Getting Started with {brandname} 10.0

Table of Contents

Chapter 1. Downloading and installing {brandname}

To run {brandname}, you'll need

- A Java 1.8 JDK
- Maven 3.2+, if you wish to use the quickstart examples or create a new project using {brandname} archetype
- the {brandname} distribution zip, if you wish to use {brandname} in server mode, or want to use the jars in an ant project

If you already have any of these pieces of software, there is no need to install them again!

1.1. JDK

Choose your Java runtime, and follow their installation instructions. For example, you could choose one of:

- OpenJDK
- Oracle Java SE

1.2. Maven

Follow the official Maven installation guide if you don't already have Maven 3.2 installed. You can check which version of Maven you have installed (if any) by running mvn --version . If you see a version newer than 3.2, you are ready to go.

You can also deploy the examples using your favorite IDE. We provide instructions for using Eclipse only.

1.3. {brandname}

Finally, download {brandname} from the {brandname} downloads page.

1.3.1. Getting {brandname} from Maven

Add to your pom:

```
<dependency>
  <groupId>org.infinispan</groupId>
  <artifactId>infinispan-embedded</artifactId>
  <!-- Replace ${version.infinispan} with the
  version of {brandname} that you're using. -->
  <version>${version.infinispan}</version>
</dependency>
```

1.3.2. Installing {brandname} inside Apache Karaf

The {brandname} jars contain the required OSGi manifest headers and can be used inside OSGi runtime environments as OSGi bundles. In addition to them you will need to install the required 3rd party dependencies. You can install them one by one if you wish but to make things easier for you we are providing Apache Karaf "features" files (also called "feature repositories") which define all required dependencies and can be used to install everything in just a few steps.

Installing bundles using "features" requires:

- registering the feature repositories inside Karaf
- installing the features contained in the repositories

You will first need to start the Apache Karaf console:

```
$ cd <APACHE_KARAF_HOME>/bin
$ ./karaf
```

To register a feature repository you need to use the `feature:repo-add` command (or `features:addUrl` if you are using Apache Karaf 2.3.x) and provide its URL (Apache Maven URLs are preferred):

```
karaf@root()> feature:repo-add mvn:org.infinispan/infinispan-
core/${version}/xml/features
```

Replace `${version}` with the actual version you plan to use. You can now get the list of available features using:

```
karaf@root()> feature:list | grep infinispan
infinispan-core               | ${version}   |           | infinispan-core-${version}
|
```

and install them using:

```
karaf@root()> feature:install infinispan-core/${version}
```

In Apache Karaf the commands are `features:list` and `features:install`.

Alternatively you can just pass the `-i` flag to the `feature:repo-add` command which will install all the features defined in that repository:

```
karaf@root()> feature:repo-add -i mvn:org.infinispan/infinispan-
core/${version}/xml/features
```

This should get you started using {brandname} in library mode. To get additional functionality just install the corresponding features. For example to use the RocksDB cachestore install:

```
karaf@root()> feature:repo-add -i mvn:org.infinispan/infinispan-cachestore-
rocksdb/${version}/xml/features
```

The URL for the feature repositories is constructed from the Maven artifact coordinates using the format:

```
mvn:<groupId>/<artifactId>/<version>/xml/features
```

To use {brandname} in client/server mode install the Hot Rod Client feature:

```
karaf@root()> feature:repo-add -i mvn:org.infinispan/infinispan-client-
hotrod/${version}/xml/features
```

Currently feature repositories are available for the following artifacts:

- infinispan-commons
- infinispan-core
- infinispan-cachestore-jdbc
- infinispan-cachestore-jpa
- infinispan-cachestore-rocksdb
- infinispan-cachestore-remote
- infinispan-client-hotrod

For more details regarding the commands available inside Apache Karaf please consult its user manual.

1.4. Download the quickstarts

The quickstarts are in GitHub, in https://github.com/infinispan/infinispan-quickstart.

