

JDG + EAP Lab 4 Guide

This explains the steps for lab 4, 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 sales account manager for Acme Inc from the RDBMS vendor (Cleora) had a meeting with the CIO of Acme this week. Because Acme Inc used JDG to improve performance instead of purchasing more DB licenses the sales account manager of Cleora decided to try to make up for the lost sales, by raising the price on the licenses that Acme are currently using. The discussion has been harsh and the CIO are really angry at the sales account manager from Cleora. At a similar meeting with the Red Hat Sales team with the CIO the Red Hat Solutions Architect (who the CIO really trusts for advices) suggested that Acme removes the database from the application and instead starts using JDG as the primary data store.

Use-case


Rewrite the application to only use JDG library mode, configure a file store and configure cluster.

These are the main tasks of lab 3

1. Remove JPA code from Task and TaskService
2. Configure a file store (using SingleFileStore)
3. Configure the cache for clustering

Step-by-Step

1. Open `src/main/java/org/jboss/infinispan/demo/TaskService.java` and remove all references to `EntityManager`. `TaskService` should look something like this:

```
 package org.jboss.infinispan.demo;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Date;
import java.util.List;
import java.util.logging.Logger;
```

```

import javax.annotation.PostConstruct;
import javax.ejb.Stateless;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

import org.apache.lucene.search.Query;
import org.hibernate.search.query.dsl.QueryBuilder;
import org.infinispan.Cache;
import org.infinispan.query.CacheQuery;
import org.infinispan.query.Search;
import org.infinispan.query.SearchManager;
import org.jboss.infinispan.demo.model.Task;

@Stateless
public class TaskService {

    @Inject
    Cache<Long,Task> cache;

    Logger log = Logger.getLogger(this.getClass().getName());

    /**
     * This methods should return all cache entries, currently contains mock
up code.
     * @return
     */
    public Collection<Task> findAll() {
        return cache.values();
    }

    /**
     * This method filters task based on the input
     * @param input - string to filter on
     * @return
     */
    public Collection<Task> filter(String input) {
        SearchManager sm = Search.getSearchManager(cache);
        QueryBuilder qb = sm.buildQueryBuilderForClass(Task.class).get();
        Query q = qb.keyword().onField("title").matching(input).createQuery(
);

        CacheQuery cq = sm.getQuery(q, Task.class);
        List<Task> tasks = new ArrayList<Task>();
        for (Object object : cq) {
            tasks.add((Task) object);
        }
        return tasks;
    }
}

```

```

    /**
     * This method persists a new Task instance
     * @param task
     *
     */
    public void insert(Task task) {
        if(task.getCreatedOn()==null)
            task.setCreatedOn(new Date());
        cache.put(task.getId(),task);
    }

```

```

    /**
     * This method persists an existing Task instance
     * @param task
     *
     */
    public void update(Task task) {
        cache.replace(task.getId(),task);
    }

    /**
     * This method deletes an Task from the persistence store
     * @param task
     *
     */
    public void delete(Task task) {
        //Note object may be detached so we need to tell it to remove based on
reference
        cache.remove(task.getId());
    }

    /**
     * This method is called after construction of this SLSB.
     *
     */
    @PostConstruct
    public void startup() {

    }

}

```

1. Implement a new way to generate unique id when inserting new tasks. Replace:

```

public void insert(Task task) {
    if(task.getCreatedOn()==null)
        task.setCreatedOn(new Date());
    cache.put(task.getId(),task);
}

```

with:

```
public void insert(Task task) {
    if (task.getCreatedOn() == null)
        task.setCreatedOn(new Date());
    if(task.getId()==null) {
        task.setId(new Long(cache.size()+1));
    }
    cache.put(task.getId(), task);
}
```

2. Remove JPA references in `src/main/java/org/jboss/infinispan/demo/model/Task.java`. The new Task class should look something like this:

```
package org.jboss.infinispan.demo.model;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Version;

import org.hibernate.search.annotations.Field;
import org.hibernate.search.annotations.Indexed;
import org.hibernate.search.annotations.Store;

/**
 * This class is the JPA entity of a Task
 * @author tqvarnst
 *
 */
@Indexed
public class Task implements Serializable {

    private static final long serialVersionUID = 2315323429163437300L;

    private Long id;

    private int version;

    @Field(store = Store.YES)
    private String title;

    private boolean done;
```

```

private Date createdOn;

private Date completedOn;

public Long getId() {
    return this.id;
}

public void setId(final Long id) {
    this.id = id;
}

public int getVersion() {
    return this.version;
}

public void setVersion(final int version) {
    this.version = version;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (!(obj instanceof Task)) {
        return false;
    }
    Task other = (Task) obj;
    if (id != null) {
        if (!id.equals(other.id)) {
            return false;
        }
    }
    return true;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((id == null) ? 0 : id.hashCode());
    return result;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

