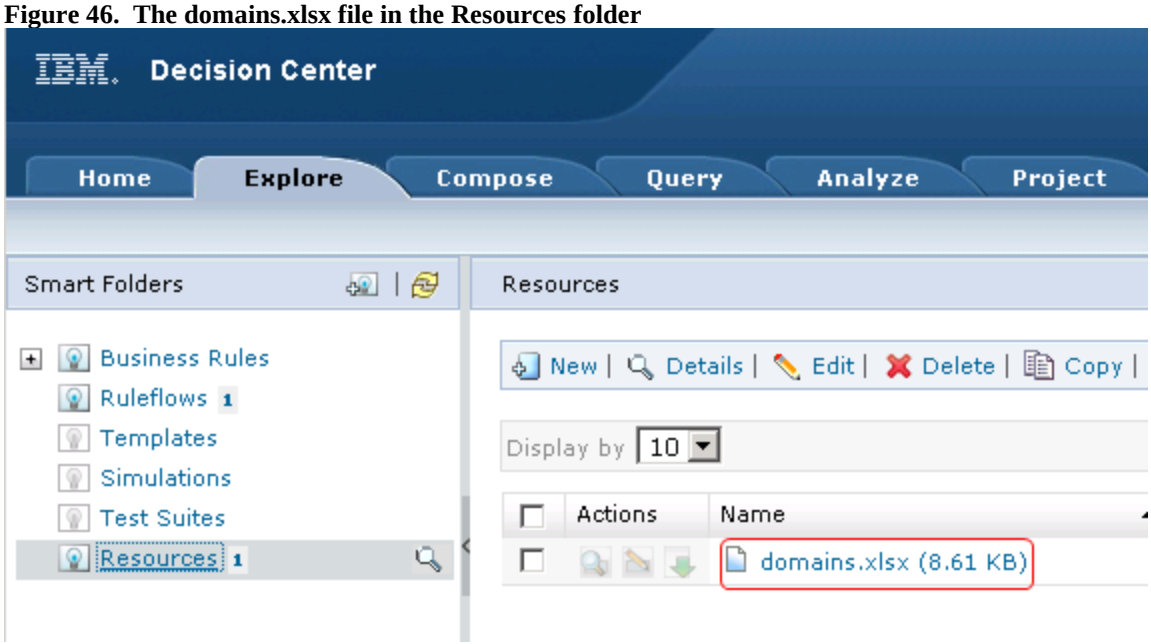
3. For Step 2. Query, indicate that all resources are to be searched and displayed. Click **Finish**.

Figure 45. Specifying a Query for the Resources folder



The domains.xlsx file should be visible as the element in this new folder.

Figure 46. The domains.xlsx file in the Resources folder

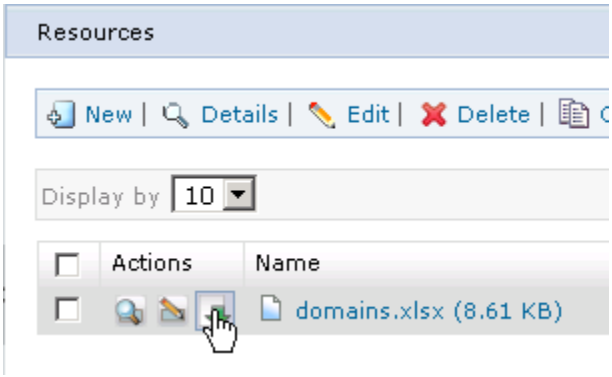


Step 3: Modify domains.xlsx in the Resources folder

In this step we download the Excel file to our local disk and add a row for a new language, Swedish. We then upload this file to the Decision Center and refresh the project.

1. Click the download icon next to the file name. Save the file to your local disk and then open it for editing.
2. Add a new row for the Swedish language.

