rfd4j_server_documentation

Greg Austin

July 2019

Contents

1 Big Picture **2**

2 RDF4J Container: Tomcat 9.0 **3**

2.1 Tomcat Installation 3

2.2 Starting / Stopping The Tomcat Server 3

2.3 Deploying RDF4J 3

3 RDF4J Web Interface **4**

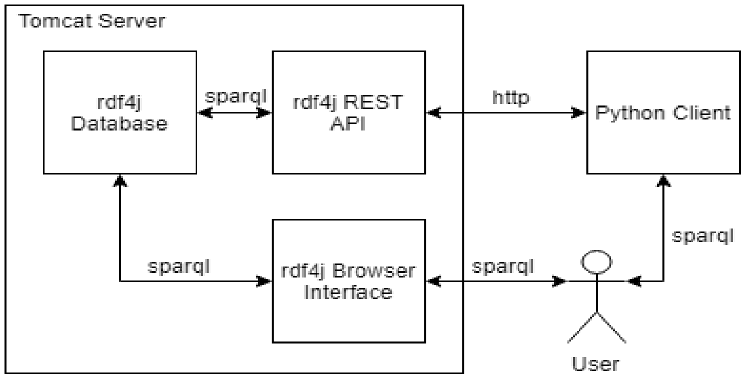
4 RDF Graph Description **5**

5 SPARQL Queries **6**

6 REST API **6**

7 Python Client **6**

1 Big Picture



- User: The user can be a person or program (like grasshopper) that wants to either create, read, update, or destroy data in the database. To do this, the user issues commands to the database in the **sparql** query language via the browser interface or using a python client program.

- Tomcat Server: A Tomcat server was used to deploy the rdf4j web app for this prototype. There are other options for deploying java web apps, but Tomcat is (as far as I know) the industry default for java web apps.

- RDF4J REST API: The rdf4j web app allows clients to use http GET and POST requests to access the database. To do this, a client program has to encode a sparql command into the parameters of the http request to the server. The REST API defines how and what to encode into the http request. The API also defines where to send such requests.

- RDF4J Browser Interface: The browser interface is another option for issuing sparql commands to the database. A user can navigate to the interface using a standard web browser and interact with rdf4j graphically.

- Python Client: For testing purposes, a few simple client programs were written in Python that would create, read, update, and delete data from the database. The clients use the REST API to send the sparql commands to the database.

- RDF4J Database: The database stores data as subject-predicate-object triples. These triples are used to describe information in the form of the nodes and edges of a graph. Sparql commands are used to create the graphs by adding and removing triples as well as searching for information by filtering the triples that are returned to the user.

2 RDF4J Container: Tomcat 9.0

2.1 Tomcat Installation

The instalation and setup guide for tomcat can be found at `https://tomcat.apache.org/tomcat-9.0-doc/setup.html`. The guide is good, but there is one important thing I missed when installing the server. The tomcat files need to be saved on the profile of the used that will be managing it. Otherwise it will not be able to deploy web apps from that user's files. Otherwise follow the setup instructions and me sure that there is an admin id and password for the server.

2.2 Starting / Stopping The Tomcat Server

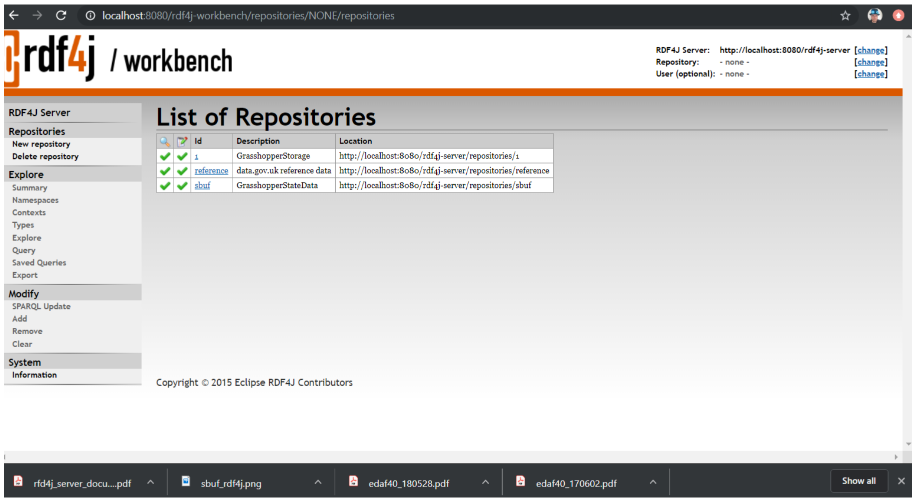
- Start a Command Prompt from the Start menu
- Navigate to the Tomcat bin directory, e.g., c:/Tomcat8/bin
- Type in startup and then hit Enter to execute the Tomcat server start up script
- A separate window will open and a series of messages will appear, followed by the message indicating the server is started
- To stop the Tomcat server, type in shutdown and then hit Enter in the original command prompt

2.3 Deploying RDF4J

Download and unpack the full rdf4j SDK from `https://rdf4j.eclipse.org/download/` and find the files called rdf4j-server.war and rdf4j-workbench.war. The file path should be `/eclipse-rdf4j-2.5.2-sdk/eclipse-rdf4j-2.5.2/war/`. Start the Tomcat server. Open a web browser and navigate to the server browser interface at http://localhost:8080. From there navigate to the manager app (there is a button in the upper right corner of the page). Login using the admin id and password that was made when setting up the server. Once logged in to the manager app, find the "Deploy" section and use the subsection called "WAR file to deploy" to upload both rdf4j-server.war and rdf4j-workbench.war. The rdf4j server app is now deployed and should be available.

