Infinispan

Jiří Holuša

jholusa@redhat.com

JBoss Data Grid QE, Red Hat

Task 0

- Clone the repository
  - git clone https://github.com/qa/pv243-a4m36jee-2016-infinispan-seminar-autumn.git
  - git checkout task1
- Run the application
  - mvn clean wildfly:run
  - After deploy the app is available at localhost:8080/infinispan-seminar
- Get yourself familiar with the application
  - How to make the cache accessible from the application?
  - How to store a list of cars so that we can list them on a web page?
  - How to do CRUD operations?

Task 1 – Basic configuration

- Configure the cache according to the following requirements:
  - One-node cluster (local mode)
  - Max. 4 entries in the cache
  - When more than 4 entries stored in the cache, the entry which was not used for longest time will be removed from the cache
  - Maximum lifespan of an (any) entry will be 30 second
- Hints
  - ISPN configuration located in CacheContainerProvider
  - eviction, expiration, clustering cache mode
  - http://infinispan.org/docs/8.2.x/user_guide/user_guide.html#_configuring_cache_programmatically
  - https://docs.jboss.org/infinispan/8.2/apidocs/org/infinispan/configuration/cache/ConfigurationBuilder.html

Task 2 – Cache store configuration

- Commit your changes and checkout task 2 assignment
  - git commit -a -m “Task 1 done” && git checkout task2
- Configure the cache according to the following requirements:
  - When more than 4 entries stored in the cache, the entry which was not used for longest time will be removed from the cache (memory) and **stored in a local file system**
    - check correct behavior by observing size of the file store (under Infinispan-SingleFileStore directory, it should increase when adding more than 4 entries)
- Hints
  - add file cache store
    - http://infinispan.org/docs/8.2.x/user_guide/user_guide.html#single_file_store
  - enable passivation
    - https://docs.jboss.org/infinispan/8.2/apidocs/org/infinispan/configuration/cache/ConfigurationBuilder.html
    - http://infinispan.org/docs/8.2.x/user_guide/user_guide.html#cache_loader_behavior_with_passivation_disabled_vs_enabled

Task 3 – Querying

- Commit your changes and checkout task 3 assignment
  - git commit -a -m “Task 2 done” && git checkout task3
- Implement searching by number plate in car list by following steps:
  - Configure cache for **local** indexing
  - Mark Car entity to be indexed and field “numberPlate” to be indexed without using analyzer (hint: analyze=Analyze.NO)
  - Implement search query using **Infinispan Query DSL** (not Lucene queries) to search cars by number plates (or substrings of them)
- Hints
  - indexing(), having(), like(), '%' operator
  - https://docs.jboss.org/infinispan/8.2/apidocs/org/infinispan/configuration/cache/ConfigurationBuilder.html
  - http://infinispan.org/docs/8.2.x/user_guide/user_guide.html#simple_example
  - http://infinispan.org/docs/8.2.x/user_guide/user_guide.html#infinispan_s_query_dsl

Task 4 – Clustering (slide 1)

- Commit your changes and checkout task 4 assignment
  - git commit -a -m “Task 3 done” && git checkout task4
- Configure cache for clustering
  - Configure global configuration to use default clustering configuration
  - Configure cache to replicated synchronous cache mode
- Instructions on how to run in on next slide
- Hints
  - clusteredDefault(), clustering(), CacheMode.REPL_SYNC
  - http://infinispan.org/docs/stable/user_guide/user_guide.html#configuring_cache_programmatically
  - https://docs.jboss.org/infinispan/8.2/apidocs/org/infinispan/configuration/cache/ConfigurationBuilder.html

Task 4 – Clustering (slide 2)

- How to run it (after configuring it as said on previous slide)
  - Duplicate the project folder, let's say you will have now project1 and project2 (two identical copies)
  - In one terminal, in project1 directory run (starts first server on 8080):
    - mvn clean wildfly:run -Dwildfly.serverArgs="-Djava.net.preferIPv4Stack=true"
  - In second terminal, in project2 directory run (starts second server on 8180):
    - mvn clean wildfly:run -Dwildfly.serverArgs="-Djboss.socket.binding.port-offset=100,-Djava.net.preferIPv4Stack=true" -Dwildfly.port=10090
  - After you start both of them, you should see in the server log something like "...Received new cluster view for channel ISPN..."
- Testing it
  - open application on localhost:8080/infinispan-seminar and insert a new car there
  - open application on localhost:8180/infinispan-seminar and refresh the home page, you should see the new car there

Task 5 – Listeners

- Commit your changes and checkout task 5 assignment (you continue only on the origin project checkout)
  - git commit -a -m “Task 4 done” && git checkout task5
- Print an info message every time an entry is created/modified/deleted using listeners
  - Use WatchListener class for it
  - Mark the class as listener
  - Create an instance of the class and bind it to the cache configuration as listener
  - Mark methods of WatchListener to listen to appropriate events
- Hints
  - http://infinispan.org/docs/stable/user_guide/user_guide.html#_Listeners_and_notifications_section

Task 6 – Client/Server mode (1)

- Commit your changes and checkout task 6 assignment
  - git commit -a -m “Task 5 done” && git checkout task6
- Setup server
  - Download and unzip Infinispan server
  - Declare a cache named “carcache” in the standalone/configuration/standalone.xml under Infinispan subsystem
- Setup client
  - Connect to the ISPN server remotely via Hot Rod using RemoteCache
- Run the Infinispan server
  - ISPN_SERVER/bin/standalone.xml -Djboss.socket.binding.port-offset=100
- Run the application as usual
  - mvn wildfly:run
- No data pre-inserted, hence test it by adding some cars
- More hints on next page

Task 6 – Client/Server mode (2)

- Hints for the client side
  - http://infinispan.org/docs/stable/user_guide/user_guide.html#java_hot_rod_client
  - You only need to specify server address and port in the configuration
    - host is 127.0.0.1 (localhost)
    - port will be 11322 (11222 is default + we have 100 port offset to avoid port collision between those two servers)
- Hints for the server side
  - in standalone/configuration/standalone.xml under Infinispan subsystem, do following steps:
    - under the already present <cache-container>, add <local-cache> named 'carcache' with no specific configuration, hint: http://infinispan.org/docs/stable/server_guide/server_guide.html#caches

Task 3 – Distributed streams

- Switch to branch infinispan-03-prepare-dist-streams
- Add search logic to **CarManager**
  - Create a query using Distributed streams API which will search for all cars matching one of these parameters: color, brand or country (might match just one of those parameters)

- Hints
  - cache.values()
  - map(), filter(), collect()
  - http://infinispan.org/docs/8.2.x/user_guide/user_guide.html#_streams
  - https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html