Figure 47. Downloading the domains.xlsx file



3. Edit the file, adding a new row for the new language. Save the file.

Figure 48. Adding a new rows to the Excel file

	A	B	C
1	**Values**	**BOM to XOM**	**Label**
2	ENGLISH	return "English"	English
3	FRENCH	return "French";	French
4	SPANISH	return "Spanish";	Spanish
5	GERMAN	return "German";	German
6	HINDI	return "Hindi";	Hindi
7	ITALIAN	return "Italian";	Italian
8	PORTUGUESE	return "Portuguese";	Portuguese
9	ALSATIAN	return "Alsatian";	Alsatian
10	SWEDISH	return "Swedish";	Swedish
11			

4. Back in the Decision Center, click the domain.xlsx link. Click Edit.

Figure 49. Editing the link to the domains.xlsx file

5. Browse and select the updated domains.xlsx file to upload it back into the Decision Center.

Figure 50. Uploading the new domains.xlsx file

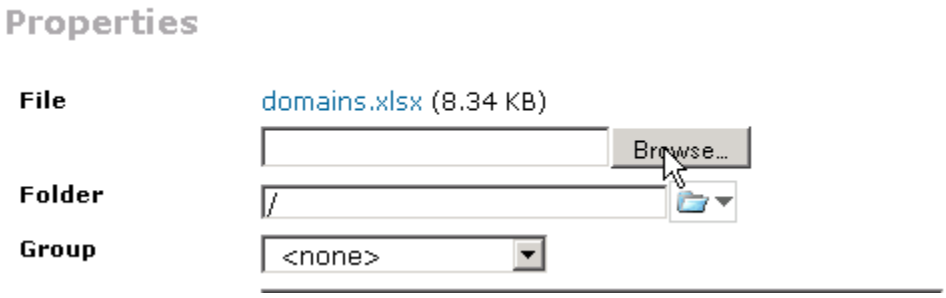
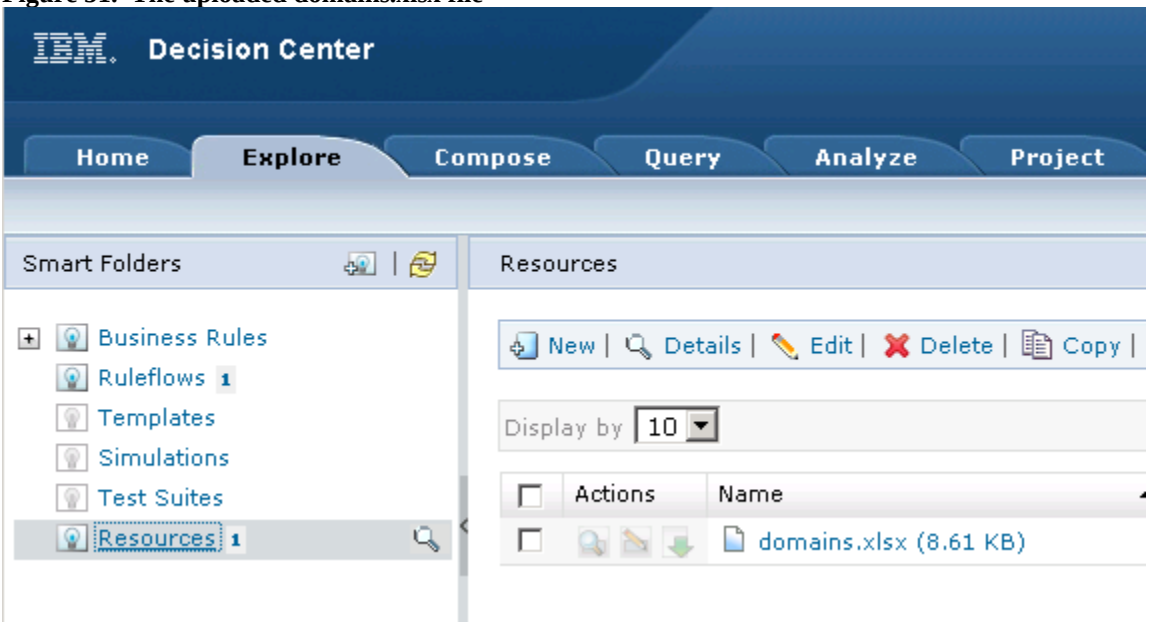