3 RDF4J Web Interface

If the rdf4j web app is deployed and functioning, the web interface should be accessable at `http://localhost:8080/rdf4j-workbench/repositories/NONE/repositories`.



List of Repositories

🔍	📝	Id	Description	Location
✓	✓	1	GrasshopperStorage	http://localhost:8080/rdf4j-server/repositories/1
✓	✓	reference	data.gov.uk reference data	http://localhost:8080/rdf4j-server/repositories/reference
✓	✓	sbuf	GrasshopperStateData	http://localhost:8080/rdf4j-server/repositories/sbuf

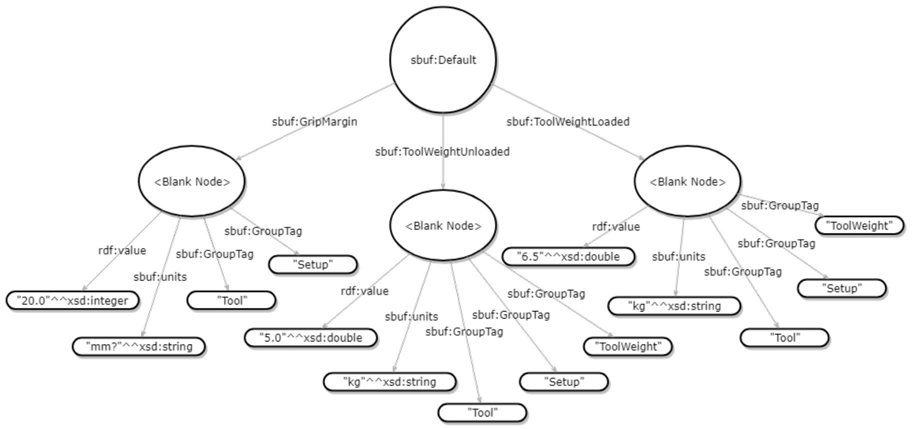
Copyright © 2015 Eclipse RDF4J Contributors

The important things to note here are the "New Repository", "Query", and "SPARQL Update" tabs.

- New Repository: Make a new repository. The important thing is to give it the id "sbuf". All of the example python clients assume "sbuf" as a repository id.
- SPARQL Update: This tab is used to execute sparql statements that alter the data in the database. If you copy and paste the text from database_initialization.rq in this tab, you can create a database that contains all of the initial data in the grasshopper program.
- Query: from this tab, you can execute sparql queries on the database/repository you're currently working with. You can see which repository is selected in the upper right corner of the page. There are some standard queries in the file std_queries.rq.

4 RDF Graph Description

The current idea for the data structure is to save all grasshopper configuration information into a graph called "¡sbuf:GrasshopperConfigurations¿". In this graph, each configuration will get a root node. For example, consider the grasshopper values in the box marked "Tool". Below is a representation of how the default values are stored in the database (the default values are the numbers that were in grasshopper on the particular day I chose to write them down).



Each edge from the root configuration node (sbuf:Default) is the name of one of the grasshopper parameters. These edges point to blank nodes which in turn have several named edges pointing to the data associated with the parameter. For instance, the parameter "sbuf:GripMargin" has a value, unit, and two group tags associated with it. That is to say, in the default configuration of grasshopper, the "Grip Margin" is 20mm and is associated with the grouping "Setup" and "Tool". The file called database_initialization.rq has a complete initialization script for the database. The easiest way to initialize the database is to copy and paste the text in this file to the SPARQL update tab in the rdf4j workbench (the browser interface) and then execute it.

5 SPARQL Queries

I've taken the liberty of including a file of example sparql statements called "std_queries.rq". These 'standard queries' are not complete, but they cover examples of the functionality that you will probably need.

6 REST API

The actual documentation for the REST API can be found at `https://rdf4j.eclipse.org/documentation/rest-api/` but here's almost everything I needed to know:

- If you want to read from the database, send a SELECT statement to the base repository url `http://localhost:8080/rdf4j-server/repositories/sbuf`.
- if you want to add change or delete anything in the database, send a DELETE or INSERT statement to the statements url `http://localhost:8080/rdf4j-server/repositories/sbuf/statements`

7 Python Client

I've also included three example python scripts that read, write, and update the database. Hopefully these scripts will be close enough to what you want to do with them that you can copy and paste them as a base. The most useful one is script called sbuf_increment_counter.py. Because SPARQL does not have an update command, to alter existing data in the database you have to delete the old data and insert the new data. Incrementing the counter and checking that the value was updated requires reading, writing and deleting data in the database from the python script. That is to say, it has example of all types of interactions that would be needed. The SPARQL statements in "std_queries.rq" can be coppied and pasted into the python scripts to run them, but that can cause problems with the scripts reading the return data. It might be easier to just check the changes in the workbench instead of trying to reqrite all of the python scripts.