JBoss Data Grid lab guides

Lab 7

Thomas Qvarnström

v1.1 2015-12-06

Table of Contents

This guide explains the steps for running lab 7, 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

The `todo` application is a success, but we don’t know much about our users. The marketing department has expressed requirements for tracking if users are using computers or tablets, which OS they are using and which browsers are more popular.

2. Use-case

We will implement a solution to store user information. To minimize any impact to performance user information should be stored unstructured. Via Map/Reduce pattern we can structure the data and make use of it using reporting tools. The user information is captured from the User-Agent HTTP header that browser typically provide.

3. These are the main tasks of lab 7

1. Create a local library mode cache together with the RemoteCache
2. Extend the REST layer to store the request data unstructured in a local cache
3. Provide a BIService (Business Intelligence Service) class that can structure the data and return data summarized views of the data.
4. Provide a BIEndpoint (REST Service) to enable UI to access the BIService

4. Step-by-Step

1. Open lab7 project in JBoss Developer Studio
2. Open `Config.java` and fix the getLocalCacheManager which should look like this:

```
private EmbeddedCacheManager getLocalCacheManager() {
    GlobalConfiguration glob = new GlobalConfigurationBuilder()
        .globalJmxStatistics().allowDuplicateDomains(true).enable().build();

    org.infinispan.configuration.cache.Configuration loc = new org.infinispan
.configuration.cache.ConfigurationBuilder()
        .expiration().lifespan(1,TimeUnit.DAYS)
        .build();

    return new DefaultCacheManager(glob, loc, true);
}
```

3. Open `TaskEndpoint` and do the following changes

   Inject the request cache like this:

   ```
   @Inject private Cache<Long, String> requestCache;
   ```

   Add the following line to all REST operations/methods

   ```
   requestCache.putAsync(System.nanoTime(), headers.getRequestHeader("user-agent").get(0
   ));
   ```

4. Open `BIService.java` and do the following changes

   Change the implementation of `getRequestStatiscsPerOs()` method.

   ```
   public Map<String,Integer> getRequestStatiscsPerOs() {
       return new MapReduceTask<Long, String, String, Integer>(requestCache
   .getAdvancedCache())
           .mappedWith(new UserOSCountMapper())
           .reducedWith(new CountReducer())
           .execute();
   }
   ```

   Change the implementation of `getRequestStatiscsPerBrowser()` method.

```java
public Map<String,Integer> getRequestStatiscsPerBrowser() {
    return new MapReduceTask<Long, String, String, Integer>(requestCache
.getAdvancedCache())
        .mappedWith(new UserBrowserVendorCountMapper())
        .reducedWith(new CountReducer())
        .execute();
}
```

5. Investigate and try to understand what is happening in the mapping classes `UserOSCountMapper` and `UserBrowserVendorCountMapper`
6. Open `CountReducer.java` and add the implementation like below:

```java
@Override
public Integer reduce(String reducedKey, Iterator<Integer> iter) {
    int sum = 0;
    while (iter.hasNext()) {
        Integer i = (Integer) iter.next();
        sum += i;
    }
    return sum;
}
```

7. Open `TaskServiceTest.java` and uncomment Test 6
8. Run the JUnit test
9. Deploy the application using the following command from projects/lab7 dir

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

10. **Fill out the lab evaluation**
11. Congratulations you are done with all the labs.