Clone this repository using:

```
$ git clone https://github.com/infinispan/infinispan-quickstart
```

Chapter 2. Creating your own {brandname} project

2.1. Maven Archetypes

{brandname} currently has 2 separate Maven archetypes you can use to create a skeleton project and get started using {brandname}. This is an easy way to get started using {brandname} as the archetype generates sample code, a sample Maven pom.xml with necessary dependencies, etc.

You don't need to have any experience with or knowledge of Maven's Archetypes to use this! Just follow the simple steps below.

2.1.1. Starting a new project

Use the newproject-archetype project. The simple command below will get you started:

```
$ mvn archetype:generate \
    -DarchetypeGroupId=org.infinispan.archetypes \
    -DarchetypeArtifactId=newproject-archetype \
    -DarchetypeVersion=1.0.23 \
    -DarchetypeRepository=http://repository.jboss.org/nexus/content/groups/public
```

You will be prompted for a few things, including the *artifactId* , *groupId* and *version* of your new project. And that's it - you're ready to go!

2.1.2. Playing with your new project

The skeleton project ships with a sample application class, interacting with {brandname}. You should open this new project in your IDE - most good IDEs such as IntelliJ and Eclipse allow you to import Maven projects, see this guide and this guide . Once you open your project in your IDE, you should examine the generated classes and read through the comments.

2.1.3. On the command line...

Try running

```
$ mvn install -Prun verify
```

in your newly generated project! This runs the main() method in the generated application class.

2.1.4. Writing a test case for {brandname}

This archetype is useful if you wish to contribute a test to the {brandname} project and helps you get set up to use {brandname}'s testing harness and related tools. Use

```
$ mvn archetype:generate \
    -DarchetypeGroupId=org.infinispan.archetypes \
    -DarchetypeArtifactId=testcase-archetype \
    -DarchetypeVersion=1.0.23 \
    -DarchetypeRepository=http://repository.jboss.org/nexus/content/groups/public
```

As above, this will prompt you for project details and again as above, you should open this project in your IDE. Once you have done so, you will see some sample tests written for {brandname} making use of {brandname}'s test harness and testing tools along with extensive comments and links for further reading.

2.1.5. On the command line...

Try running

```
$ mvn test
```

in your newly generated project to run your tests.

The generated project has a few different profiles you can use as well, using Maven's -P flag. E.g.,

```
$ mvn test -Pudp
```

Available profiles

The profiles available in the generated sample project are:

- udp: use UDP for network communications rather than TCP
- tcp: use TCP for network communications rather than UDP
- jbosstm: Use the embedded JBoss Transaction Manager rather than {brandname}'s embedded transaction manager

Contributing tests back to {brandname}

If you have written a functional, unit or stress test for {brandname} and want to contribute this back to {brandname}, your best bet is to fork the {brandname} sources on GitHub . The test you would have prototyped and tested in an isolated project created using this archetype can be simply dropped in to {brandname}'s test suite. Make your changes, add your test, prove that it fails even on {brandname}'s upstream source tree and issue a pull request .

2.1.6. Versions

The archetypes generate poms with dependencies to specific versions of {brandname}. You should edit these generated poms by hand to point to other versions of {brandname} that you are interested in.

2.1.7. Source Code

The source code used to generate these archetypes are on GitHub . If you wish to enhance and contribute back to the project, fork away!

Chapter 3. {brandname} GUI demo

This document walks you through using the {brandname} GUI demo that ships with {brandname}, and assumes that you have downloaded the latest version of {brandname} and unzipped the archive.

I will refer to the {brandname} directory created by unzipping the archive as `${INFINISPAN_HOME}`.

