Integrate Business Rules with Watson services on IBM Bluemix, Part 1: Build a Business Rules app that uses Personality Insights to match job applicants

Eugene Crujcov
Neil D. Delima

June 23, 2016

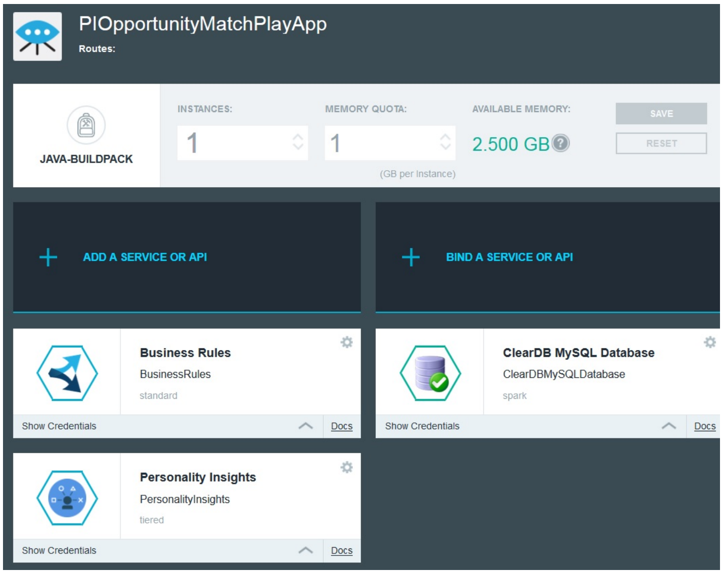
This series guides you through creating an application that uses the Business Rules service on IBM Bluemix and automates decisions based on the results of the Personality Insights service. Part 1 describes how you can use the Business Rules service to construct and deploy business rules on Bluemix, based on the Personality Insights data. You learn how to define a Business Rules data model from the Personality Insights data model. Finally, you learn how to deploy and test the Personality Insights-driven Business Rules project on Bluemix.

View more content in this series

The Watson Personality Insights service on IBM® Bluemix can help your business understand the personality traits of individuals that are part of a target audience. You can use the information to make informed decisions in various situations. For example, your human resources department might want to hire the most suitable candidate for a job, determined not just based on functional skills, but also personality traits.

The two-part Integrate Business Rules with Watson services on IBM Bluemix series shows how to use the Business Rules service to automate decisions based on the results of the Personality Insights service. You learn how to create a simple example of a job opportunity matching web application with Play Framework 2 and then deploy it on Bluemix. Play is a high-productivity Java™ and Scala web application framework that integrates components and APIs for modern web application development.

The following screen capture shows an overview of the components of an example job opportunity matching web application called **PIOpportunityMatchPlayApp** that is driven by the Personality Insights and Business Rules services in IBM Bluemix.



PIOpportunityMatchPlayApp
Routes:

JAVA-BUILDPACK

INSTANCES: 1
MEMORY QUOTA: 1 (GB per instance)
AVAILABLE MEMORY: 2.500 GB

Business Rules
BusinessRules
standard

ClearDB MySQL Database
ClearDBMySQLDatabase
spark

Personality Insights
PersonalityInsights
tiered

This Part 1 describes how you can construct business rules based on the Personality Insights data and then deploy them on Bluemix using the Business Rules service. You learn how to define a business rule data model from the Personality Insights data model and how to construct business rules from your model and then deploy them to Bluemix.

In Part 2, you develop a simple job opportunity matching web application using the Play Framework, and you deploy it to Bluemix. Part 2 demonstrates how to call the Bluemix cloud services and how to automate decisions based on Personality Insights data (using the business rules that you defined in Part 1).

The example application contains a basic web form for a job application and a list view of submitted applications. The purpose of the application is to flag potential candidates whose personality attributes meet a certain criteria associated with a job opportunity.

