CREATING A REMOTE REPOSITORY FOR GRAPHDB WITH RDF4J PROGRAMMATICALLY

HENRIETTE HARMSE

In my previous post I have detailed how you can create a local Ontotext GraphDB repository using RDF4J. I indicated that there are some problems when creating a local repository. Therefore, in this post I will detail how to create a remote Ontotext GraphDB repository using RDF4J. As with creating a local repository, there are three steps:

1. Create a configuration file, which is as for local repositories.
2. Create `pom.xml` file, which is as for local repositories.
3. Create the Java code.

The benefit of creating a remote repository is that it will be under the control of the Ontotext GraphDB Workbench. Hence, you will be able to monitor your repository from the Workbench.

1. Java Code

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

import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Iterator;

import org.eclipse.rdf4j.model.Model;
import org.eclipse.rdf4j.model.Resource;
import org.eclipse.rdf4j.model.Statement;
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.http.config.HTTPRepositoryConfig;
import org.eclipse.rdf4j.repository.manager.RemoteRepositoryManager;
import org.eclipse.rdf4j.repository.manager.RepositoryManager;
import org.eclipse.rdf4j.repository.manager.RepositoryProvider;
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;
```

Date: 9th April 2018.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.Marker;
import org.slf4j.MarkerFactory;

public class CreateRemoteRepository {
  private static Logger logger = LoggerFactory.getLogger(CreateRemoteRepository.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
      String strServerUrl = "http://localhost:7200";

      // Instantiate a local repository manager and initialize it
      RepositoryManager repositoryManager  = RepositoryProvider.getRepositoryManager(strServerUrl);
      repositoryManager.initialize();
      repositoryManager.getAllRepositories();

      // 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);
    }
  }
}
```

2. Conclusion

In this post I detailed how you can create remote repository for Ontotext GraphDB using RDF4J, as well as the benefit of creating a remote repository rather than a local repository. You can find the complete code of this example on github.