DRAFT: Coherence Developer Exercise Notebook (Edition 1.0)

For Coherence Version 3.3

Brian Oliver
brian.oliver@oracle.com

Introduction

Welcome to the Oracle Coherence Developer Exercise Notebook - a compendium of step-by-step hands-on development exercises designed to lead you through the process of understanding and integrating Oracle Coherence into Java and .NET based applications.

While much of the information contained with in this notebook is available online, in the Oracle Coherence Wiki, Forum and download package, this book organizes such material into a step-by-step learning process, enabling you to become proficient with Oracle Coherence, covering everything from downloading and installing Coherence through to advance Grid processing.

Exercise: Downloading Coherence (Java Edition)

Objective:

Download the latest version of Coherence (Java Edition) from www.oracle.com.

Duration:

10 minutes.

Prerequisite Exercises:

(none)

Other Prerequisites:

(none)

Knowledge:

Coherence (Java Edition) ships as a single zip file, typically called coherence.zip.

System Variables:

$DOWNLOADS_FOLDER$ = The path to the download folder.

$COHERENCE_ZIP$ = The downloaded Coherence filename.

Steps:

1. Open the following URL:
   http://www.oracle.com/technology/software/index.html

2. In the "Middleware" section, select the "Coherence" link.
   (You may need to log into your oracle.com account)

3. Click "Accept the License Agreement".

4. Select the "Oracle Coherence Version x.y.z – Pure Java …" link to download Coherence.

   Once completed you should have a file called coherence-xyz.zip, from here on referred to as $COHERENCE_ZIP$, that is around 10mb in size located in the download folder of your choosing, from here on referred to as $DOWNLOADS_FOLDER$.

   Eg: If you have downloaded Coherence 3.3.1 into c:\downloads, $COHERENCE_ZIP$ = coherence-331.zip and $DOWNLOADS_FOLDER$ = c:\downloads

Exercise: Installing Coherence (Java Edition)

Objective:

Install Coherence (Java Edition).

Duration:

15 minutes.

Prerequisite Exercises:DRAFT: Coherence Developer Exercise Notebook (Edition 1.0)

Other Prerequisites:

1. Administration privileges to install software and set system environment variables.

2. An understanding of how to use a command prompt (windows) or shell (unix), including setting environment variables and creating and moving between folders.

3. A utility application to unzip the downloaded $COHERENCE_ZIP$. For example; winzip or unzip

4. A working installation of a Java Software Development Kit (SDK) version 1.4.2 or above (for these exercises Java 5 is required)

5. The JAVA_HOME environment variable set to the base of the JDK installation.

6. JAVA_HOME/bin set in the system path variable.

7. Being able to successfully execute "java –version" from a system command shell to determine the current version of Java that is installed. Should this step fail, you must correctly install a Java Development Kit (JDK).

Knowledge:

To install coherence you simply need to unzip the distribution file, typically called coherence.zip, into a folder of your choosing. Additionally it's often desirable to set a COHERENCE_HOME environment variable.

System Variables:

$DOWNLOADS_FOLDER$ = The path to the downloads folder.

$COHERENCE_ZIP$ = The downloaded Coherence filename.

$LIBRARIES_HOME$ = Path where any other Java Libraries your using have been previously installed.

$COHERENCE_HOME$ = Path where Coherence was installed.

Steps:

1. Using your unzip utility application, unzip the Coherence zip file

$COHERENCE_ZIP$ where it exists in the $DOWNLOADS_FOLDER$. This will produce a folder called "coherence" in the $DOWNLOADS_FOLDER$.

2. [optional] Rename the "coherence" folder in the $DOWNLOADS_FOLDER$ to reflect the version downloaded. Eg: If you downloaded Coherence version 3.3.1, Rename the unzipped "coherence" to "coherence-3.3.1".

