Oracle® Coherence

Tutorial for Oracle Coherence

Release 3.4.1

E14135-01

January 2009

ORACLE®

Oracle Coherence Tutorial for Oracle Coherence, Release 3.4.1

E14135-01

Primary Author: Thomas Pfaeffle

Contributing Author: Noah Arliss, Jason Howes, Mark Falco, Alex Gleyzer, Gene Gleyzer, Adam Leftik, David Leibs, Andy Nguyen, Brian Oliver, Patrick Peralta, Cameron Purdy, Jonathan Purdy, Everet Williams, Tom Beerbower, John Speidel

Contents

List of Examples

List of Tables

List of Figures

Preface

Oracle Coherence is an in-memory data grid solution that enables organizations to predictably scale mission-critical applications by providing fast access to frequently used data. Data grid software is a middleware that reliably manages data objects in memory across many servers. By automatically and dynamically partitioning data, Oracle Coherence enables continuous data availability and transactional integrity, even in the event of a server failure. Oracle Coherence provides organizations with a robust, scale-out data abstraction layer.

Developers can easily take advantage of the features of Coherence using the standard Java collections API to access and modify data, and use the standard JavaBean event model to receive data change notifications.

Audience

This book is targeted at software developers, architects, and administrators. It provides a brief overview of the Oracle Coherence data grid, installation, configuration, developing with, and finally deploying Oracle Coherence.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at 1.800.223.1711. Complete instructions for using the AT&T relay services are available at `http://www.consumer.att.com/relay/tty/standard2.html`. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Related Documents

For more information, see the following documents in the Oracle Coherence documentation set:

- *Getting Started with Oracle Coherence*
- *User's Guide for Oracle Coherence*
- *Developer's Guide for Oracle Coherence*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Installing Coherence and JDeveloper

This chapter describes how to install and set up your environment for running Oracle JDeveloper Studio Edition 11g and Oracle Coherence 3.4.x. This chapter contains the following sections:

- Downloading and Installing Coherence
- Downloading and Installing JDeveloper
- Testing a Coherence Installation

This chapter assumes that you have the privileges to install software and set system environment variables as the oracle user, an understanding of how to use a terminal window, including setting environment variables, and creating and moving between directories. It also assumes that you have a working installation of Java SE (JDK) version 1.6.

Downloading and Installing Coherence

To download and install Oracle Coherence:

1. Download Oracle Coherence 3.4.x to your desktop.

   Coherence (Java edition) ships as a single zip file, typically called `coherence-<version>.zip`. You can obtain Coherence 3.4.x from the following URL:

   `http://www.oracle.com/technology/products/coherence/index.html`

2. Extract the contents of the zip file to a directory named `C:\oracle\product`.

   The zip file contains the coherence directory with these subdirectories:

   - `bin`—contains command scripts
   - `lib`—contains the required library files
   - `examples`— contains the example code
   - `doc`—contains the product documentation

Downloading and Installing JDeveloper

To download and Install Oracle JDeveloper Studio Edition 11g:

1. Download Oracle JDeveloper Studio.

   You can obtain JDeveloper Studio 11g from the following URL:

```
http://www.oracle.com/technology/products/jdev/index.html
```

2. Run the JDeveloper Studio installer.

   Follow the prompts in the installation screens. If the installer asks for a **Middleware Home**, enter `C:\oracle\product` if it does not already exist.

   If you are asked for a **User Role** when JDeveloper starts, select **Default Role** to enable all technologies.

Testing a Coherence Installation

In this exercise, you test whether your Coherence installation can cluster Java processes together. This will ensure that Coherence-based applications that ship with Coherence will run as expected. If Coherence is not capable of clustering on a single machine, then you will need to reconfigure your network and firewall settings.

This exercise assumes that you have installed Oracle Coherence (Java Edition) 3.4.1 (See "Downloading and Installing Coherence" on page 1-1).

Coherence uses a variety of network addresses and ports to enable communication between clustered processes. If these addresses and/or ports are unavailable due to other applications using them, or because of a firewall, then Coherence may be unreliable, may fail to cluster, or may not work at all. By default, Coherence assumes that the network addresses and ports listed in Table 1–1 are available:

Table 1–1 Network Addresses and Ports Used by Coherence

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 and not the loop back address.)

Coherence ships with two simple command-line (shell-based) applications that can be used to determine whether Coherence will operate correctly.