The Personality Insights service

To effectively engage your users, or customers, it is important to understand them, their needs, and their preferences. The Personality Insights service assists with this goal by providing a glimpse into an individual's personality characteristics, needs, and values. It performs linguistic analysis on various forms of digital communication from the individual, such as blog posts or email messages. The result is a JavaScript Object Notation (JSON) or a comma-separated values (CVS) file output.

According to the Personality Insights documentation, the output contains inferred personality characteristics from the following three personality models:

- The Big Five personality model (the most widely used model for generally describing how a person engages with the world), which measures agreeableness, conscientiousness, extraversion, emotional range (also referred to as neuroticism or natural reactions), and openness.

- The Needs personality model, which describes the aspects of a product resonate that with a person, including excitement, harmony, curiosity, ideal, closeness, self-expression, liberty, love, practicality, stability, challenge, and structure.
- The Values personality model (motivating factors that influence a person's decision making), which measures self-transcendence/helping others, conservation/tradition, hedonism/taking pleasure in life, self-enhancement/achieving success, and open to change/excitement.

To learn more about the personality models and the characteristics that they measure, read the Personality Insights documentation.

The Personality Insights service can be applied to a wide range of scenarios to provide a personalized and engaging application experience. Example applications include targeted marketing, matching up clients with sales personnel that have compatible personality characteristics, or making a movie recommendation to an individual. While the Personality Insights service provides insight into a personality characteristics, human intervention is still needed to make a decision or take some action based on the insight. The decision making is where the Bluemix Business Rules service can fit in. The Business Rules service makes it possible to automate decisions based on a user's personality.

The Business Rules service

Based on the IBM Operational Decision Manager product, the Business Rules service on Bluemix makes it possible to define business decision logic in a natural language form instead of hard-coding logic programmatically in your application. You craft business rules with business entities that are defined and verbalized in natural language in a business object model.

Use any of the following four categories of business rules:

- Action rules, which are written in natural language in a simple if-condition-then-action format.
- Decision tables, where decision conditions and actions are represented in the form of a spread sheet.
- Decision trees, which represent more complex branching logic, represented graphically as a tree.
- Technical rules, which capture more complex decisions represented in a higher level programming language called the ILog Rule Language.

Business rules are developed using an Eclipse-based integrated development environment called Rule Designer. Rules are contained in rule projects that have input and output parameters and associated business object models. Then, you use Rule Designer to package rules in execution units called Rule Apps to deploy them to the Business Rules service on the Bluemix runtime environment. Independent of the application code, you can update business policy changes and decisions that represented as business rules.

For more information, see the Getting Started with Business Rules documentation.

The steps in this tutorial describe how to create a Business Rules project, the Java execution object model (XOM), the business object model (BOM), and a sample business rule that uses the

BOM. The XOM mirrors the Personality Insights service, and the BOM represents the conceptual Personality Insights model (the Big Five, Needs, and Values personality models). You learn how to deploy your app on Bluemix, how to test it, and how to run it.

What you need to build your application

- Familiarity with IBM Operational Decision Manager and Rules Designer.
- An IBM Bluemix account. If you do not have an existing Bluemix account, start your free trial. Or, better yet, did you know that developerWorks Premium gives you a 12-month subscription to IBM Bluemix and 240 USD in cloud credits on Bluemix?
- Familiarity with the Business Rules service on Bluemix.
- Get the sample code from GitHub: https://github.com/nddelima/PIOpportunityMatchPlayApp

Get the code

1. Set up the Business Rules service on Bluemix

To set up a Business Rules service on Bluemix complete the following steps:

1. Log in to the Bluemix portal.
2. Go to **Catalog** > **Services** and click on **Business Rules**.
3. Leave default settings on, and create the service with the following default settings:
    - Space: **Dev**
    - App: **Leave Unbound**
    - Service name: **Business Rules-gp**
    - Selected Plan: **Standard**

2. Set up your development environment

To develop a business rules project and a rule application, you need to download and use the Rule Designer development tool. For instructions on how to install Rule Designer, how to create and deploy a rule project and a rule application, and how to bind and call the rules service from an application, see Getting Started with Business Rules .