3. [optional] Move the unzipped and possibly renamed "coherence" folder from $DOWNLOADS_FOLDER$ into a suitable location for use in applications, typically called the $LIBRARIES_HOME$. Eg: Move c:\downloads\coherence-3.3.1 into c:\libraries

4. Define a system-level environment variable called COHERENCE_HOME that points to the location of the unzipped Coherence folder. Eg: define COHERENCE_HOME = c:\libraries\coherence-3.3.1

5. [unix only] Ensure all of the unix scripts in the $COHERENCE_HOME$/bin folder are executable by executing the following;

On Unix (bash shell)

```
chmod a+x $COHERENCE_HOME/bin/*.sh
```

6. A correctly installed Coherence $COHERENCE_HOME$ folder will have the following sub-folders;

   bin
   doc
   examples
   lib

Exercise: Testing a Coherence Installation

Objective:

Ensure that an installation of Coherence will cluster Java processes together. Should an installation not be capable of Clustering on a single machine, you may need to reconfigure your network and or firewall settings.

Duration:

15 minutes (assuming a correctly configured network is available)

Prerequisite Exercises:

Exercise: Installing Coherence (Java Edition)

Other Prerequisites:

(None)

Knowledge:

Coherence uses a variety of network addresses and ports to enable communication between clustered processes. Should these addresses and/or ports be unavailable, for example, due to other applications using said addresses and ports or use of a firewall, Coherence may be unreliable, fail to cluster or work at all.

By default Coherence assumes the following network addresses and ports to be available.

Address / Port / Type	**Purpose**
224.3.3.1 / 33389 / Multicast	Cluster member discovery and broadcast.
localhost / 8088+ / Unicast	Inter-process communication between cluster members. (localhost is the local ip address, not the loopback address)

Coherence ships with two simple command-line (shell-based) applications that can be used to determine if Coherence will operate correctly. The first, called the 'cache-server', is a simple application that hosts and manages data on behalf of other applications in a cluster. The second, called the 'coherence shell', is a simple application that enables a developer to access, process and update cached data within a cluster, together with information about the said cluster.

By executing these applications on either a single host or several hosts, you can determine if Coherence is operating correctly locally or across a network.

When an application uses Coherence out-of-the-box, objects placed in to Coherence Caches are typically stored and managed in-process within the application.

However to increase availability of the said objects, typically to survive an application

outage (either deliberate or accidental), Coherence may manage objects in-memory but out of the application process, in what are called “Cache Servers”.

The purpose of a Coherence Cache Server is to manage application state, in a cluster, outside of the application process. Much like a Database server, but with out the requirement for storage.

This exercise ensures that Coherence-based applications that ship with Coherence may cluster together, hence ensuring Coherence will run as expected.

System Variables:

$COHERENCE_HOME$ = Path where Coherence was installed.

Steps:

1. Open a Command Prompt (on Windows) or a new Terminal or Shell (on Unix).

2. Change to the path where Coherence is installed $COHERENCE_HOME$

```
cd $COHERENCE_HOME$
```

3. Execute the Cache Server application, located in the Coherence bin folder.

On Windows:

```
bin\cache-server.cmd
```

On Unix:

```
bin/cache-server.sh
```

After a short delay of around 3 seconds (while Coherence attempts to locate a cluster to connect with) something like the following will be displayed.

```
java version "1.5.0_13"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05-237)
Java HotSpot(TM) Server VM (build 1.5.0_13-119, mixed mode)

2008-01-20 14:54:56.219 Oracle Coherence 3.3.1/389 <Info> (thread=main,
member=n/a): Loaded operational configuration from resource "jar:file:
…/lib/coherence.jar!/tangosol-coherence.xml"
2008-01-20 14:54:56.224 Oracle Coherence 3.3.1/389 <Info> (thread=main,
member=n/a): Loaded operational overrides from resource "jar:file:
…/lib/coherence.jar!/tangosol-coherence-override-dev.xml"

Oracle Coherence Version 3.3.1/389
Grid Edition: Development mode
Copyright (c) 2000-2007 Oracle. All rights reserved.

2008-01-20 14:54:56.531 Oracle Coherence GE 3.3.1/389 <Info> (thread=main,
member=n/a): Loaded cache configuration from resource "jar:file:
…/lib/coherence.jar!/coherence-cache-config.xml"

. . .

2008-01-20 14:55:00.679 Oracle Coherence GE 3.3.1/389 <Info> (thread=main,
member=1): Started DefaultCacheServer...
```

```
SafeCluster: Name=n/a

Group{Address=224.3.3.1, Port=33389, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=1, Timestamp=2008-01-20 14:54:56.959,
Address=172.16.154.1:8088, MachineId=9729, Location=process:
251@Macintosh-2.local)
  OldestMember…
  ActualMemberSet…
    )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

Services
  (
  TcpRing…
  DistributedCache…
  ReplicatedCache…
  Optimistic…
  InvocationService…
  )
```