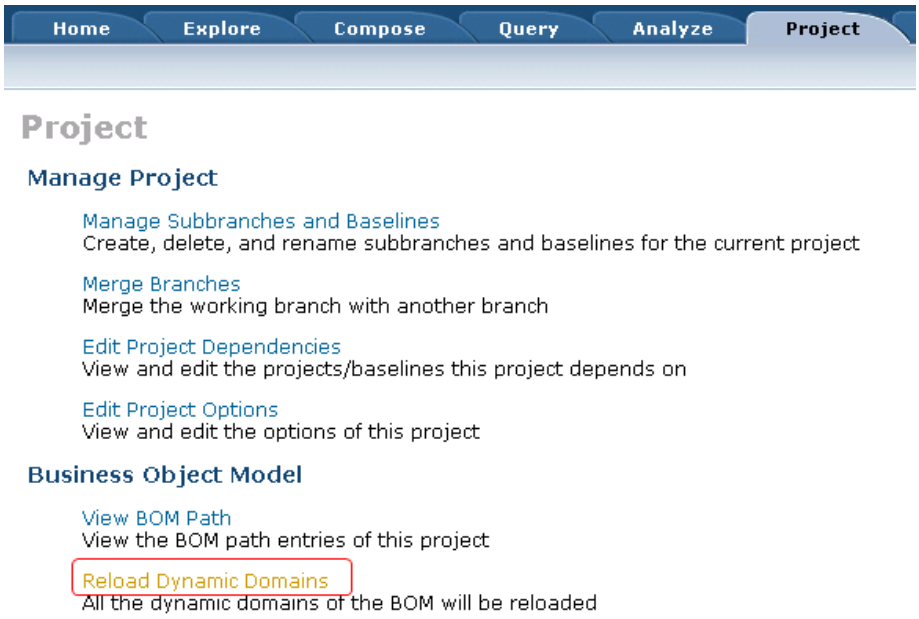
Figure 51. The uploaded domains.xlsx file



Step 4: Reload dynamic domains

From the Project tab select Reload Dynamic Domains to refresh the project.

Figure 52. Reloading the dynamic domain



Step 5: Add the new value to the decision table

The new value Swedish can now be used when authoring a rule. To make it available in the Decision Center, update the decision table.

