Chapter 5. Client Cache

5.1. Introduction

In this lab, you will gain hands-on experience working with a GemFire client cache configuration.

Concepts you will gain experience with

- Configuring a client cache XML file
- Defining regions both for local private use as well as connecting to cache server
- Configuring cache properties using gemfire.properties
- Initializing the client cache in an application

Estimated completion time: 45 minutes

5.2. Quick Instructions

Quick instructions for this exercise have been embedded within the lab materials in the form of TODO comments. To display them, open the `Tasks` view (Window -> Show view -> Tasks (*not Task List*)).

5.3. Detailed Instructions

Instructions for this lab are divided into specific section. Each section describes the steps to perform specific tasks. Before beginning this lab, make sure you have started the server side processes using the `startServer.sh` script (`startServer.bat` for Windows) in the `server-bootstrap` lab folder.

5.3.1. Configuring the Cache Client XML file

The client's cache.xml <client-cache> declaration automatically configures it as a standalone Gemfire application. At this point, the locator and server will have already been started and the listener is listening on port `41111`.

In this section, you will edit the `xml/clientCache.xml` file to perform the following tasks.

1. (`TODO-01`) Define a pool configuration with one entry configured to contact the locator at `localhost` and listening on port `41111`
2. (`TODO-02`) Define a region called `Customer` as a proxy type region using either the refid (region shortcut) or explicitly using region attributes. Similarly, create a region called `BookMaster` but this time create it as a caching proxy type region.

5.3.2. Configuring additional cache properties

In addition to providing configuration information in the cache xml file, you can also define properties using a gemfire.properties file. In this section, you will use this file to define the logging level that will be used by the client application. Some of the possible configurations are found in the table following. Consult the GemFire User Guide appendix for a full list of properties.

Table 5.1. Client Cache properties

Property	Meaning
log-level	Level of detail of the messages written to the system member's log. Setting log-level to one of the ordered levels causes all messages of that level and greater severity to be printed. Valid values from lowest to highest are fine, config, info, warning, error, severe, and none. Default = config
durable-client-id	Used only for clients in a client/server installation. If set, this indicates that the client is durable and identifies the client. The ID is used by servers to reestablish any messaging that was interrupted by client downtime. Default = none
durable-client-timeout	Used only for clients in a client/server installation. Number of seconds this client can remain disconnected from its server and have the server continue to accumulate durable events for it. Default = 300

(`TODO-03`) Open the `gemfire.properties` file and set the log level initially to `warning`.

5.3.3. Configure client application to initialize client cache

Open the `ClientCacheTests.java` file in the `com.gopivotal.bookshop.tests` package (under `src/test/java`) and add code for the following methods.

- (`TODO-04`) In the `setup()` method to, add code to create a `ClientCache`. Use the appropriate property on the `ClientCacheFactory` class to define the `cache-xml-file` property to point to `xml/clientCache.xml`.
- (`TODO-05`) Also, add code to the same `setup()` method to get the `Customer` region and assign to the class private field `customers`. Add a similar statement to get the `BookMaster` region and assign to the private field `books`.

> **Note**
>
> You will not edit the test code itself, which performs the get operations. Details on how to make such calls to the region will be covered in the next section.

- (`TODO-06`) Finally, run the program to verify you were able to connect. If you implemented the client cache configuration and the client setup code correctly, one test should pass and one test is currently being ignored (and will be addressed in the next section of this lab).
- As an optional step, go back and edit the `gemfire.properties` file and change the log level to `fine`. Re-run the application and notice the additional detal that is output when running your client application.

5.3.4. Define a local region

In this final section, you will define a local region and run a test to load and fetch data from it.

- (`TODO-07`) Open the `xml/clientCache.xml` file and add an additional region called `LocalRegion`. Use the region shortcuts to define this as a local region.
- (`TODO-08`) Return to the `ClientCacheTests` class and comment the `@Ignore` annotation on `testLocalRegion()`.
- (`TODO-09`) Finally re-run the tests and verify that all tests pass.

5.3.5. Creating a DAO object for BookMaster

To begin, open the `BookMasterDao.java` class in the com.gopivotal.bookshop.buslogic package in the IDE. The purpose of this class is to hide the basic operations involved in creating, reading, updating and deleting entries (the CRUD operations).

Note that the region is defined to store `BookMaster` objects with an `Integer` key as illustrated by the private field defined at the top of the DAO. The constructor is responsible for initializing the region using the `ClientCache` that is passed in.

You will implement the basic functionality of this class in the following steps.

- (`TODO-10`) Before implementing the functionality in `BookMasterDao`, open the corresponding JUnit test class (`DataOperationsTests.java` in the com.gopivotal.bookshop.tests package in `src/test/java`). Take a moment to inspect the various test methods to see how the DAO is used.
- (`TODO-11`) Implement the `insertBook()` method. Use the key and book object to insert into the region.

> **Tip**
>
> Recall that there are different methods that can be used to insert an entry. Use the method that enforces that the entry can't already exist.

- (`TODO-12`) Implement the `getBook()` method. Use the supplied key to retrieve the associated entry from the region and return it.
- (`TODO-13`) Implement the `updateBook()` method. Use the supplied key and `BookMaster` object to update the existing entry in the region.
- (`TODO-14`) Implement the `removeBook()` method. Use the supplied key to delete the entry from the region. Recall that there are two methods to do this. Either one is acceptable.
- (`TODO-15`) When you have completed all the steps above, run the test suite (`DataOperationsTests`). Make sure the GemFire server process (locator and server) are already running if you haven't done so already.

Congratulations!! You have completed this lab.