Should the **Member Id** (in yellow above), be anything other than **1**, this means the Cache Server has clustered with one or more other Cache Servers on the network or running locally on your host. While this is default behavior for Coherence – to cluster with any other Coherence instances running locally or on a network – for these exercises it is strongly advised to restrict Coherence to your own host. To achieve this either disconnect from the network or disable networking on your host or perform the Exercise: Configuring Coherence to run on a Single Host.

Should any errors or exceptions occur when starting the Cache Server, your network settings may need to modified. Try each of the following one at a time, restarting the Cache Server between each attempt;

a. If connected of a VPN, disconnect from it. By default most VPN networks are not configured to permit multi-cast and some unicast traffic, and hence Coherence as it is configured out-of-the-box, may not work. Coherence can be configured to run across a VPN, but this requires some advanced settings. This will be covered later.

b. If running a firewall, configure it to allow the above specified addresses and ports.

c. If you are still experiencing problems, unplug or disconnect from all networks. This includes wireless and wired networks.

If all of the above fail, setup Coherence to run on a single host with the Exercise: Configuring Coherence to run on a Single Host.

Should you received the warning like the following:

```
UnicastUdpSocket failed to set receive buffer size to 1428 packets (2096304
bytes); actual size is 44 packets (65507 bytes). Consult your OS
documentation regarding increasing the maximum socket buffer size.
```

```
Proceeding with the actual value may cause sub-optimal performance.
```

You may continue to use Coherence for development. However, for testing, staging and production uses, this issue will need to be resolved. Resolution is platform dependant and beyond the scope of this document.

4. Open another Command Prompt (on Windows) or a new Terminal or Shell (on Unix).

5. Change to the path where Coherence is installed $COHERENCE_HOME$

```
cd $COHERENCE_HOME$
```

6. Execute the Coherence Shell application, located in the Coherence bin folder.

On Windows:

```
bin\coherence.cmd
```

On Unix:

```
bin/coherence.sh
```

After a short delay, something like the following will be displayed.

```
** Starting storage disabled console **
java version "1.5.0_13"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05-237)
Java HotSpot(TM) Server VM (build 1.5.0_13-119, mixed mode)

. . .

Oracle Coherence Version 3.3.1/389
 Grid Edition: Development mode
Copyright (c) 2000-2007 Oracle. All rights reserved.

. . .
2008-01-26 18:32:59.245 Oracle Coherence GE 3.3.1/389 <Info>
(thread=Cluster, member=n/a): This Member(Id=2, Timestamp=2008-01-26
18:32:59.048, Address=192.168.1.89:8089, MachineId=26969, Location=process:
1183@Urbanville.home, Edition=Grid Edition, Mode=Development, CpuCount=2,
SocketCount=2) joined cluster with senior Member(Id=1, Timestamp=2008-01-26
18:32:21.227, Address=192.168.1.89:8088, MachineId=26969, Location=process:
1167@Urbanville.home, Edition=Grid Edition, Mode=Development, CpuCount=2,
SocketCount=2)

. . .
SafeCluster: Name=n/a

Group{Address=224.3.3.1, Port=33389, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=2, Timestamp=2008-01-26 18:32:59.048,
Address=192.168.1.89:8089, MachineId=26969, Location=process:
1183@Urbanville.home)
  OldestMember=Member(Id=1, Timestamp=2008-01-26 18:32:21.227,
Address=192.168.1.89:8088, MachineId=26969, Location=process:
1167@Urbanville.home)
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2008-01-26 18:32:21.227,
```

```
Address=192.168.1.89:8088, MachineId=26969, Location=process:
1167@Urbanville.home)
    Member(Id=3, Timestamp=2008-01-26 18:32:59.048,
Address=192.168.1.89:8089, MachineId=26969, Location=process:
1183@Urbanville.home)
    )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

Services
  (
  TcpRing{TcpSocketAccepter{State=STATE_OPEN,
ServerSocket=192.168.1.89:8089}, Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0,
Version=3.3, OldestMemberId=1}
  )

Map (?):
```

