Red Hat JBoss Data Virtualization 6.2
Quick Starts Guide

David Sage

Red Hat JBoss Data Virtualization 6.2 Quick Starts Guide

David Sage
dlesage@redhat.com

Legal Notice

Abstract

These quick starts give an overview of product functionality and can be adapted for your own needs.

Table of Contents

PREFACE **3**

CHAPTER 1. INTRODUCTION AND GOALS OF THIS GUIDE **4**

CHAPTER 2. WHAT ARE THE QUICK STARTS? **5**

2.1. PREREQUISITES 5

2.2. GENERAL TIPS ON RUNNING QUICK STARTS 5

CHAPTER 3. CONCLUSION **13**

PREFACE

CHAPTER 1. INTRODUCTION AND GOALS OF THIS GUIDE

Congratulations! You now have quite a bit of experience with Red Hat JBoss Data Virtualization: you can perform a basic installation, run the software and use the Dashboard Builder to connect to data sources.

It is now time to showcase some more of the product’s features that you can utilise to build your own solutions going forward. These features are demonstrated in the various "quick starts" that come with the product.

CHAPTER 2. WHAT ARE THE QUICK STARTS?

The quick starts are sample projects. The purpose of these quick start samples is to illustrate various JBoss Data Virtualization features. You can find quick starts in EAP_HOME/quickstarts directory.

2.1. PREREQUISITES

Name	Description
Java Developer Kit	You must have a Java 8 JDK installed on your system.
Java Virtual Machine	Use the JVM installed with your JDK or use OpenJDK or the non-open source Azul Zing.
Maven	You must have Maven installed on your machine. For information about installing this, please refer to the Red Hat JBoss EAP documentation.
Red Hat JBoss Data Virtualization	You must have Data Virtualization installed and running. Refer to "Red Hat JBoss Data Virtualization Getting Started Guide" for details on how to do this.

2.2. GENERAL TIPS ON RUNNING QUICK STARTS

When you want to run a quickstart remember the following points:

1. Each quickstart has different requirements. These are documented in their individual README.md files.
2. If the server is running in a mode other than standalone mode, you can run only some of the quickstarts.

2.2.1. Simple Client Quick Start

The Simple Client quick start teaches you how to connect to JBoss Data Virtualization using either the JDBC driver or a data source. You can use the simple client quick start to test your SQL queries against any JBoss Data Virtualization instance with an active virtual database. Other quick start examples in this guide use this example as a means to test suggested queries. Refer to the individual README.md file for those quick starts in order to setup the environment correctly before executing a test query.

To run the quick start, go to the EAP_HOME/quickstarts/simpleclient directory and open the README.md file in a text editor and read the instructions it contains.

2.2.2. Data Federation Quick Start

The Data Federation quick start teaches you how Red Hat JBoss Data Virtualization can federate data from different sources into one virtual database.

The quick start uses three sample data sources to illustrate this concept, namely

1. Account Information (stored in an H2 database)
2. Market Data (stored in a CSV text file)
3. Other Personal Holdings, (stored in a Microsoft Excel file).

The virtual database to be deployed is called Portfolio. It defines a model for each data source; these will be named Accounts, MarketData and PersonalValuations, respectively. The VDB will be deployed to the Teiid Runtime, thereby making it accessible to the user application via JDBC.

To run the quick start, go to the EAP_HOME/quickstarts/dynamicvdb-datafederation directory and open the README.md file in a text editor and read the instructions it contains.

When you run the quick start’s test query, it joins data from the Accounts and MarketData models. Red Hat JBoss Data Virtualization then accesses both the relational and non-relational sources, calculates the portfolio values, and returns the results.

The dynamicvdb-datafederation/src/vdb/portfolio-vdb.xml file defines the resources that can be accessed by the client application:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb name="Portfolio" version="1">

    <description>The Portfolio Dynamic VDB</description>

    <!--
      Setting to use connector supplied metadata. Can be "true" or
"cached".
      "true" will obtain metadata once for every launch of Teiid.
      "cached" will save a file containing the metadata into
      the deploy/<vdb name>/<vdb version/META-INF directory
    -->
    <property name="UseConnectorMetadata" value="true" />


    <!--
      Each model represents a access to one or more sources.
      The name of the model will be used as a top level schema name
      for all of the metadata imported from the connector.