3. Create the Personality Insights Java and business object models

After you install Rule Designer create your rule project, you develop your execution model and BOM so you can create rules. Because you are automating personality-based decisions, these models must be based on the output of the Personality Insights data model. The following code snippet shows the Personality Insights output JSON data model:

```
{
  "id": String,
  "source": String,
  "word_count": Integer,
  "processed_lang": String,
  "tree": {
    "id": "r",
    "name": "root",
    "children": [
      . . .
      {
        "id": String,
        "name": String,
        "category": String,
        "children": [
          {
            "id": String,
            "name": String,
            "category": String,
            "children": [
              {
                "id": String,
                "name": String,
                "category": String,
                "percentage": Float,
                "children": [
    ...
```

As you can see from the example, the Personality Insights output data model represents a nested tree containing attributes such as the ID, name, category, and children.

Now create a Java project that contains the XOM. Make sure that the Java classes that are part of the XOM resemble the Personality Insights output data model. The following code snippet shows the two main classes on the XOM and their contained attributes:

```
 public PersonalityAttribute tree;
 public String id;
 public String source;
 public int word_count;
 public String word_count_message;
 public String processed_lang;
    ....
}

public class PersonalityAttribute {
 public String id;
 public String name;
 public BigDecimal percentage;
 public Vector<PersonalityAttribute> children;
 public BigDecimal sampling_error;
 public String category;
    ...
}
```

With this approach, an application can call the Personality Insights business rule project (which you create in a future step) without needing to transform the output data from the Personality Insights service to a structure that the Personality Insights business rule application understands.

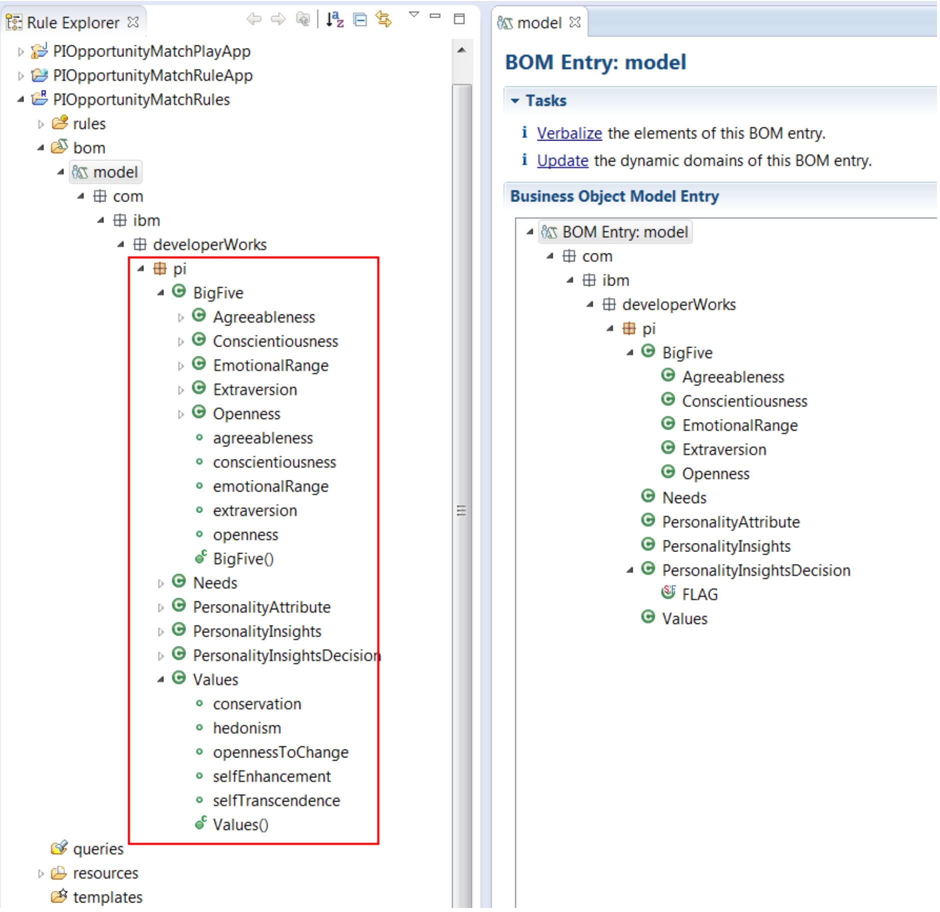
To author rules based on the Personality Insights data model, you must create another Business Rules project in Rule Designer. Create the default BOM and verbalize the model from the Java classes in the Java XOM. The classes that represent the Personality Insights data model are referenced by the Business Rules project and are in the XOM.