7. Execute the following commands in the Coherence Shell

    help
    cache mycache
    get message
    put message “gudday”
    size
    get message
    put message “this is cool”
    get message
    remove message
    size
    get message
    put message “one more time”
    bye

Questions:

1. If you run multiple Coherence Shells at the same time and use the same cache (“mycache”), can each Coherence Shell observe changes made from another? Why?

2. If you close each of the Coherence Shells (“bye”), and then restart them, is data from the previous session still available? Why?

3. If you run another Coherence Cache Server and then kill the initial one started (using CTRL-C or by closing the Command Console Window), is data still available from the Coherence Shells? Why?

Exercise: Configuring Coherence to run on a Single Host

Objective:

Learn how to configure applications using Coherence to only cluster on a single host.

Duration:

15 Minutes

Introduction:

When developing applications in a team environment or where there are many Coherence applications running on a network, it's often useful or a requirement to ensure that an application you're developing using Coherence **does not cluster** with other applications. While removing or disabling a network connection is effective, it's often not desirable if an application additionally requires network access, ie: to connect to a Database or the Internet etc. Consequently you need to configure the Coherence network layer to restrict cluster traffic and this clustering.

This exercise outlines how to ensure Coherence applications do not cluster with other applications outside a single host.

Prerequisite Exercises:

Exercise: Testing a Coherence Installation

Other Prerequisites:

(none)

Knowledge:

The principle behind restricting Coherence clustering to a single host (or number of hosts), is to restrict the cluster discovery traffic.

The simplest way to ensure the Coherence cluster discovery network traffic does not leave a single host (or is contained within a single host), is to set the multi-cast packet time to live (ttl) to 0 using the following JVM parameter:

-Dtangosol.coherence.ttl = 0

Solution Files:

tangosol-coherence-override.xml

Steps:

1. Open the cache-server.sh (unix) or cache-server.cmd (windows) files located in $COHERENCE_HOME$/bin folder using your favorite text editor (on windows use Notepad)

2. Change the line:

JAVA_OPTS="-Xms$MEMORY -Xmx$MEMORY"

to read:

JAVA_OPTS="-Xms$MEMORY -Xmx$MEMORY –Dtangosol.coherence.ttl=0"

3. Apply the same changes to the coherence.sh (unix) and coherence.cmd (windows) files in the $COHERENCE_HOME%/bin folder.

4. For your application, change the run profile to additionally provide the JVM parameter;

   -Dtangosol.coherence.ttl=0

   Your Coherence-based clustering will now be restricted to a single host. Additionally when you use Coherence the logging should show a ttl of 0.

```
Group{Address=224.3.3.1, Port=33389, TTL=0}
```

5. Alternatively you can place the provided tangosol-coherence-override.xml at the front of your classpath. Coherence will use this file to override the default cluster multicast settings.

Exercise: Your first Coherence-based Java Application

Objective:

Develop a simple Java console-based application to access, update and removing simple types of information from a Coherence clustered cache.

Duration:

30 minutes.

Prerequisite Exercises:

Exercise: Testing a Coherence Installation

Other Prerequisites:

(none)

Knowledge:

Unlike Client-Server applications, where client applications typically "connect" and "disconnect" from a server application, Coherence-based Clustered applications simply "ensure" they are in a Cluster, after which they may use the services of the Cluster. More specially, Coherence-based applications typically do not "connect" to a Cluster of applications; they become part of the Cluster.