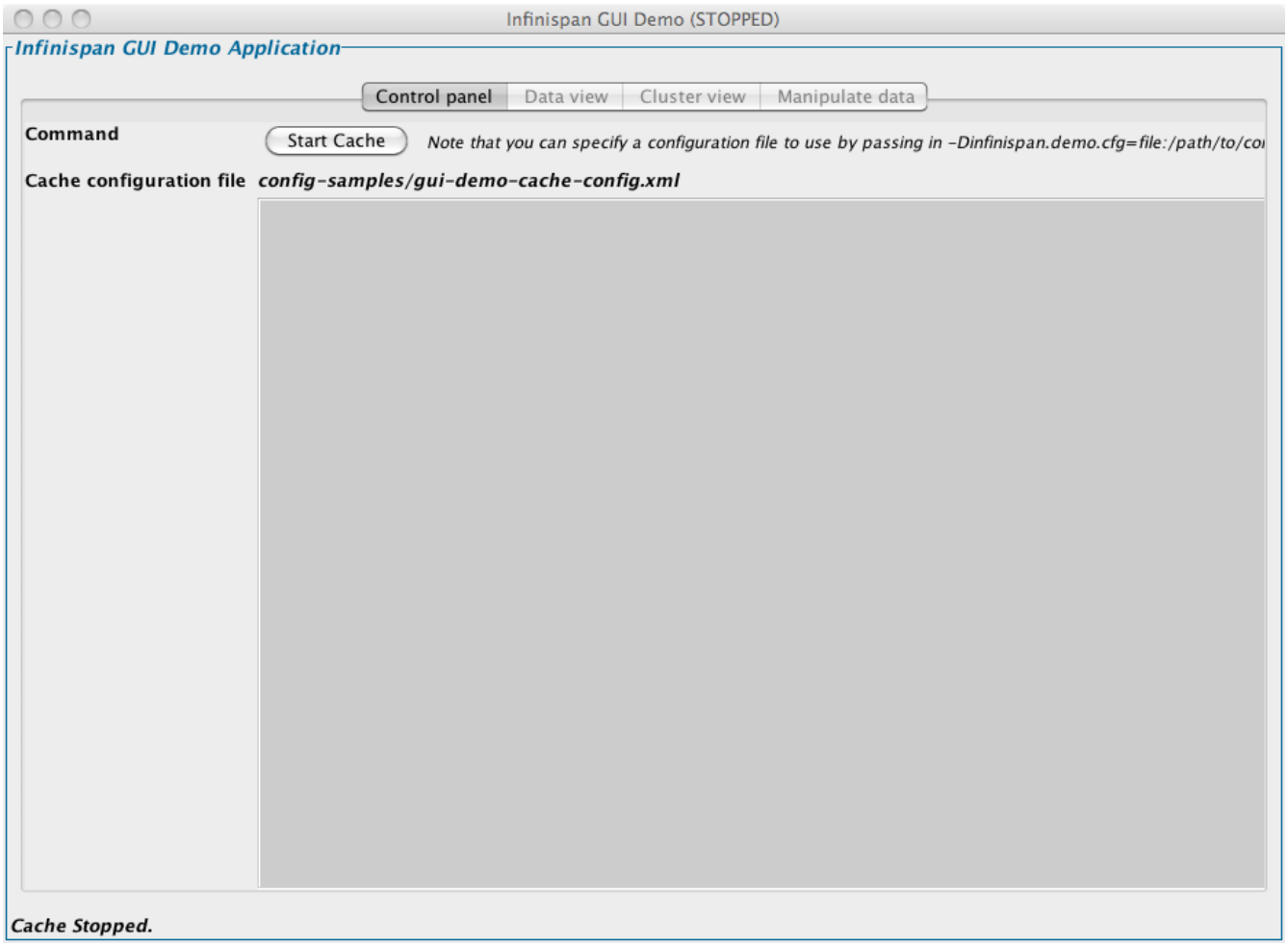
You will need either the -bin.zip or -all.zip version for this demo.

3.1. Step 1: Start the demo GUI

Open a console and enter:

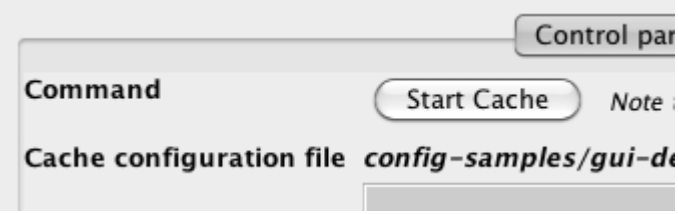
$ cd ${INFINISPAN_HOME} $ bin/runGuiDemo.sh

An equivalent `runGuiDemo.bat` file is also provided for Windows users.