Follow the instructions in the Rule Designer for creating a Business Rules project and the BOM.

As mentioned earlier, the Java XOM represents the Personality Insights output data model, which is a nested tree containing attributes like ID, name, category and children. Creating business rules with a BOM that mimics the Java XOM is unnecessarily complicated. Instead, create a BOM representing the Big Five, Needs, and Values personality models. The object models map objects to the underlying Java Personality Insights data model. Authoring rules is easier and more comprehensible if you use the Big Five, Needs, and Values BOMs.

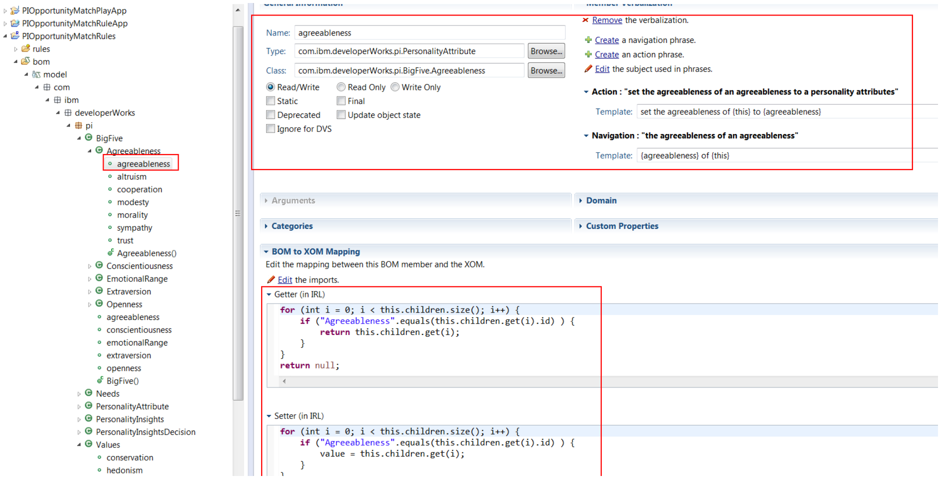
The sample code on Github includes a business rules project called `PIOpportunityMatchRules`. The project contains the definition of the Java XOM and the BOM, as previously described.

The following screen capture shows an example of the BOM of the rule project with BOM classes representing the personality models:



In the BOM, map the business classes that represent a personality model to the underlying Personality Insights data model object. Each object and attribute in the model is of the `PersonalityAttribute` type. The `PersonalityAttribute` class has the attribute that mimics the attributes of the child elements from the Personality Insights JSON object.

The following screen capture shows the BOM-to-XOM mapping of the `agreeableness` attribute of the `Agreeableness` class that belongs to the parent `BigFive` class.



Name: agreeableness

Type: com.ibm.developerWorks.pi.PersonalityAttribute Browse...

Class: com.ibm.developerWorks.pi.BigFive.Agreeableness Browse...

● Read/Write ○ Read Only ○ Write Only

□ Static □ Final

□ Deprecated □ Update object state

□ Ignore for DVS

Remove the verbalization.

Create a navigation phrase.