Useful Tip:

To change the level of logging produced by Coherence, use the following JVM parameter: -Dtangosol.coherence.log.level=*level* where *level* is a number between 0 and 9. The default is 5. A value of 0 means no logging. A value of 9 means extremely verbose. A value of 3 is often useful enough for most application development.

Solution Files:

YourFirstCoherenceApplication.java

Steps:

1. Ensure you've completed the Exercise: Testing a Coherence Installation and leave both a Cache-Server and Coherence Console running.

2. In the Coherence Java Documentation (javadoc), shipped in the $COHERENCE_HOME$/lib folder, investigate the methods available on the CacheFactory class.

3. Develop a simple Java console application that uses the CacheFactory class to join a cluster (using the ensureCluster method) and then leave the cluster (using the shutdown method).

4. Extend your application to display (using the System.out.println) the Cluster object returned from the "ensureCluster" method.

5. Using javadoc, investigate the methods available on the “NamedCache” interface.

6. Extend your application to use the CacheFactory method “getCache” to acquire a NamedCache for the cache called “mycache” (the same cache name used in Exercise: Testing a Coherence Installation).

7. With the NamedCache instance, use the “get” method to retrieve the value for the key “message” (the same key used in the Exercise: Testing a Coherence Installation)

8. Output the value to the standard out using the System.out.println(…); method.

9. Using the running Coherence Shell, change the value of the key “message”. Re run your application to see the changed values.

10. Rerun your application with the JVM parameter “-Dtangosol.coherence.distributed.localstorage=false”. Is the output of the application different from previous executions?

11. Shutdown all of your Cache Server and Coherence Shell instances then rerun your application (with the above JVM parameter set). Is the output different?

Questions:

1. If you change the value of the “message” key in your application (using the “put”) method, is the new value available via the Coherence Shell?

2. What does the JVM parameter -Dtangosol.coherence.distributed.localstorage=false do to a Coherence Java process?

Exercise: Caching an Object (using Java Serialization)

Objective:

Create a simple domain object that can be placed into a Coherence Cache.

Duration:

30 minutes.

Prerequisite Exercises:

Exercise: Installing Coherence (Java Edition)

Other Prerequisites:

A solid understanding of the rules for Java Serialization.
http://java.sun.com/developer/technicalArticles/Programming/serialization/ has an overview.

Knowledge:

Placing any non-primitive Object into a Coherence Cache requires the said class to be Java Serializable, the reason being, such instances may need to be transported across process boundaries – ie: between Java Virtual Machines, across networks or processes on the same physical machine. The standard way to achieve such transport is to Serialize, transmit and Deserialize the object. Coherence requires Objects placed in a Cache to be Serializable.

Solution Files:

EndOfDayStockSummary.java

CacheAnObject.java

Steps:

1. Create a Java class called EndOfDayStockSummary that implements the java.io.Serializable interface to capture the open, high, low, close and adjusted close prices (in dollars) of a Stock (called a symbol) for a specific date together with the trading volume. Example attribute definitions for the class could be as follows;

   ```
   private String symbol;
   private long date;
   private double openPrice;
   private double highPrice;
   private double lowPrice;
   private double closePrice;
   private double adjustedClosePrice;
   private long volume;
   ```

2. Define a method called “getKey()” to return a String that is a combination of the

symbol and date attributes.

3. Declare a serialVersionUID.

4. Define a public default constructor – EndOfDayStockSummary()

5. Define a public constructor that accepts values for all of the attributes specified above.

6. Create a console application called CacheAnObject (which contains a main method) that creates an instance of an EndOfDayStockSummary class and places (using the NamedCache put method) the said instance into a Coherence cache called “dist-eodStockSummaries” and then retrieves it to display on the console.

Questions:

1. What happens if you forget to implement a default no arguments constructor for the EndOfDayStockSummary class? If you don’t know, comment it out and attempt to run your application again.