1. Under Business Rules, click the Greetings link and then click Edit.

Figure 53. Selecting the Greetings decision table

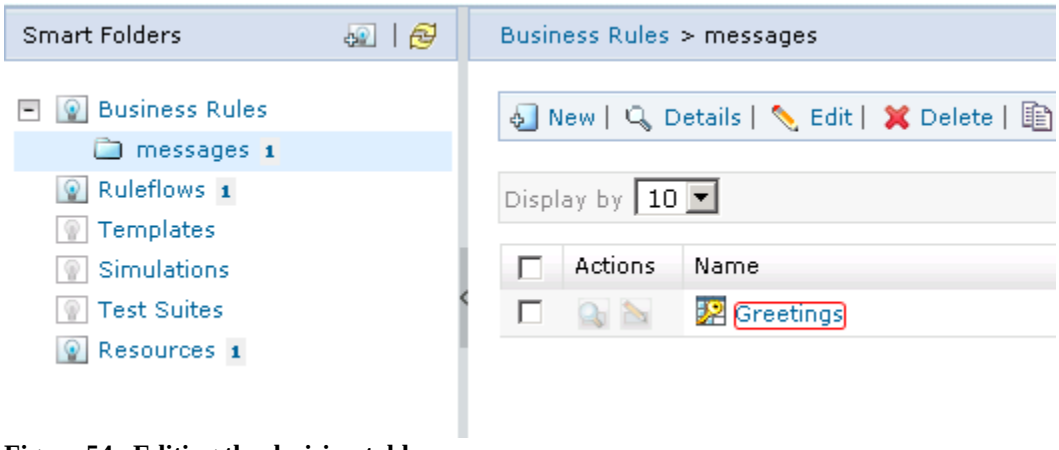
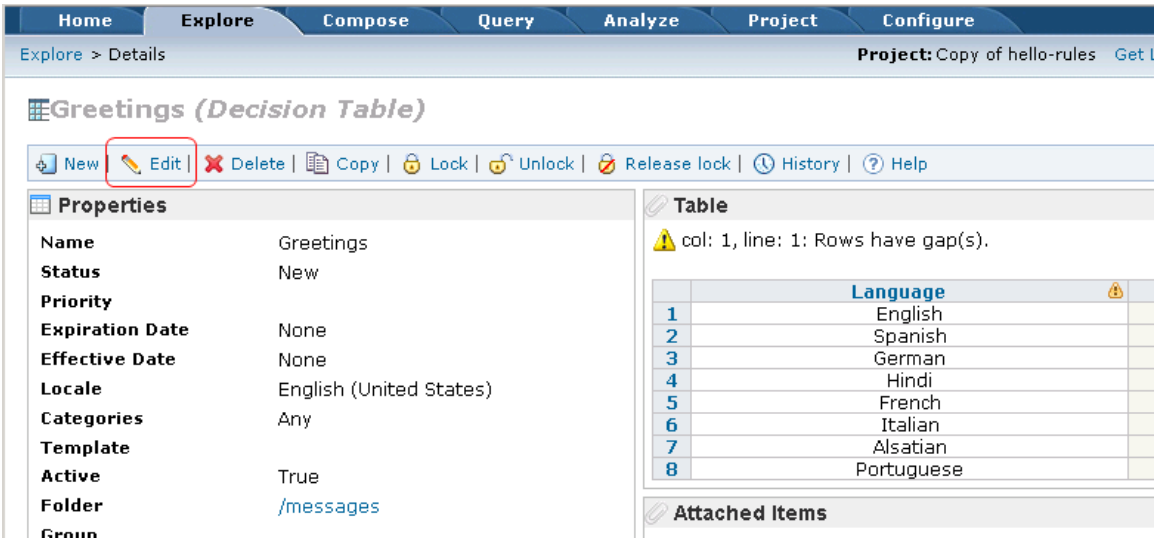
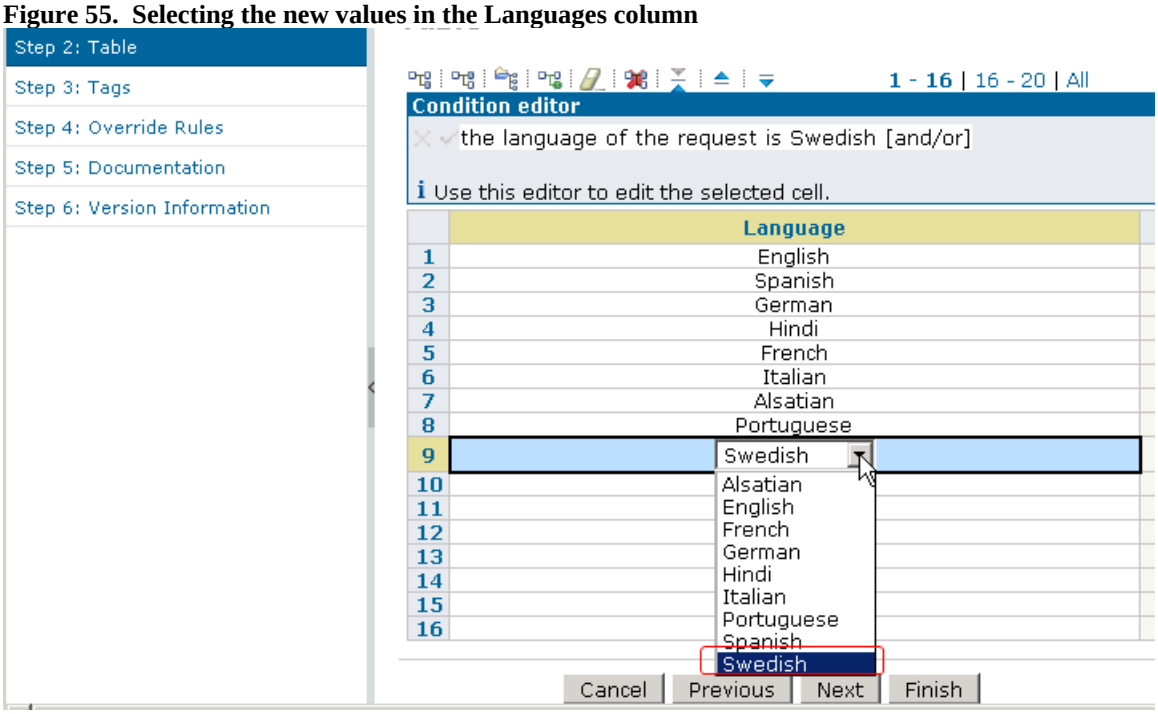


