JDG + EAP Lab 5 Guide

This explains the steps for lab 5, either follow them step-by-step or if you feel adventurous try to accomplish goals without the help of the step-by-step guide.

Background

The myTODO application is a huge success and over 100 000 users are today using it to synchronise the task list between different devices. Even though each entry are typically small the data is growing fast. By mid next year numbers of users and expected to grow to 500 000 and 1 million add the end of the year. However the success gives new challenges. The memory used to store tasks is growing fast.

Memory usage today in the future

The average length of a title is 20 characters (about 80 bytes). Together with two Date objects (32 bytes each) and the other fields we can assume that each Task object will take about 200 bytes of memory. The average number of tasks per user is 50. Today the effective data storage (data without meta-data) is about 1 GB of data [100 000 x 50 tasks per user x 200 bytes per task / (1024^3)]. Given a bit of meta-data the amount of java heap used to store objects is about 1.2 GB.

1. What's the expected storage need for data mid next year?
2. What's the expected storage need for data end of next year?

Given that myTODO application is currently using replicated mode these 1.5 GB are stored the same on each node. Switching to distributed mode would mean that we can grow the in-memory storage by adding more nodes.

Using library mode we are using the same java heap for storage as the application itself is using. Using Client/Server mode with HotRod can bring lots of benefits since we can resource optimize. Java EE applications needs CPU resources and may actually behave better with a lower java heap, while JDG Server uses less CPU and can be optimized to use larger java heap.

Use-case

Rewrite the application to only use JDG Client Server mode.

These are the main tasks of lab 5

1. Setup the lab environment
2. Add dependencies to HotRod client
3. Rewrite the `TaskService` to use HotRod instead of client mode
4. Remove dependencies to jdg-core

Setup the lab environment

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

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=5
    ```

    Stop any existing servers from previous labs and Start the servers in separate consoles using the following commands

    EAP Server:

    ```
    $ ./target/jboss-eap-6.3/bin/standalone.sh
    ```

    JDG Server:

```
$ ./target/jboss-datagrid-6.3.0-server/bin/standalone.sh -Djboss.socket.binding.port-offset=100
```

Step-by-Step

1. Discussion pros and cons with different options for solution to the memory usage issuses with you the person to your collegues.
2. Open `~/jdg-labs/project/lab5` in JBoss Developer Studio
3. Add HotRod client development and runtime dependencies by opening `pom.xml` and uncomment the following lines:

   ```
   <dependency>
       <groupId>org.infinispan</groupId>
       <artifactId>infinispan-commons</artifactId>
       <scope>compile</scope>
   </dependency>
   <dependency>
       <groupId>org.infinispan</groupId>
       <artifactId>infinispan-client-hotrod</artifactId>
       <scope>compile</scope>
   </dependency>
   ```

   Note: We are here using the scope `compile` since we want Apache Maven to add these jars and their transient dependencies to the subdirectory `src/main/webapp/WEB-INF/lib`

4. Before we rewrite the contents of `TaskService.java` we need to configure the HotRod client using CDI to produce a `RemoteCache` object. Open `src/main/java/org/jboss/infinispan/demo/Config.java` and add the following to it:

   ```
   @Produces
   public RemoteCache<Long, Task> getRemoteCache() {
       ConfigurationBuilder builder = new ConfigurationBuilder();
       builder.addServer().host("localhost").port(11322);
       return new RemoteCacheManager(builder.build(), true).getCache("default");
   }
   ```

   Note: We are reusing the default cache in this lab, in lab 6 we will configure our own cache instead.

5. You also need to add the following `import` statements if you IDE doesn't fix that

   ```
   import javax.enterprise.inject.Produces;

   import org.infinispan.client.hotrod.RemoteCache;
   import org.infinispan.client.hotrod.RemoteCacheManager;
   import org.infinispan.client.hotrod.configuration.ConfigurationBuilder;
   import org.jboss.infinispan.demo.model.Task;
   ```

6. Open the `src/main/java/org/jboss/infinispan/demo/TaskService.java`

7. Inject a `RemoteCache` object like this:

   ```
   @Inject
   RemoteCache<Long, Task> cache;
   ```

8. You also need to add the following import statements

   ```
   import javax.inject.Inject;
   import javax.ejb.Stateless;
   import java.util.Date;
   import org.infinispan.client.hotrod.RemoteCache;
   ```

9. Implement the `findAll()` method like this:

   ```
   public Collection<Task> findAll() {
       return cache.getBulk().values();
   }
   ```

10. Implement the `insert(Task)` method like this:

    ```
    public void insert(Task task) {
        if(task.getCreatedOn()==null) {
            task.setCreatedOn(new Date());
        }
        task.setId(System.nanoTime());
        cache.putIfAbsent(task.getId(), task);
    }
    ```

11. Implement the `update(Task)` method like this:

    ```
    public void update(Task task) {
        cache.replace(task.getId(), task);
    }
    ```

12. Implement both `delete(Task)` method and `delete(Long)` methods like this:

    ```
    public void delete(Long id) {
        cache.remove(id);
    }


    public void delete(Task task) {
        this.delete(task.getId());
    }
    ```

13. Save the `TaskService.java` file

14. Open `TaskServiceTest.java` and uncomment the the `File[] jars = ....` and `.addAsLibraries(...)`

15. Add these `import` statements

    ```
    import java.io.File;
    import org.jboss.shrinkwrap.resolver.api.maven.Maven;
    ```

16. Run the JUnit test and verify that everything works.

17. Deploy the application using the following command from lab7 dir

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

18. Congratulations you are done with lab 5.