Exercise: Caching an Object (using ExternalizableLite)

Objective:

Understand how to improve the serialization performance of domain objects placed in Coherence using the proprietary ExternalizableLite interface.

Duration:

30 minutes.

Prerequisite Exercises:

Execute the Coherence Shell application, located in the Coherence bin folder.

Other Prerequisites:

(none)

Knowledge:

Standard Java Serialization performance is often a significant bottleneck for applications that communicate across process boundaries, especially those that depend on networks. Java Serialization also produces serialization streams that are often very verbose their binary format, typically containing much more information than that required by an application or encoded in such a manner that it's inefficient to send across a network, construct and destruct in a Java heap (memory) that consequently leads to increased garbage collection requirements.

To resolve some of these issues including; dramatically improving serialization performance, reducing the binary format size and the impact on garbage collection, Coherence provides an extension to the standard Java serialization interface – called ExternalizableLite.

Extending the java.io.Serializable interface, the com.tangosol.io.ExternalizableLite interface introduces two simple methods - readExternal and writeExternal -that permit a developer to explicitly read and write serialized object attributes from provided low-level byte-based DataInput and DataOutput streams respectively. By operating at the byte-level, instead of the Object level as do the standard Java Serializable and Externalizable interfaces, Coherence provides developers with greater control as to how Objects are serialized and deserialized, often leading to significantly (5 to 10x) smaller binary representations, faster serialization (10x) and less garbage collection.

To reduce the development effort to serialize and deserialize object attributes byte-at-a-time with the provided DataInput and DataOutput streams respectively, use the statically defined methods of the provided com.tangosol.util.ExternalizableHelper class.

Recommendations:

1. Whenever possible, use the methods of the com.tangosol.util.ExternalizableHelper class to implement the serialization and deserialization of object attributes in the readExternal and writeExternal methods respectively.

2. To safely serialize and deserialize Java Strings (that a represented in UTF), use the com.tangosol.util.ExternalizableHelper readSafeUTF and writeSafeUTF methods respectively.

Solution Files:

EndOfDayStockSummaryExternalizableLite.java

Steps:

1. Modify the EndOfDayStockSummary class from the Execute the Coherence Shell application, located in the Coherence bin folder. to implement the com.tangosol.io.ExternalizableLite interface.

2. Using the static methods defined on the com.tangosol.util.ExternalizableHelper class and the non-static methods defined on the DataInput and DataOutput streams, implement the readExternal and writeExternal methods of the com.tangosol.io.ExternalizableLite interface.

3. Modify your CacheAnObject.java class from the Execute the Coherence Shell application, located in the Coherence bin folder. to use the EndOfDayStockSummary that implemented ExternalizableLite.

Questions:

1. What happens if the order of reading object attributes in the readExternal method was different to the order in which the said attributes where written in the writeExternal method?

2. How would you serialize and deserialize a boolean attribute?

3. How would you serialize and deserialize a Java 5 Enumerated Type?

4. How would you serialize and deserialize nullable attribute?

5. Are all Java 1.5+ serialized objects compatible with Java 1.4.2?

6. How would you serialize an object that contains one of each of the following Java collections; a Linked List, a Tree Set and a Hashtable?

Exercise: Understanding Cached Object Semantics (are they Copies or References?)

Objective:

Understand the semantics of Coherence Caches, in particular whether they cache copies of objects or references to objects.

Duration:

30 minutes.

Prerequisite Exercises:

Exercise: Caching an Object (using ExternalizableLite)

Other Prerequisites:

Knowledge of the Java memory model, including the concept of object references and object cloning.

Knowledge:

The semantics of Coherence caches, in particular the type of object returned when calling "get" on the NamedCache interface may differ between the types of caches used. While this rarely causes any issues for developers, it's important to know if you make assumptions about what you're expecting to be returned from a cache.