```

```

public boolean isDone() {
    return done;
}

public void setDone(boolean done) {
    this.done = done;
}

public Date getCreatedOn() {
    return createdOn;
}

public void setCreatedOn(Date createdOn) {
    this.createdOn = createdOn;
}

public Date getCompletedOn() {
    return completedOn;
}

public void setCompletedOn(Date completedOn) {
    this.completedOn = completedOn;
}

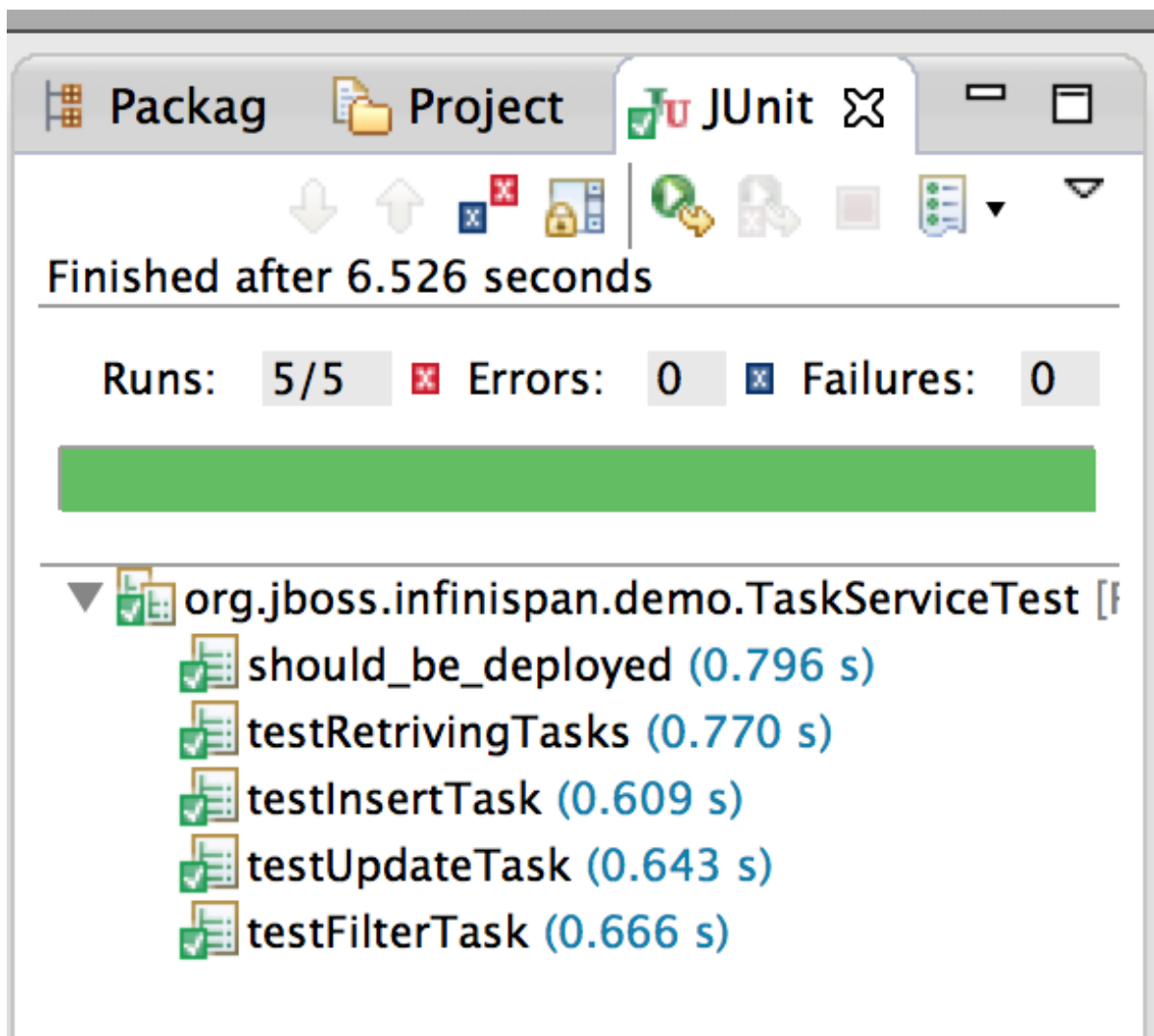
@Override
public String toString() {
    String result = getClass().getSimpleName() + " ";
    if (title != null && !title.trim().isEmpty())
        result += "title: " + title;
    result += ", done: " + done;
    return result;
}

```



```
}
```

1. Run the JUnit test to verify your changes so far.



2. Open `src/main/java/org/jboss/infinispan/demo/Config.java` and add the following to Configuration builder:

```

>- .persistence()
    .addSingleFileStore()
        .location(System.getProperty("jboss.home.dir") + "/cache-store")
        .fetchPersistentState(true)
        .ignoreModifications(true)
        .shared(false)
        .preload(false)
        .async()
            .enable()
            .threadPoolSize(500)
            .flushLockTimeout(1)
            .modificationQueueSize(1024)
            .shutdownTimeout(25000)
    
```

3. Run the JUnit test to verify that your changes works.
4. Add Clustering using CacheMode DIST_ASYNC with 2 owners to Configuration builder.

```

>- ...
    Configuration loc = new ConfigurationBuilder().jmxStatistics()
    
```

```
.enable() // Enable JMX statistics
.clustering().cacheMode(CacheMode.DIST_ASYNC)
.hash().numOwners(2)
...

```

5. Configure the transport for the cluster by adding `jgroups-udp.xml` to the `GlobalConfigurationBuilder`

```
GlobalConfiguration glob = new GlobalConfigurationBuilder()
    .clusteredDefault()
    .transport().addProperty("configurationFile", "jgroups-udp.xml")
    .globalJmxStatistics().allowDuplicateDomains(true).enable()
    .build();

```

1. Run the JUnit test again to verify your changes
2. Deploy the application and test that everything works as before.

```
mvn clean package jboss-as:deploy

```

3. Congratulations you are done with lab 4