- The "cache server," is a simple application that hosts and manages data on behalf of other applications in a cluster.
- The "coherence shell," is a simple application that enables a developer to access, process, and update cached data within a cluster. It also provides informaiton about the cluster. By executing these applications on either a single host or several hosts, you can determine whether Coherence is operating correctly locally or across a network.

  When an application uses Coherence out-of-the-box, objects placed into Coherence caches are typically stored and managed in-process within the application. However, to increase the availability of the objects, Coherence may manage objects in-memory but out of the application process. This allows objects to survive possible application outages (either deliberate or accidental). To manage objects in this way, Coherence uses "Cache Servers". The purpose of a Coherence cache server is to manage application state in a cluster outside the application process. It is much like a database server, but without the requirement for storage.

To set up and run the cache server and client:

1. Open a terminal window and verify that the `PATH` environment variable is set to include `\oracle\product\jdk160_05\bin`. If the `PATH` environment variable does not include `jdk\jdk160_05\bin` directory, then set the variable as follows:

a. Set the JAVA_HOME environment variable to the base of the JDK installation.

```
set JAVA_HOME=\oracle\product\jdk160_05
```

b. Include JAVA_HOME\bin in the PATH environment variable.

```
set PATH=%JAVA_HOME%\bin;%PATH%
```

2. Navigate to the directory where Coherence is installed. Edit cache-server.cmd and set the COHERENCE_HOME variable to point to the Coherence installation directory.

```
cd C:\oracle\product\coherence\bin
```

In cache-server.cmd, set the COHERENCE_HOME environment variable:

```
COHERENCE_HOME=C:\oracle\product\coherence
```

Save cache-server.cmd and close the file.

Example 1–1 illustrates cache-server.cmd with the edited value of COHERENCE_HOME.

Example 1–1 cache-server.cmd File with an Edited COHERENCE_HOME

```
@echo off
@
@rem This will start a cache server
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify the JVM heap size
set memory=512m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch

set java_opts="-Xms%memory% -Xmx%memory%"

"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar" com.tangosol.net.DefaultCacheServer %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\cache-server.cmd
goto exit

:exit
endlocal
@echo on
```

3. Execute the cache server application that is located in the `coherence\bin` directory.

   ```
   C:\oracle\product\coherence\bin>cache-server.cmd
   ```

   When you start the first cache server, there is a slight delay because the cache server looks for an existing cluster. When it determines that there are no clusters to join, it starts one. On startup, the cache server will produce output similar to the code in Example 1–2.

   Several important features are highlighted in the example:

   - the Java JDK version number: **`java version "1.6.0_05"`**
   - information about how configuration files are loaded. The default is to load from JAR: **`Loaded operational configuration from resource "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"`**
   - the Coherence version number: **`Oracle Coherence Version 3.4.1/407`**
   - the multicast address. This address changes with each Coherence version. Note the **`3.4.1`** in the address for Coherence version 3.4.1: **`Group{Address=224.3.4.1, Port=34407, TTL=4}`**
   - the Member Id indicates the number of members in your cluster. For the purposes of this exercise, the value should be 1. **`ThisMember=Member(Id=1 ...`**