Figure 54. Editing the decision table



2. Select Step 2: Table and use the dropdown menu to reveal the refreshed list of predefined values for languages. In the Language column, select Swedish.

Figure 55. Selecting the new values in the Languages column



3. In the Greeting column, indicate the translated value to be displayed when Swedish is requested.
4. Click Finish and save your work.

Figure 56. Entering the translated values in the Greeting column

Table

1 - 16 | 16 - 20 | All

Action editor

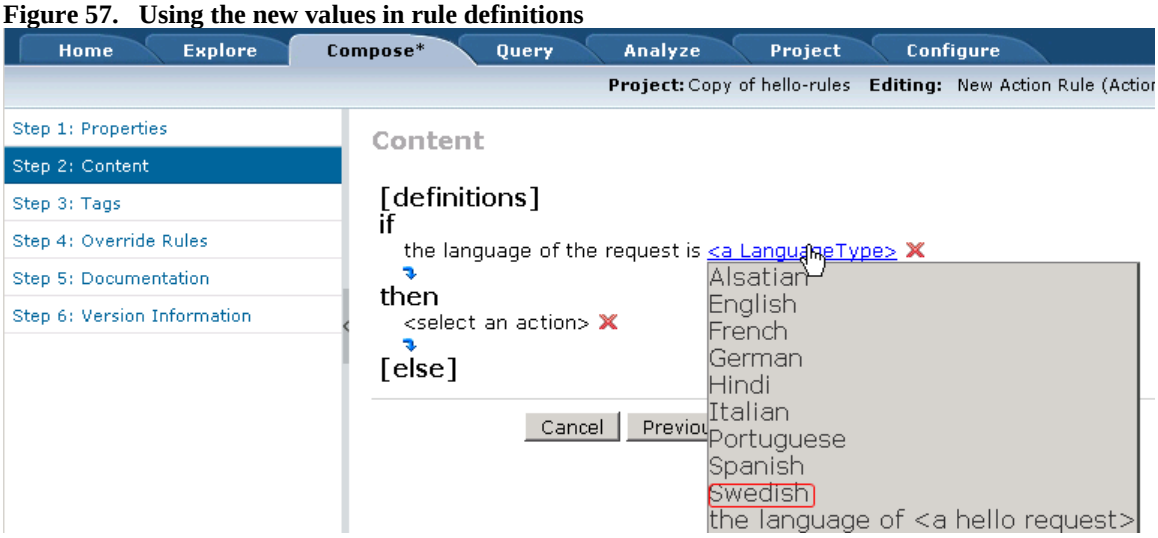
set the hello message of the response to Hej + " " + the name of the request [±]

Use this editor to edit the selected cell.

	Language	Greeting
1	English	Hello
2	Spanish	Hola
3	German	Hallo
4	Hindi	Namaste
5	French	Bonjour
6	Italian	Ciao
7	Alsatian	Salu
8	Portuguese	Ola
9	Swedish	Hej
10		

The new value is immediately available for use in rule authoring.

Figure 57. Using the new values in rule definitions



Conclusion

This article demonstrates the advantages of using dynamic domains in WebSphere Operational Decision Management V7.5. The use of a dynamic domain ensures a measure of error prevention during rule authoring and offers benefits over use of static enumerations. By using a simple Excel file as the underlying data source for a domain, rule developers can provide business users with the flexibility to modify the domain from within the Decision Center without the need for IT support. And because adding a new domain value does not result in changes to service contracts, developers do not need to redeploy the rule service or change the service clients.