Solution Files:

CachedObject.java

CachedObjectSemantics.java

Steps:

1. In the CachedObjectSemantics.java application, ensure the definition of the variable cacheName is as follows;

   String cacheName = "dist-myCache";

2. Execute the CachedObjectSemantics.java application. What is the output?

3. Change the definition of the variable cacheName to be as follows;

   String cacheName = "repl-myCache";

4. Execute the CachedObjectSemantics.java application. What is the output?

5. Change the definition of the variable cacheName to be as follows;

   String cacheName = "local-myCache";

6. Execute the CachedObjectSemantics.java application. What is the output?

7. Change the definition of the variable cacheName to be as follows;

   String cacheName = "near-myCache";

8. Execute the CachedObjectSemantics.java application. What is the output?

Questions:

1. Each call to the NamedCache get(…) method on a cache based on the distributed scheme returns a new copy of the originally cached object. Why?

2. Each call to the NamedCache get(…) method on a cache based on the replicated scheme returns a reference to the originally cached object (in the JVM). Why?

3. Explain the behavior of the NamedCache get(…) method for caches based on a near scheme.

Exercise: Loading Data into a Cache

Objective:

Learn how to populate a Coherence Cache with domain objects read from text files. Additionally discover the most efficient method of loading data into a Cache (in sequence).

Duration:

2 Hours

Prerequisite Exercises:

Exercise: Caching an Object (using Java Serialization)

Exercise: Caching an Object (using ExternalizableLite)

Other Prerequisites:

A knowledge of reading and parsing text files using BufferedReaders, the String.split method and Date parsing with a SimpleDateFormat.

Knowledge:

1. To open and close a text file in the variable called *filename* with a java.io.BufferedReader, use the following;

   BufferedReader in = new BufferedReader(new FileReader(filename));

   your reading code here

   in.close();

2. To split a String into an array of sub-strings based on a delimiter (say a comma), use the following;

   String[] parts = stringToSplit.split(“,”);

3. To parse a Date, formatted as a String in the format “yyyy-MM-dd” into a java.util.Date instance, use the following;

   SimpleDateFormat sdf = new SimpleDateFormat(“yyyy-MM-dd”);

   Date date = sdf.parse(dateAsAString);

Solution Files:

CacheLoading.java

StopWatch.java

EndOfDayStockSummary.java

EndOfDayStockSummaryExternalizableLite.java

The folder “endofdaystocksummaries”

Steps:

1. Develop a console application called CacheLoading to load all of the end of day stock summaries contained in the folder “endofdaystocksummaries” into a single Coherence cache called “eodss”. Reuse the EndOfDayStockSummary class from the previous exercises.

2. Using the provided StopWatch class, determine how long it takes to “put” the summaries into the cache and how many may be “put” per second. Run your application using the following cluster configurations;

    a. No Cache Servers Running

    b. A Single Cache Server Running

    c. Two Cache Servers Running

    d. With the CacheLoading application configured with

    -Dtangosol.coherence.distributed.localstorage = false

3. Modify your application to use the ExternalizableLite implementation of the EndOfDayStockSummary class. What effect does this have on throughput?

4. Modify your application to use the NamedCache.putAll method instead of the NamedCache.put method. What effect does this have on throughput?

5. Complete the following table with your results;

CacheLoading Application Configuration	**No Cache Servers** **Throughput (per sec)**	**One Cache Server** **Throughput (per sec)**	**Two Cache Servers** **Throughput (per sec)**
Using the Java Serialization version of EndOfDaySummary			
Using the Java Serialization version of EndOfDaySummary with local storage disabled	N/A		
Using the ExternalizableLite version of EndOfDaySummary			
Using the ExternalizableLite	N/A		

version of EndOfDaySummary with local storage disabled			

6. [Advanced] Modify your application to use multiple threads or multiple CacheLoading instances to load the caches in parallel. Are there any performance improvements? In a large cluster would you expect to see any throughput improvement?

Questions:

1. Based on your results, what is the most efficient method of performing cache loading?

2. Using the StopWatch class, determine the disk IO cost for loading the EndOfDayStockSummary instances from disk. How does this compare with the cost to load the instances into Coherence? Would loading the instances in parallel (concurrently) effect the throughput?