Create an action phrase.

Edit the subject used in phrases.

Action : "set the agreeableness of an agreeableness to a personality attributes"

Template: set the agreeableness of {this} to {agreeableness}

Navigation : "the agreeableness of an agreeableness"

Template: {agreeableness} of {this}

BOM to XOM Mapping

Edit the mapping between this BOM member and the XOM.

Edit the imports.

Getter (in IRL)

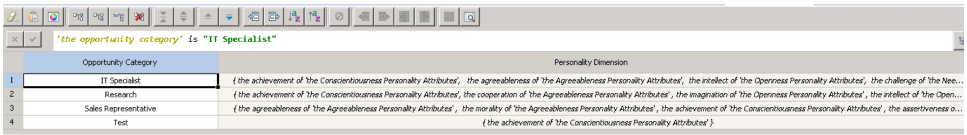
```
for (int i = 0; i < this.children.size(); i++) {
    if ("Agreeableness".equals(this.children.get(i).id) ) {
        return this.children.get(i);
    }
}
return null;
```

Setter (in IRL)

```
for (int i = 0; i < this.children.size(); i++) {
    if ("Agreeableness".equals(this.children.get(i).id) ) {
        value = this.children.get(i);
    }
```

4. Define business decision tables based on Personality Insights data

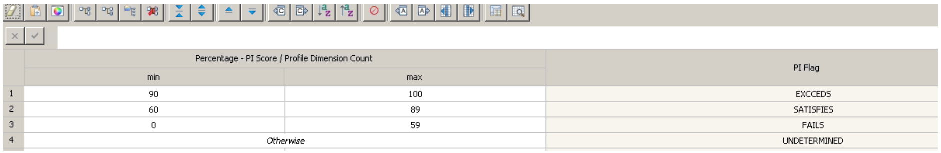
Now that you defined and verbalized the BOM, you can author rules using the BOM. In the sample application that you downloaded from Github, two decision tables are defined and included in a rule flow in the project. The first decision table, shown in the following screen capture, assigns a score by summing up the percentage values of the personality attributes associated with a specific job category.



'the opportunity category' is "IT Specialist"

	Opportunity Category	Personality Dimension
1	IT Specialist	{ the achievement of 'the Conscientiousness Personality Attributes', the agreeableness of 'the Agreeableness Personality Attributes', the intellect of 'the Openness Personality Attributes', the challenge of 'the Nee…
2	Research	{ the achievement of 'the Conscientiousness Personality Attributes', the cooperation of 'the Agreeableness Personality Attributes' , the imagination of 'the Openness Personality Attributes' , the intellect of 'the Open…
3	Sales Representative	{ the agreeableness of 'the Agreeableness Personality Attributes' , the morality of 'the Agreeableness Personality Attributes' , the achievement of 'the Conscientiousness Personality Attributes' , the assertiveness o…
4	Test	{ the achievement of 'the Conscientiousness Personality Attributes' }

The second decision table, shown in the following screen capture, then assigns a status flag of `EXCEEDS`, `SATISFIES`, `FAILS` or `UNDETERMINED` for the job category, depending on the personality score calculated in the previous decision table.



	Percentage - PI Score / Profile Dimension Count		PI Flag
	min	max	
1	90	100	EXCCEDS
2	60	89	SATISFIES
3	0	59	FAILS
4	*Otherwise*		UNDETERMINED

The advantage of using the Business Rules service is you can change the rules and then redeploy them, without redeploying the entire application (for example, when a new opportunity category is added, or when personality traits associated with an existing opportunity need to change).

The inputs to the rule project are the Personality Insights data and the job category. The outputs are the personality data with the computed score and the status flag. In the sample code, see the `sample-personality-insights-result.json` file, which represents the input to the business rules project, and the `sample-pi-business-rules-result.json` file, which represents the results of running the rules project against the previously described input.