Example 1–2 Output from Starting a Coherence Cache Server

```
C:\oracle\product\coherence\bin>cache-server.cmd
java version "1.6.0_05"
Java(TM) SE Runtime Environment (build 1.6.0_05-b13)
Java HotSpot(TM) Server VM (build 10.0-b19, mixed mode)

2008-12-09 14:29:11.968/1.391 Oracle Coherence 3.4.1/407 <Info> (thread=main, me
mber=n/a): Loaded operational configuration from resource "jar:file:/C:/oracle/p
roduct/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2008-12-09 14:29:11.983/1.406 Oracle Coherence 3.4.1/407 <Info> (thread=main, me
mber=n/a): Loaded operational overrides from resource "jar:file:/C:/oracle/produ
ct/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2008-12-09 14:29:11.983/1.406 Oracle Coherence 3.4.1/407 <D5> (thread=main, memb
er=n/a): Optional configuration override "/tangosol-coherence-override.xml" is n
ot specified
2008-12-09 14:29:11.983/1.406 Oracle Coherence 3.4.1/407 <D5> (thread=main, memb
er=n/a): Optional configuration override "/custom-mbeans.xml" is not specified
Oracle Coherence Version 3.4.1/407
 Grid Edition: Development mode
Copyright (c) 2000-2008 Oracle. All rights reserved.

2008-12-09 14:29:12.827/2.250 Oracle Coherence GE 3.4.1/407 <Info> (thread=main,
 member=n/a): Loaded cache configuration from resource "jar:file:/C:/oracle/prod
uct/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2008-12-09 14:29:14.093/3.516 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Service Cluster joined the cluster with senior service member n/a

2008-12-09 14:29:17.343/6.766 Oracle Coherence GE 3.4.1/407 <Info> (thread=Clust
er, member=n/a): Created a new cluster "cluster:0x23CB" with Member(Id=1, Timest
amp=2008-12-09 14:29:13.702, Address=130.35.99.248:8088, MachineId=49912, Locati
on=site:us.oracle.com,machine:tpfaeffl-pc,process:4440, Role=CoherenceServer, Ed
```

```
ition=Grid Edition, Mode=Development, CpuCount=1, SocketCount=1) UID=0x822363F80
000011E1DDE4966C2F81F98
2008-12-09 14:29:17.671/7.094 Oracle Coherence GE 3.4.1/407 <D5> (thread=Distrib
utedCache, member=1): Service DistributedCache joined the cluster with senior se
rvice member 1
2008-12-09 14:29:17.827/7.250 Oracle Coherence GE 3.4.1/407 <D5> (thread=Replica
tedCache, member=1): Service ReplicatedCache joined the cluster with senior serv
ice member 1
2008-12-09 14:29:17.843/7.266 Oracle Coherence GE 3.4.1/407 <D5> (thread=Optimis
ticCache, member=1): Service OptimisticCache joined the cluster with senior serv
ice member 1
2008-12-09 14:29:17.858/7.281 Oracle Coherence GE 3.4.1/407 <D5> (thread=Invocat
ion:InvocationService, member=1): Service InvocationService joined the cluster w
ith senior service member 1
2008-12-09 14:29:17.874/7.297 Oracle Coherence GE 3.4.1/407 <Info> (thread=main,
 member=1): Started DefaultCacheServer...

SafeCluster: Name=cluster:0x23CB

Group{Address=224.3.4.1, Port=34407, TTL=4}


MasterMemberSet
  (
  ThisMember=Member(Id=1, Timestamp=2008-12-09 14:29:13.702, Address=130.35.99.2
48:8088, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proces
s:4440, Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2008-12-09 14:29:13.702, Address=130.35.99
.248:8088, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proc
ess:4440, Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2008-12-09 14:29:13.702, Address=130.35.99.248:8088,
MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:4440, R
ole=CoherenceServer)
    )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

Services
  (
  TcpRing{TcpSocketAccepter{State=STATE_OPEN, ServerSocket=130.35.99.248:8088},
Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Vers
ion=3.4, OldestMemberId=1}
  DistributedCache{Name=DistributedCache, State=(SERVICE_STARTED), LocalStorage=
enabled, PartitionCount=257, BackupCount=1, AssignedPartitions=257, BackupPartit
ions=0}
  ReplicatedCache{Name=ReplicatedCache, State=(SERVICE_STARTED), Id=2, Version=3
.0, OldestMemberId=1}
  Optimistic{Name=OptimisticCache, State=(SERVICE_STARTED), Id=3, Version=3.0, O
ldestMemberId=1}
  InvocationService{Name=InvocationService, State=(SERVICE_STARTED), Id=4, Versi
on=3.1, OldestMemberId=1}
  )
```

> **Note:** By default, Coherence uses multicast only to search for cluster members. Multicast is not used for data transfer. If you cannot use or do not want to use multicast, this can be configured.

The output of `cache-server.cmd` indicates whether you have one or more members in your cluster. The value of Member ID should be equal to 1:

```
...
In MasterMemberSet
ThisMember=Member(Id should be equal to 1)
...
```

If Member ID is greater than 1, then multiple clusters are being formed in your subnet. For the purposes of these exercises, there should only be one member in the cluster. Follow the steps in "Restricting Coherence to Your Own Host" on page 1-12 to restrict Coherence to your own host.

4. Open another terminal window to run the cache client.

   Verify that the `PATH` environment variable is set to include `\oracle\product\jdk160_05\bin`. If the `PATH` environment variable does not include the `jdk160_05\bin` directory, then set the variable as follows:

   a. Set the `JAVA_HOME` environment variable to the base of the JDK installation.

      ```
      set JAVA_HOME=\oracle\product\jdk160_05
      ```

   b. Include `JAVA_HOME\bin` in the `PATH` environment variable.

      ```
      set PATH=%JAVA_HOME%\bin;%PATH%
      ```

5. Navigate to the `\oracle\product\coherence\bin` directory. Edit `coherence.cmd` and set the `COHERENCE_HOME` variable to point to the Coherence installation directory. Save and close the file.

