PojoCache Tutorial

Ben Wang <ben.wang@jboss.com>

Galder Zamarreño <galder.zamarreno@jboss.com>

Release 2.0.0

June 2007

1. Introduction

PojoCache is an in-memory, transactional, and replicated POJO (plain old Java object) cache system that allows users to operate on a POJO transparently without active user management of either replication or persistency aspects. This tutorial focuses on the usage of the PojoCache API.

For details of configuration, usage and APIs, please refer to the users manual [1].

2. What You Will Learn

- PojoCache creation and modification
- Replication of POJO fields
- Using Collections in PojoCache
- Transactions

3. Configuration

First download the JBoss Cache 2.x distribution from the download page [2] . You probably want the `JBossCache-pojo-2.X.Y.zip` distribution. Unzip it, and you will get a directory containing the distribution, such as `JBoss-Cache-pojo-2.X.Y` . For the sake of this tutorial, I will refer to this as `PojoCache` .

The configuration files are located under the `PojoCache/etc` directory. You can modify the behavior of the underlying cache through editing the various configuration files.

- `log4j.xml` . Logging output. You can enable logging, specify log levels or change the name and path to the log file.
- `META-INF/replSync-service.xml` . Cache configuration file used for this tutorial.
- `pojocache-aop.xml` . PojoCache configuration file that contains, amongst other things, the annotation to use on

[1] http://labs.jboss.org/portal/jbosscache/docs/index.html
[2] http://labs.jboss.org/portal/jbosscache/download/index.html

POJOs so that they're aspectised. For more information, please the PojoCache users manual [3] .

4. Script

The only script needed for this tutorial is the `PojoCache/build.xml` ant script and the accompanying driver scripts (`build.sh` for Unix and `build.bat` for Windows).

5. Example POJOs

The example POJO classes used for PojoCache demo are: `org.jboss.cache.pojo.test.Person` and `org.jboss.cache.pojo.test.Address` . They are located under `tests/functional` directory.The demo will demonstrate that once a POJO has been attached to the cache, plain get/set POJO methods will be intercepted by the cache.

Here is the snippet of the class definition for `Person` and `Address` with the `Replicable` annotation.

```
@org.jboss.cache.pojo.annotation.Replicable
public class Person {
   ...
   public String getName() { return name; }
   public void setName(String name) { this.name=name; }
   ...
   public List<String> getLanguages() { return languages; }
   public void setLanguages(List<String> languages) { this.languages = languages; }
   ...
   public Address getAddress() { return address; }
   public void setAddress(Address address) { this.address = address; }
   ...
}
```

```
@org.jboss.cache.pojo.annotation.Replicable
public class Address {
   ...
   public String getStreet() { return street; }
   public void setStreet(String street) { this.street=street; }
   ...
}
```

6. Running The Demo GUI

The demo is run by calling the ant script (via the driver) with the `run.demo.pojocache` target. E.g.,

```
./build.sh run.demo.pojocache
```

This will cause a GUI window to appear, giving you a tree view of the cache in the top pane and a BeanShell view of the JVM in the lower pane.

[3] http://labs.jboss.org/portal/jbosscache/docs/index.html

The BeanShell view is preset with the following variables:

- `cache` - a reference to the PojoCache interface, used by the GUI instance.
- `transactionManager` - a reference to the registered transaction manager.

The references made available to the BeanShell window point to the same cache instance used by the tree view in the GUI above.

To run the demo as a replicated demo, it is useful to start another command line window and run the ant script again as you did above. Now you will have two cache instances running in two separate GUIs, replicating state to each other.

7. Tutorials

It is recommended that you shut down and restart the demo GUI for each of the following tutorials, to ensure clean caches every time. To inspect POJO attribute changes via GUI, please refer to the PojoCache user manual [4] to understand how the POJOs are mapped internally in the cache.

7.1. PojoCache API, POJO manipulation, and Replication

For this tutorial, start two instance of the demo GUI. In this tutorial, we will:

- Attach POJOs to the cache and see them being replicated.
- After attaching, manipulate the POJOs and see the individual changes replicated.
- Retrieve POJOs from the cache, manipulate them and see the changes replicated.
- Create POJOs that share a common POJO and the consequences of changes to this.
- Detach POJOs from the cache.
- After detaching, manipulates the POJOs and see how the values in the cache are unchanged.

1. In the 1st GUI instance, create a POJO, i.e. a Person with an Address:

```
joe = new Person();
joe.setName("Joe Black");
joe.setAge(31);

addr = new Address();
addr.setCity("Sunnyvale");
addr.setStreet("123 Albert Ave");
addr.setZip(94086);

joe.setAddress(addr);
```

[4] http://labs.jboss.org/portal/jbosscache/docs/index.html

2. Attach the POJO to the cache:

```
cache.attach("pojo/joe", joe);
```

3. Change attributes of the POJO and see the individual changes being propagated to the 2nd cache GUI:

```
joe.setAge(41);
```

4. In the 2nd GUI instance, get a reference to the Person in the cache and create a second Person with the existing Person's Address:

```
joe = cache.find("pojo/joe");

mary = new Person();
mary.setName("Mary White");
mary.setAge(30);

mary.setAddress(joe.getAddress());
```

5. Attach the new POJO to the cache:

```
cache.attach("pojo/mary", mary);
```

6. Now, change either Person's Address and see how the change applies to both POJOs and has been propagated to the other cache, visible in the 1st GUI instance:

```
mary.getAddress().setZip(95000);
```

7. Still in the 2nd GUI instance, detach the POJOs from the cache and see how the POJOs are no longer visible:

```
cache.detach("pojo/joe");
cache.detach("pojo/mary");
```

8. Finally, in any of GUI instances, change some attributes of the POJO and see these changes have no effect in

the cache:

```
joe.setName("Joe White");
```

7.2. Collections

For this tutorial, start two instances of the demo GUI. In this tutorial, we will:

- Attach a POJO to the cache and see it being replicated.
- Set a Collection attribute in this POJO
- Manipulate this Collection attribute and see the changes visible in the GUI and being replicated
- Detach a POJO from the cache.

1. In the 1st GUI instance, create a POJO with a Collection attribute:

```
joe = new Person();
joe.setName("Joe Black");

lang = new ArrayList();
lang.add("Spanish");

joe.setLanguages(lang);
```

2. Attach the POJO to the cache:

```
cache.attach("pojo/joe", joe);
```

3. Get a proxy reference to the Collection and add a new element to it:

```
proxyLang = joe.getLanguages();
proxyLang.add("English");
```

4. Detach the pojo from the cache:

```
```

```
cache.detach("pojo/joe");
```

5. Use the proxy reference to the Collection to add another element and see how this does not get added to the cache:

```
proxyLang.add("French");
```

8. Transactions

For this tutorial, start two instances instance of the demo GUI. Repeat the exercises in the previous tutorial, only starting transactions before attaching/detaching nodes or modiying the POJOs. This will depict how replication only occurs on transaction boundaries. Try rolling back a few transactions as well, to see how nothing gets replicated in these cases.