CREATING A LOCAL REPOSITORY FOR GRAPHDB WITH RDF4J PROGRAMMATICALLY

HENRIETTE HARMSE

If you want to create a local repository for Ontotext GraphDB, according to the documentation. The are essentially 3 steps:

1. Create a configuration file.
2. Create a `pom.xml` file.
3. The Java code.

However, there are reasons why you may not want to do this, which I detail.

1. Configuration File

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rep: <http://www.openrdf.org/config/repository#>.
@prefix sr: <http://www.openrdf.org/config/repository/sail#>.
@prefix sail: <http://www.openrdf.org/config/sail#>.
@prefix owlim: <http://www.ontotext.com/trree/owlim#>.

[] a rep:Repository ;
  rep:repositoryID "graphdb-repo" ;
  rdfs:label "graphdb-repo-label" ;
  rep:repositoryImpl [
    rep:repositoryType "graphdb:FreeSailRepository" ;
    rep:repositoryType "owlim:MonitorRepository" ;
    sr:sailImpl [
      sail:sailType "graphdb:FreeSail" ;

      owlim:base-URL "http://myexample.ontotext.com/graphdb#" ;
      owlim:defaultNS "" ;
      owlim:entity-index-size "10000000" ;
      owlim:entity-id-size  "32" ;
      owlim:imports "" ;
      owlim:repository-type "file-repository" ;
      owlim:ruleset "owl-horst-optimized" ;
      owlim:storage-folder "storage" ;

      owlim:enable-context-index "true" ;
      owlim:cache-memory "256m" ;
      owlim:tuple-index-memory "224m" ;

      owlim:enablePredicateList "true" ;
      owlim:predicate-memory "32m" ;

      owlim:fts-memory "0" ;
```

Date: 9th April 2018.

```
      owlim:ftsIndexPolicy "never" ;
      owlim:ftsLiteralsOnly "true" ;

      owlim:in-memory-literal-properties "true" ;
      owlim:enable-literal-index "true" ;
      owlim:index-compression-ratio "-1" ;

      owlim:check-for-inconsistencies "false" ;
      owlim:disable-sameAs "false" ;
      owlim:enable-optimization "true" ;
      owlim:transaction-mode "safe" ;
      owlim:transaction-isolation "true" ;
      owlim:query-timeout "0" ;
      owlim:query-limit-results "0" ;
      owlim:throw-QueryEvaluationException-on-timeout "false" ;
      owlim:useShutdownHooks "true" ;
      owlim:read-only "false" ;
    ]
  ].
```

2. pom.xml File

In the `pom.xml` file you need only specify a dependency on Ontotext GraphDB, which will add the required RDF4J dependencies:

```
<dependency>
   <groupId>com.ontotext.graphdb</groupId>
   <artifactId>graphdb-free-runtime</artifactId>
   <version>8.4.1</version>
</dependency>
```

3. Java Code

```
package org.graphdb.rdf4j.tutorial;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.file.Path;
import java.nio.file.Paths;

import org.eclipse.rdf4j.model.Resource;
import org.eclipse.rdf4j.model.impl.TreeModel;
import org.eclipse.rdf4j.model.util.Models;
import org.eclipse.rdf4j.model.vocabulary.RDF;
import org.eclipse.rdf4j.repository.Repository;
import org.eclipse.rdf4j.repository.RepositoryConnection;
import org.eclipse.rdf4j.repository.config.RepositoryConfig;
import org.eclipse.rdf4j.repository.config.RepositoryConfigSchema;
import org.eclipse.rdf4j.repository.manager.LocalRepositoryManager;
import org.eclipse.rdf4j.repository.manager.RepositoryManager;
import org.eclipse.rdf4j.rio.RDFFormat;
import org.eclipse.rdf4j.rio.RDFParser;
```

```
import org.eclipse.rdf4j.rio.Rio;
import org.eclipse.rdf4j.rio.helpers.StatementCollector;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.Marker;
import org.slf4j.MarkerFactory;

public class CreateLocalRepository {
  private static Logger logger = LoggerFactory.getLogger(CreateLocalRepository.class);
  // Why This Failure marker
  private static final Marker WTF_MARKER = MarkerFactory.getMarker("WTF");

  public static void main(String[] args) {
    try {
      Path path = Paths.get(".").toAbsolutePath().normalize();
      String strRepositoryConfig = path.toFile().getAbsolutePath() +
          "/src/main/resources/repo-defaults.ttl";

      // Instantiate a local repository manager and initialize it
      RepositoryManager repositoryManager = new LocalRepositoryManager(new File("."));
      repositoryManager.initialize();

      // Instantiate a repository graph model
      TreeModel graph = new TreeModel();

      // Read repository configuration file
      InputStream config = new FileInputStream(strRepositoryConfig);
      RDFParser rdfParser = Rio.createParser(RDFFormat.TURTLE);
      rdfParser.setRDFHandler(new StatementCollector(graph));
      rdfParser.parse(config, RepositoryConfigSchema.NAMESPACE);
      config.close();

      // Retrieve the repository node as a resource
      Resource repositoryNode =  Models.subject(graph
         .filter(null, RDF.TYPE, RepositoryConfigSchema.REPOSITORY))
         .orElseThrow(() -> new RuntimeException(
             "Oops, no <http://www.openrdf.org/config/repository#> subject found!"));

      // Create a repository configuration object and add it to the repositoryManager
      RepositoryConfig repositoryConfig = RepositoryConfig.create(graph, repositoryNode);
      repositoryManager.addRepositoryConfig(repositoryConfig);

      // Get the repository from repository manager, note the repository id
      // set in configuration .ttl file
      Repository repository = repositoryManager.getRepository("graphdb-repo");

      // Open a connection to this repository
      RepositoryConnection repositoryConnection = repository.getConnection();

      // ... use the repository

      // Shutdown connection, repository and manager
      repositoryConnection.close();
```

```
      repository.shutDown();
      repositoryManager.shutDown();
    } catch (Throwable t) {
      logger.error(WTF_MARKER, t.getMessage(), t);
    }
  }
}
```

4. Why you may not want to do this

`new LocalRepositoryManager(new File("."));` will create a repository where ever your Java application is running from. This means the repository will not be under the control of your Ontotext GraphDB Workbench. Hence, you will not be able to run SPARQL queries or monitor your database from the Workbench. I am not aware of any way via which you can instruct GraphDB to look for repositories in an additional directory.

If you change the directory to `$GRAPH_DB_INSTALL$/data/repositories`, the repository will be under the control of Ontotext GraphDB (assuming you have a local GraphDB instance) only if GraphDB is not running. If you start GraphDB after running your program, you will be able to see the repository in GraphDB workbench.

5. Conclusion

In this post I have detailed how you can create an Ontext GraphDB repository using RDF4J and why you may not want to do this. In my next post I detail how to create a remote repository, which addresses the problem I detailed here. You can find the complete code of this example on github.

References