JBoss Data Grid lab guides

Lab 3

Thomas Qvarnström

v1.1 2015-12-06

Table of Contents

This guide explains the steps for running lab 3, either follow the steps in the step-by-step section or if you feel adventurous try to accomplish goals without the step-by-step guide.

1. Background

In lab 2 we implemented querying by providing search mapping as to how to index objects via configuration. The search mapping for lab 2 was very simple, but what if your data model was much more complex with links between objects and inheritance. Then the configuration will become more and more complex. Keeping the search mapping configuration and data model in sync will become harder and harder over time as your data models grows.

The best practice recommendation for situations like this is to keep the mapping close to the model. So similar to JPA, Hibernate Search (which is the base for JDG Querying) supports annotation as meta data that describes the search mapping.

This does requires that we can edit the data model.

2. Objectives

Your task in lab 3 is to move the meta-data/configuration of how to index the Task object from the Config object to instead provide this as annotations to the data model object.

3. These are the main tasks of lab 3

1. Remove the search mapping from the `org.jboss.infinispan.demo.Config`.
2. Update the `org.jboss.infinispan.demo.model.Task` model with Search mapping annotation.

4. Setup the lab environment

To assist with setting up the lab environment we have provided a shell script that does this.

NOTE	If you previously setup up lab 1 or lab 2 using this script there is no need to do this for lab 3

1. Run the shell script by standing in the jdg lab root directory (~/jdg-labs) execute a command like this

```
$ sh init-lab.sh --lab=3
```

5. Step-by-Step

1. Open `src/main/java/org/jboss/infinispan/demo/Config.java`
2. Uncomment the search mapping entries like this:

```
GlobalConfiguration glob = new GlobalConfigurationBuilder()
    .globalJmxStatistics().allowDuplicateDomains(true).enable() // This method enables
the jmx statistics of the global configuration and allows for duplicate JMX domains
    .build();

//SearchMapping mapping = new SearchMapping();
//mapping.entity(Task.class).indexed().providedId()
//  .property("title", ElementType.METHOD).field();

Properties properties = new Properties();
//properties.put(org.hibernate.search.Environment.MODEL_MAPPING, mapping);
properties.put("default.directory_provider", "ram");
```

3. Open `src/main/java/org/jboss/infinispan/demo/model/Task.java`
4. Add `@org.hibernate.search.annotations.Indexed` as a class modifier
5. Add `@org.hibernate.search.annotations.Field(store = org.hibernate.search.annotations.Store.YES)` as the modifier to the `private String title` field.
6. Run the JUnit test to verify that everything works.
7. Deploy and test the application

```
$ mvn package jboss-as:deploy
```

8. Congratulations you are done with lab 3.

6. Summary

In lab 2 and lab 3 we can see two different ways to enable searching objects. With properties we do not have to modify the object being indexed and can be a good solution when we for example don't have access to the source code. The other solution with Annotating the object it self is preferable if we have can modify the object. The reason is that it's easier to maintain the indexing meta-data closer to the actual object. When updating an index object developers can also update the indexing.

The search function in JDG is very powerful, in many cases much more powerful then you would find in a typical RDBMS. For example fields can be indexed using different methods enables more advanced queries like sounds like. Note: Complex SQL queries may be hard to migrate to JDG queries and may

require that you redesign data models etc. Another possible solution may be to use MapReduce functions instead, but we will cover that more in lab 7.