3. How does multi-core CPU effect throughput?

4. Does re-running the CacheLoader many times without restarting Cache Servers affect the throughput results? If so, why?

Exercise: Observing Events in a Cache with MapListeners

Objective:

Use Coherence MapListeners to observe cache entry changes.

Duration:

30 Minutes

Prerequisite Exercises:

Exercise: Loading Data into a Cache

Other Prerequisites:

Knowledge of the use and declaration of anonymous inner classes in Java. More information can be found here:
http://java.sun.com/docs/books/tutorial/java/javaOO/nested.html

Knowledge:

Through the use of Coherence MapListeners and the standard Java Bean Event model, it is possible to observe a Cache as entries are inserted, updated or deleted – in real time.

To observe said changes in cache entries, simply register an implementation of the com.tangosol.util.MapListener class using the addMapListener(…) methods on the ObservableMap interface, implemented by instances of the NamedCache interface.

Solution Files:

EndOfDayStockSummaryMapListener.java

EndOfDayStockSummaryAbstractMapListener.java

EndOfDayStockSummaryAbstractMapListenerForORCL.java

EndOfDayStockSummaryMultiplexingMapListener.java

Steps:

1. Using the Coherence Java documentation, investigate the methods available to register MapListeners on the ObservableMap interface.

2. Develop a Java console application that uses an anonymous inner class based on the MapListener interface to display the EndOfDayStockSummary objects as they are inserted. Start this application and then re-run the CacheLoading application from the previous exercise.

3. Develop a Java console application that uses an anonymous inner class based on the AbstractMapListener class to display the EndOfDayStockSummary objects as they are inserted. Start this application and then re-run the CacheLoading application from the previous exercise.

4. Develop a Java console application that uses an anonymous inner class based on the MultiplexingMapListener interface to display the EndOfDayStockSummary objects as they are inserted. Start this application and then re-run the CacheLoading application from the previous exercise.

5. Using the Coherence Java documentation, investigate the purpose and methods of the MapEventFilter.

6. Modify the AbstractMapListener version of the application to use a MapEventFilter to ensure only "insert" events are raised by Coherence in the application.

7. Modify the AbstractMapListener version of the application to use a MapEventFilter combined with an EqualsFilter to ensure only "insert" events for "ORCL" stocks are raised by Coherence in the application.

8. [Advanced] Modify the AbstractMapListener version of the application to use a MapEventFilter combined with an EqualsFilter to ensure only "insert" events for "ORCL" stocks where the close price is above $40.00 are raised by Coherence in the application.

Questions:

1. Does avoiding to override the onUpdate and onDelete events of the AbstractMapListener class ensure that such events are not delivered?

2. What is the purpose of "lite events"?

3. What impact does MapListeners have on the throughput of cache updates (observed from the CacheLoading times)?

4. What effect does increasing the number of threads on a distributed cache have on the throughput of MapListeners? Use the following JVM parameter to increase the number of threads (default workers is 0).

   -Dtangosol.coherence.distributed.threads=1

Exercise: Chat Application

Objective:

Develop a multi-user Chat application using Coherence.

Duration:

1 Hours

Prerequisite Exercises:

All previous exercises.

Hints:

1. Create a class to represent an individual message from a user in the chat. This class should encapsulate the attributes such as, who the message is from, the time the message was created, the text of the message (or content, that may include a file), potentially the subject or group in which the message is "destined".

2. Start simply, by allowing an end-user to start the application, specify their chat name and then accept messages for a single "chat group". The application will need to register a listener on the Cache that is being used to store messages.

3. [Advanced] Add a command in the chat application to list messages from a specified individual, sorted in the order in which the messages had been produced.

4. [Advanced] Add a command to allow a user to attach/send an arbitrary file to a message.

5. [Advanced] Add a command to remove all messages from a specified chat user.

6. [Advanced] Add a command to save all messages to disk. Further add a command to load (restore) existing messages from disk.

7. [Advanced] Add a command to count the number of messages from a specified chat user.