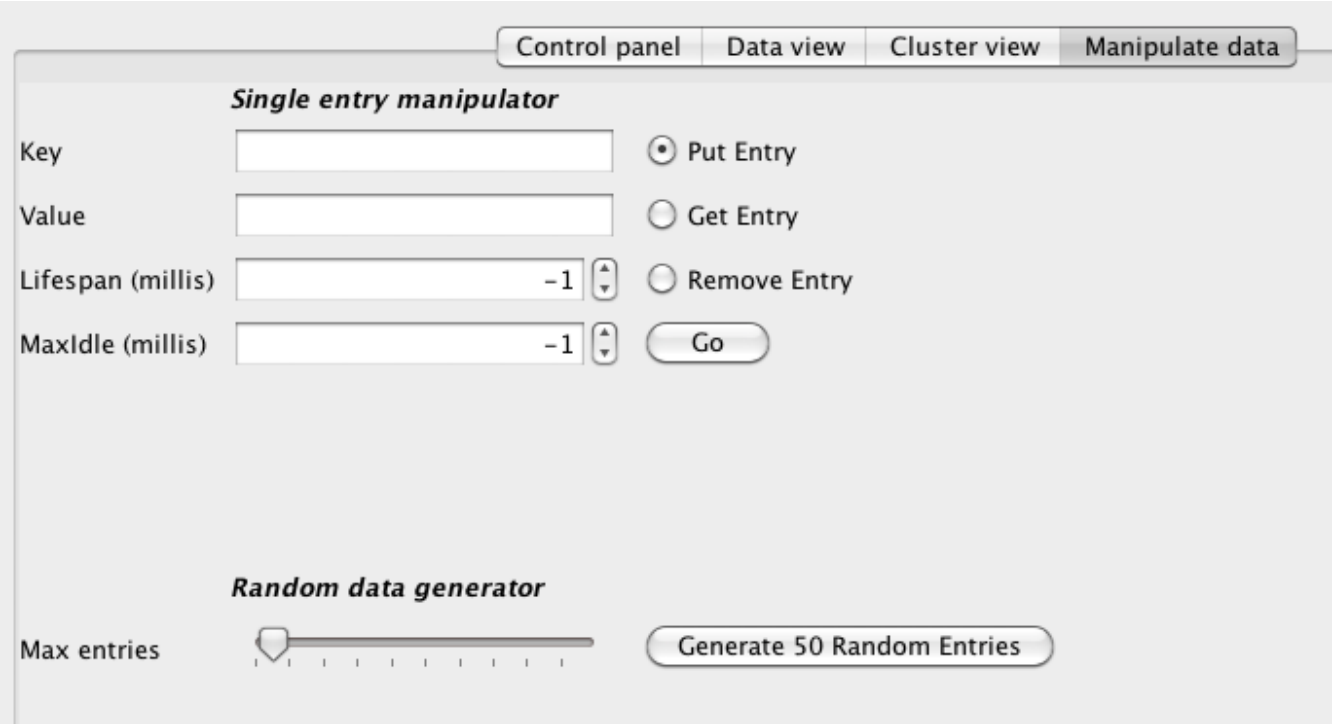
3.2. Step 2: Start the cache

Start the cache in the GUI that starts up, using the *Start Cache* button.