5. Deploy the Personality Insights business rules project

After you create the business rules, you need to deploy the business rule project to the Business Rules runtime environment on Bluemix, called the Rule Execution Server. In Rule Designer, you first create a Rule Application, also known as a RuleApp. To create a RuleApp, see Authoring rules and creating a RuleApp.

Then, you create a Rule Execution Server configuration that points to the Rule Execution Server instance on Bluemix. For instructions, see Creating a Rule Execution Server configuration.

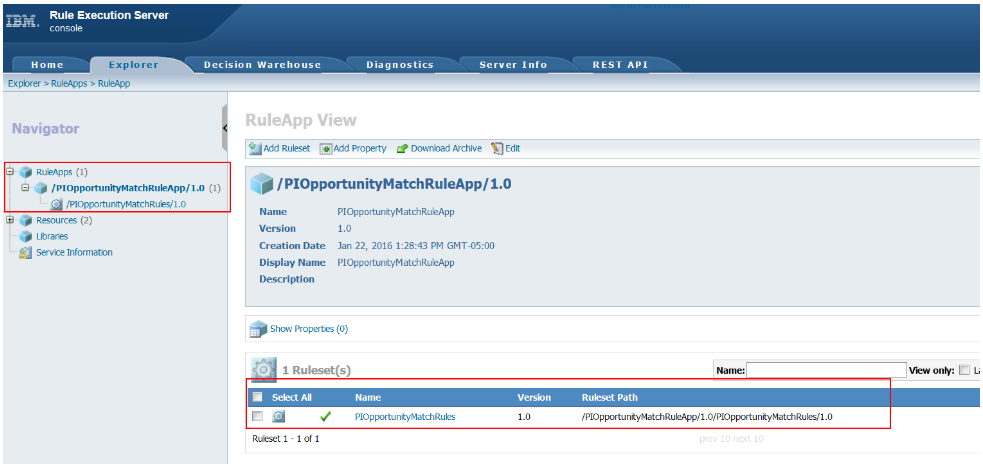
Finally, you deploy the RuleApp to the Rule Execution Server configuration. Follow the instructions at Deploying the RuleApp to the Business Rules service instance.

6. Review the deployed business rules app

View the Business Rules app that you deployed to the Business Rules service on Bluemix. In the Rule Execution Server runtime environment, navigate to the Business Rules service instance page and click **Open Console** on the Connection Settings tab. The Rule Execution Server console opens.

Log in to the Rule Execution Server with the credentials that are available on the Connection Settings tab of the Business Rules service instance page.

The following screen capture shows the deployed rules for the sample rule application.



/PIOpportunityMatchRuleApp/1.0

Name	PIOpportunityMatchRuleApp
Version	1.0
Creation Date	Jan 22, 2016 1:28:43 PM GMT-05:00
Display Name	PIOpportunityMatchRuleApp
Description	

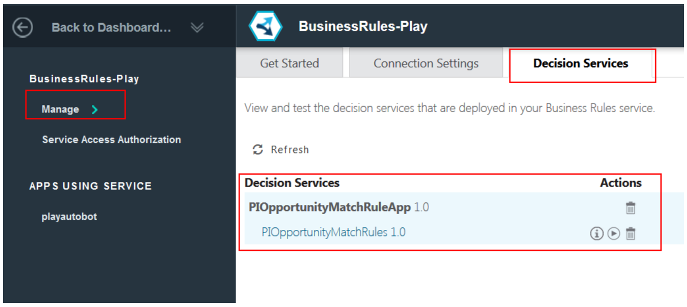
□ Select All		Name	Version	Ruleset Path
□	✓	PIOpportunityMatchRules	1.0	/PIOpportunityMatchRuleApp/1.0/PIOpportunityMatchRules/1.0

Ruleset 1 - 1 of 1

7. Test the deployed business rules app

Complete the following steps to test the deployed `PIOpportunityMatchRuleApp` business rules app from the Business Rules services tab of the Business Rules service on Bluemix:

1. From the RuleApp, click **Test Your Service** for the rule project, as shown in the following screen capture:



View and test the decision services that are deployed in your Business Rules service.