      NOTE: Multiple models, with different import settings, can be
bound to
      the same connector binding and will be treated as the same source
at
      runtime.
    -->
    <model name="MarketData">
```

```
        <!--
            Each source represents a translator and data source. There
are
            pre-defined translators, or you can create one.
        -->
        <source name="text-connector" translator-name="file"
connection-jndi-name="java:/marketdata-file"/>
    </model>

    <model name="Accounts">
        <!--
          JDBC Import settings

          importer.useFullSchemaName directs the importer to drop the
source
          schema from the Teiid object name, so that the Teiid fully
qualified name
          will be in the form of <model name>.<table name>
        -->
        <property name="importer.useFullSchemaName" value="false"/>

         <!--

            This connector is defined to reference the H2 localDS"

          -->

        <source name="h2-connector" translator-name="h2" connection-
jndi-name="java:/accounts-ds"/>

    </model>

    <model name="PersonalValuations">
        <property name="importer.headerRowNumber" value="1"/>
        <property name="importer.ExcelFileName"
value="otherholdings.xls"/>
        <source name="excelconnector" translator-name="excel"
connection-jndi-name="java:/excel-file"/>
        <metadata type="DDL"><![CDATA[

        SET NAMESPACE 'http://www.teiid.org/translator/excel/2014' AS
teiid_excel;

        CREATE FOREIGN TABLE Sheet1 (
        ROW_ID integer OPTIONS (SEARCHABLE 'All_Except_Like',
"teiid_excel:CELL_NUMBER" 'ROW_ID'),
        ACCOUNT_ID integer OPTIONS (SEARCHABLE 'Unsearchable',
"teiid_excel:CELL_NUMBER" '1'),
        PRODUCT_TYPE string OPTIONS (SEARCHABLE 'Unsearchable',
"teiid_excel:CELL_NUMBER" '2'),
        PRODUCT_VALUE string OPTIONS (SEARCHABLE 'Unsearchable',
"teiid_excel:CELL_NUMBER" '3'),
        CONSTRAINT PK0 PRIMARY KEY(ROW_ID)
        ) OPTIONS ("teiid_excel:FILE" 'otherholdings.xls',
"teiid_excel:FIRST_DATA_ROW_NUMBER" '2');
```

```
          ]> </metadata>
    </model>

    <model name="Stocks" type="VIRTUAL">
        <metadata type="DDL"><![CDATA[

        CREATE VIEW StockPrices (
            symbol string,
            price bigdecimal
            )
            AS
               SELECT SP.symbol, SP.price
                FROM (EXEC MarketData.getTextFiles('*.txt')) AS f,
                    TEXTTABLE(f.file COLUMNS symbol string, price
bigdecimal HEADER) AS SP;


        CREATE VIEW Stock (
            product_id integer,
            symbol string,
            price bigdecimal,
            company_name   varchar(256)
            )
            AS
                SELECT  A.ID, S.symbol, S.price, A.COMPANY_NAME
                    FROM StockPrices AS S, Accounts.PRODUCT AS A
                    WHERE S.symbol = A.SYMBOL;


        ]> </metadata>
    </model>

    <model name="StocksMatModel" type="VIRTUAL">
    <metadata type="DDL"><![CDATA[
    CREATE view stockPricesMatView
        (
            product_id integer,
            symbol string,
            price bigdecimal,
            company_name   varchar(256)
        ) OPTIONS (MATERIALIZED 'TRUE', UPDATABLE 'TRUE',
               MATERIALIZED_TABLE 'Accounts.h2_stock_mat',
               "teiid_rel:MATVIEW_TTL" 120000,
               "teiid_rel:MATVIEW_BEFORE_LOAD_SCRIPT" 'execute
accounts.native(''truncate table h2_stock_mat'');',

                "teiid_rel:MATVIEW_AFTER_LOAD_SCRIPT" 'execute
pg.native(''ALTER TABLE mat_actor RENAME TO mat_actor_temp'');execute
pg.native(''ALTER TABLE mat_actor_staging RENAME TO
mat_actor'');execute pg.native(''ALTER TABLE mat_actor_temp RENAME TO
mat_actor_staging;'')',

               "teiid_rel:ON_VDB_DROP_SCRIPT" 'DELETE FROM
Accounts.status WHERE Name=''stock'' AND schemaname = ''Stocks''',
               "teiid_rel:MATERIALIZED_STAGE_TABLE"
'Accounts.h2_stock_mat',
                "teiid_rel:ALLOW_MATVIEW_MANAGEMENT" 'true',
                "teiid_rel:MATVIEW_STATUS_TABLE" 'status',
                "teiid_rel:MATVIEW_SHARE_SCOPE" 'NONE',
                "teiid_rel:MATVIEW_ONERROR_ACTION" 'THROW_EXCEPTION')
         AS SELECT  A.ID, S.symbol, S.price, A.COMPANY_NAME
                    FROM Stocks.StockPrices AS S, Accounts.PRODUCT AS
A
                    WHERE S.symbol = A.SYMBOL;
    ]>
       </metadata>
    </model>