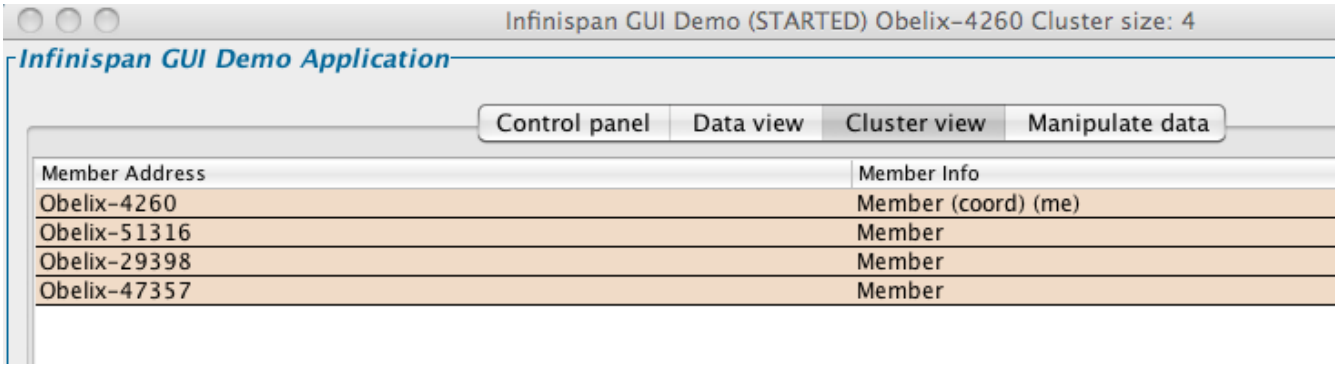
3.3. Step 3: Manipulate data

In the *Manipulate Data* tab, add entries, generate random data, etc.



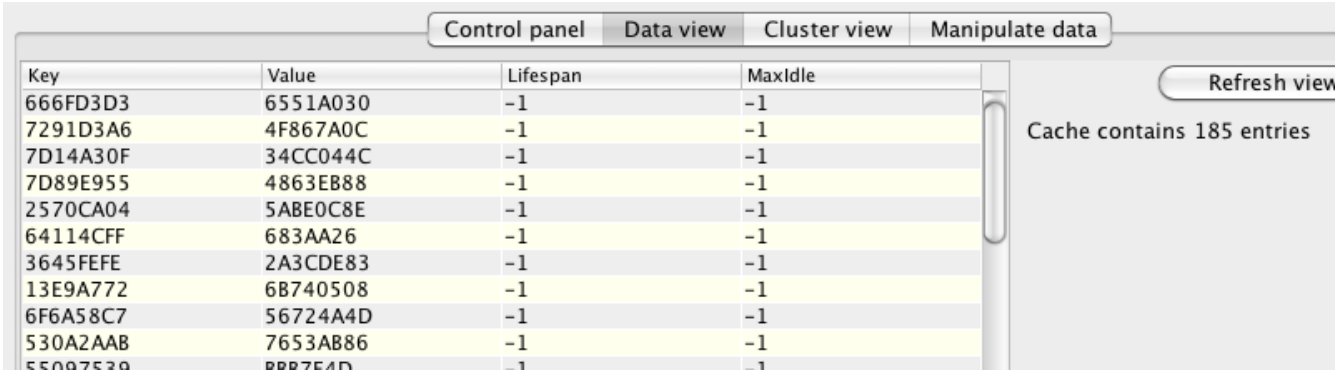
3.4. Step 4: Start more cache instances

Repeat steps 1 and 2 to launch and start up more caches. Watch cluster formation in the *Cluster View* tab.



3.5. Step 5: Manipulate more data

Add and remove data on any of the nodes, and watch state being distributed. Shut nodes down as well to witness data durability.


Control panel
Data view
Cluster view
Manipulate data
Key
Value
Lifespan
MaxIdle
666FD3D3 6551A030 -1 -1
7291D3A6 4F867A0C -1 -1
7D14A30F 34CC044C -1 -1
7D89E955 4863EB88 -1 -1
2570CA04 5ABE0C8E -1 -1
64114CFF 683AA26 -1 -1
3645FEFE 2A3CDE83 -1 -1
13E9A772 6B740508 -1 -1
6F6A58C7 56724A4D -1 -1
530A2AAB 7653AB86 -1 -1
Refresh view
Cache contains 185 entries