2. In the Decision Request text area, enter the JSON message payload that contains the JSON code returned from the Personality Insights service and the Job category. The following code snippet contains part of a sample Decision Request JSON message.

```
{ "PersonalityInsights" :
{
  "tree" : {
    "id" : "r",
    "name" : "root",
    "children" : [ {
      "id" : "personality",
      "name" : "Big 5 ",
      "children" : [ {
        "id" : "Openness_parent",
        "name" : "Openness",
        "percentage" : 0.9759973297659938,
        "children" : [ {
          "id" : "Openness",
          "name" : "Openness",
          "percentage" : 0.9759973297659938,
          "children" : [ {
...
    } ]
  },
  "id" : "*UNKNOWN*",
  "source" : "*UNKNOWN*",
  "word_count" : 1841,
  "word_count_message" : "There were 1,841 words in the text, we recommend text with at least 100 (and
 preferably 2,000) words",
  "processed_lang" : "en"
},
"category" : "Research"
}
```

A sample request file `sample-personality-insights-result.json` file is included with the sample code on Github.

3. Click **Run Test** and observe the decision response message. Notice that the following response message contains a status flag and a list of personality attributes, previously set in the business rule project when the rules for job category research ran.

```
{
    "__DecisionID__": "d5a6cb16-79c7-4ae7-93e1-3cabe8f166960",
    "decision": {
        "statusFlag": "SATISFIES",
        "score": 0.7023401221123601,
        "personalityAttributes": [
            {
                "id": "Achievement striving",
                "name": "Achievement striving",
                "percentage": 0.9380747767506684,
                "children": [],
                "sampling_error": 0,
                "category": "personality"
            },
            {
                "id": "Cooperation",
                "name": "Cooperation",
                "percentage": 0.821930730676822,
                "children": [],
                "sampling_error": 0,
                "category": "personality"
            },
            ...
        ]
    }
}
```

8. Determine the Personality Insights and Business Rules REST service endpoint that is used in your application

The value of the REST Execution API field contains the base path to the rule project. The complete path consists of the value of REST Execution API field concatenated to the path of the deployed rule project on the Rule Execution Server.

For the sample code use with this tutorial, the path is `https://Rule_Execution_Server_instance_name.ng.bluemix.net/DecisionService/rest/v1/PIOpportunityMatchRuleApp/1.0/PIOpportunityMatchRules/1.0.` The version numbers on the path are optional. When the version is not specified, the latest version of the deployed rules is picked up.

Conclusion

In this tutorial, you learned how to automate decisions based on the Personality Insights service by using the Business Rules Bluemix service. You examined the public API of the Personality Insights service and Business Rules service to find out how to call these services and to learn their input and output data structures. You created a Business Rules project and built a Business Object Model that you can use to process data from the Personality Insights service. Finally, you learned how to deploy and test the business rules project on Bluemix.

Now you know how to use the Personality Insights service to build an application driven by personality data, with flexible business logic from the Business Rules service. You can reuse the Java XOM and BOM and extend them to meet the needs of your application. You can use the extended BOM to write business rules that encompass the business logic. Then you can deploy and manage the business rules from the Rule Execution Server that runs on Bluemix.

In Part 2, you learn how to design and implement a web application that integrates with the Personality Insights and Business Rules services that are running on Bluemix.

Acknowledgements

The authors of this article would like to thank Paul Pacholski and Warren Leslie for their technical and editorial review and feedback.

Related topics

- The Business Rules service on IBM Bluemix
- The Personality Insights service on IBM Bluemix
- Getting Started with Business Rules
- Personality Insights service on IBM Watson Developer Cloud
- Authoring rules and creating a RuleApp
- Deploying the RuleApp to the Business Rules service instance
- Creating a Rule Execution Server configuration
- IBM Operational Decision Manager product documentation
- Watson Developer Cloud REST APIs