</vdb>
```

In this model, the "vdb" element defines the virtual database that has a name of "Portfolio" with version "1" A "model" element represents a schema that is being integrated. This sample defines three sources. The "source" element inside the "model" element defines the "name" of the source (can be any name), the name of the translator (defines the type of the source like oracle, db2, mysql, h2, file or WS) and the "connection-jndi-name" defines the source’s JNDI name in the Red Hat JBoss EAP container.

Also note that inside the "model" elements, some "property" elements can be used to define how metadata is to be imported from the source. When you configure the quick start, the portfolio-vdb.xml is deployed to the JBoss EAP server. The portfolio-vdb.xml.dodeploy file (also referred to as a marker file, which is empty) is also deployed so that the Teiid Runtime is triggered to load the virtual database. The VDB’s data sources will then be validated for availability, and if successful, the VDB status will be set to "Active".

If the data sources are created and the VDB is deployed correctly, then you will see an information message in the JBoss EAP console that states that the VDB has been deployed and is in an "active" state. If there are any errors then you will see that the VDB was not deployed or is currently in an "inactive" state. Find the issues via the console and fix them before you proceed any further.

There are three virtual models defined in the dynamic vdb; Stocks, StocksMatModel and OtherHoldings. The Stocks virtual model contains 2 basic views. One, Stocks, to access the stock prices in the text data source and the second, StockPrices, that will join the Stocks with Products to federate the data from both data sources. The OtherHoldings contains a view that joins Acccounts with OtherHoldings to show who the holdings belongs to.

The StocksMatModel contains a similar view to the StockPrices, but its defined as a materialized view. This is an example to demonstrate the use case. If a query is slow when federating across multiple data sources, you can materialize the view to improve performance. The materialized view is configured to update every two minutes. Hence, if you execute the following query to add a new product and then wait two minutes to allow the materialization view to be updated, you can query the view and see that the results will contain a new row: INSERT INTO PRODUCT (ID,SYMBOL,COMPANY_NAME) VALUES(2000,'RHT','Red Hat Inc')

This examples uses three data sources, a relational database, a CSV text file and an Excel file. You can substitute any other relational database for H2, as long as you have a suitable JDBC driver. The schema file provided at dynamicvdb-datafederation/src/teiidfiles/customer-schema.sql, is specific to H2 but you can convert it for use with other databases

In order to use a text file as the source, we need a data file which defines the data in a CSV format (comma separated values). Each data file contains column information for the table. The column information is typically defined on line 1, as header line in the file, and all the following lines contain the actual rows of data. Each line corresponds to single row. A portion of the sample file is shown below. The complete sample file is dynamicvdb-datafederation/src/teiidfiles/data/marketdata-price.txt.

```
SYMBOL,PRICE
IBM,83.46
RHT,11.84
BA, 44.58
ORCL,17.37
```

The Excel file contains sheet1, that defines a header row and subsequent data rows, similar to a text file. This example will demonstrate that excel files can be used on different operating systems, not just windows. You can use the provided data files or create your own data files. When performing the set up step, data sources will be configured.

2.2.3. Data Roles Quick Start

The dynamicvdb-dataroles quick start demonstrates how you can use data roles to control access to data. (This includes read-only and read-write access.)

In this quick start, the virtual database (portfolio-vdb.xml) has these two data access roles:

1. read-only - this role provides only read-only access (in other words, selects). This role is given to everybody who has a login credential. (Use the user called "user" with password "user".)

2. read-write - this role give reads access, in addtion write access (insert/update/delete). This access is given only to users with the "superuser" JAAS role. (Use the user called "portfolio" with password "portfolio".)

To run the quick start, go to the EAP_HOME/quickstarts/dynamicvdb-dataroles directory and open the README.md file in a text editor and read the instructions it contains. Look at the application-roles.properties and application-users.properties, because you have to use the ./add-user.sh (bat) scripts (as indicated in the README).

The quick start shows use of the read-write data-role in the portfolio-vdb.xml file:

```
  <data-role name="ReadWrite">
     <description>Allow Reads and Writes to tables and
procedures</description>

     <permission>
         <resource-name>Accounts</resource-name>
         <allow-create>false</allow-create>
         <allow-read>true</allow-read>
         <allow-update>true</allow-update>
     </permission>

     <permission>
         <resource-name>MarketData</resource-name>
         <allow-create>false</allow-create>
         <allow-read>true</allow-read>
         <allow-update>true</allow-update>
     </permission>

     <!--
         This role must defined in the JAAS security domain, the sample
UserRolesLoginModules based roles file provided
```

