JDG + EAP Lab 3 Guide

This explains the steps for lab 3, either follow them step-by-step or if you feel adventurous to accomplish the objectives without the help of the step-by-step section.

Background

In Lab 2 we implemented querying by providing search mapping as to how to index objects via configuration. The search mapping for Lab2 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.

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.

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.

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 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
```

Step-by-Step

1. Open `src/main/java/org/jboss/infinispan/demo/Config.java`

2. Look at the search mapping entries in the `defaultEmbeddedCacheConfiguration()` method

```
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 modifier for the Class called **Task**
5. Add `@org.hibernate.search.annotations.Field(store = org.hibernate.search.annotations.Store.YES)` as the modifier to the String called **title**.
6. The resulting code will contain

```
@org.hibernate.search.annotations.Indexed

public class Task implements Serializable {

.....

  @Column(length = 100)

  @org.hibernate.search.annotations.Field(store =
org.hibernate.search.annotations.Store.YES)

  private String title;
```

7. Ensure that the local JBoss EAP instance is running.
8. Run the JUnit test to verify that everything works.
9. Change directory to projects/lab3.
10. Deploy using this command

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

11. Test the web application at http://localhost:8080/mytodo
12. Congratulations, you are done with lab 3.

Summary

In lab 2 and lab 3 we can see two different ways to enable searching objects. The lab 2 approach, using properties, we do not have to modify the indexed object and this can be a good solution when you don't have access to the source code.

With Annotations in lab 3, we modify the object provided we have the source code. 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 JBoss Data Grid 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, thus enabling more advanced queries like "Sounds Like" searches.

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 Map Reduce functions instead, but we will cover that in another lab.