```
         in this sample directory. copy these "teiid-security-
roles.properties" and "teiid-security-users.proeprties"
         into "<jboss-install>/modules/org/jboss/teiid/conf" directory
and replace the old ones.
     -->
     <mapped-role-name>supervisor</mapped-role-name>
 </data-role>
```

To see how the users and roles were defined for JAAS, see the src/security/teiid-security-roles.properties and src/security/teiid-security-users.properties files. JAAS uses the teiid-security-users.properties file file JAAS to determine user credentials and the teiid-security-roles.properties file maps the username to the mapped-role-name element.

2.2.4. Web Service Data Source Quick Start

The Web-Services-as-a-Data-Source quick start demonstrates how to use the Web Service Translator to call a web service that returns an XML document. The quick start also demonstrates how to define a transformation using XMLTABLE and XMLPARSE to translate the XML document into a relational database.

To run the quick start, go to the EAP_HOME/quickstarts/webservices-as-a-datasource directory and open the README.md file in a text editor and read the instructions it contains.

When the quick start is configued, the CustomerRESTWebSvc.war is deployed. This represents the web service data source that JBoss Data Virtualization will access. (It was designed this way so that no external resource is required in order to execute the quick start.) The dynamic VDB, namely webservice-vdb.xml, defines the VIEW using DDL, in order to read and transform the XML document returned from the web service.

2.2.5. JBoss Data Grid Running in Library Mode as a Data Source Quick Start

The JBoss-Data-Grid-Running-in-Library-Mode-as-a-Data-Source quick start demonstrates how to access a JBoss Data Grid cache running in library mode using the infinispan-cache translator. (JBoss Data Grid is a distributed cache for clusters.)

To run the quick start, go to the EAP_HOME/quickstarts/jdg-local-cache directory and open the README.md file in a text editor and read the instructions it contains.

This quick start shows you how to do the following:

1. Define the source model to the object cache using DDL metadata.
2. Configure a translator override to enable SupportsLuceneSearching on the cache..
3. Define the translator type of "infinispan-cache" used to read the cache.
4. An example web application is used to define the cache container and preload its contents.
5. Configure the resource adapter for connecting to the JDG Cache as a data source, for which the translator is mapped to via the connection-jndi-name.
6. Demonstrate the ability to do reads and writes (Insert, Update and Delete) to the cache. Example queries are provided.

2.2.6. JBoss Data Grid Remote Cache as a Data Source Quick Start

The JBoss-Data-Grid-Running-in-Library-Mode-as-a-Data-Source example demonstrates how to access JBoss Data Grid cache running in library mode using the infinispan-cache translator.

To run the quick start, go to the EAP_HOME/quickstarts/jdg-remote-cache directory and open the README.md file in a text editor and read the instructions it contains.

The source model metadata will be created dynamically based on the configured resource adapter. Notice there is no DDL is being used, as in the local cache quick start. This quick start is defaulting to the dynamically created metadata. However, you can define your own metadata, as in the local cache quick start and provide any necessary overrides.

This example shows you how to do the following:

1. Define the source model to the object cache by specifying the translator type of "infinispan-cache-dsl".
2. Set up the Infinispan-dsl resource-adapter in order to communicate to the remote cache
3. Configure the JDG Cache as a data source, for which the translator is mapped to via the connection-jndi-name. It also demonstrates the ability to do reads and writes (Insert, Update and Delete) to the cache. Example queries are provided, but the actual content in the DML may not match the content you loaded in the cache.

2.2.7. Use Hibernate with Red Hat JBoss Data Virtualization Quick Start

When you utilize Hibernate, it is normally a one-object-to-one-data-source mapping. By using JBoss Data Virtualization as the data source, you approach the integration at the data layer rather than application layer. This makes it easier to join related information and expose it through hibernate, rather than merge the related data at the application layer.

The Hibernate-on-top-of-teiid quick start demonstrates how hibernate can take advantage of multiple data sources through a single Java object by using the data federation capabilities of JBoss Data Virtualization. This quickstart extends the Portfolio VDB and creates a view that is mapped to a single relationally mapped object in hibernate.

To run the quick start, go to the EAP_HOME/quickstarts/hibernate-on-top-of-teiid directory and open the README.md file in a text editor and read the instructions it contains.

CHAPTER 3. CONCLUSION

During this hour, you have been taken on a tour of Red Hat JBoss Data Virtualization’s various features. You have run various quick starts that demonstrate the product’s functionality and teach you a little about how it works.

You can now go forward and modify these quick start to build your own solutions to suit your business needs. Use the pre-packaged quick starts as templates to guide you and refer to the full documentation suite to learn more about the product’s capabilities.