   Example 1–3 illustrates the `coherence.cmd` file, with `COHERENCE_HOME=\oracle\product\coherence`.

Example 1–3 coherence.cmd File with an Edited COHERENCE_HOME

```
@echo off
@
@rem This will start a console application
@rem demonstrating the functionality of the Coherence(tm) API
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify if the console will also act as a server
set storage_enabled=false

@rem specify the JVM heap size
set memory=64m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions
```

```
if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch

if "%storage_enabled%"=="true" (echo ** Starting storage enabled console **) else
(echo ** Starting storage disabled console **)

set java_opts="-Xms%memory% -Xmx%memory%
-Dtangosol.coherence.distributed.localstorage=%storage_enabled%"

"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar" com.tangosol.net.CacheFactory %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\coherence.cmd
goto exit

:exit
endlocal
@echo on
```

6. Execute the `coherence.cmd` file to start the cache client. This application shows you the basic distributed cache functionality that is built within Coherence.

   ```
   coherence.cmd
   ```

   Example 1–4 illustrates the output from starting the cache client. Note the following features of the output:

   - the client is the second member of the cluster (the server is the first member): **`ThisMember=Member(Id=2,...`**
   - at the end of the output, you should see the **`Map(?)`** prompt

Example 1–4 Output from Starting the Coherence Cache Client

```
** Starting storage disabled console **
java version "1.6.0_05"
Java(TM) SE Runtime Environment (build 1.6.0_05-b13)
Java HotSpot(TM) Server VM (build 10.0-b19, mixed mode)

2008-12-09 19:35:54.952/0.719 Oracle Coherence 3.4.1/407 <Info> (thread=main, me
mber=n/a): Loaded operational configuration from resource "jar:file:/C:/oracle/p
roduct/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2008-12-09 19:35:54.968/0.735 Oracle Coherence 3.4.1/407 <Info> (thread=main, me
mber=n/a): Loaded operational overrides from resource "jar:file:/C:/oracle/produ
ct/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2008-12-09 19:35:54.968/0.735 Oracle Coherence 3.4.1/407 <D5> (thread=main, memb
er=n/a): Optional configuration override "/tangosol-coherence-override.xml" is n
ot specified
2008-12-09 19:35:54.968/0.735 Oracle Coherence 3.4.1/407 <D5> (thread=main, memb
er=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.4.1/407
 Grid Edition: Development mode
Copyright (c) 2000-2008 Oracle. All rights reserved.
```

```
2008-12-09 19:35:56.843/2.610 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Service Cluster joined the cluster with senior service member n/a
2008-12-09 19:35:56.999/2.766 Oracle Coherence GE 3.4.1/407 <Info> (thread=Clust
er, member=n/a): Failed to satisfy the variance: allowed=16, actual=125
2008-12-09 19:35:56.999/2.766 Oracle Coherence GE 3.4.1/407 <Info> (thread=Clust
er, member=n/a): Increasing allowable variance to 29
2008-12-09 19:35:57.264/3.031 Oracle Coherence GE 3.4.1/407 <Info> (thread=Clust
er, member=n/a): This Member(Id=2, Timestamp=2008-12-09 19:35:57.046, Address=13
0.35.99.248:8089, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-
pc,process:4716, Role=CoherenceConsole, Edition=Grid Edition, Mode=Development,
CpuCount=1, SocketCount=1) joined cluster "cluster:0x23CB" with senior Member(Id
=1, Timestamp=2008-12-09 18:30:10.436, Address=130.35.99.248:8088, MachineId=499
12, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:4148, Role=Coherence
Server, Edition=Grid Edition, Mode=Development, CpuCount=1, SocketCount=1)
2008-12-09 19:35:57.311/3.078 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Member 1 joined Service DistributedCache with senior member 1
2008-12-09 19:35:57.311/3.078 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Member 1 joined Service ReplicatedCache with senior member 1
2008-12-09 19:35:57.311/3.078 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Member 1 joined Service OptimisticCache with senior member 1
2008-12-09 19:35:57.311/3.078 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Member 1 joined Service InvocationService with senior member 1
SafeCluster: Name=cluster:0x23CB

Group{Address=224.3.4.1, Port=34407, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=2, Timestamp=2008-12-09 19:35:57.046, Address=130.35.99.2
48:8089, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proces
s:4716, Role=CoherenceConsole)
  OldestMember=Member(Id=1, Timestamp=2008-12-09 18:30:10.436, Address=130.35.99
.248:8088, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proc
ess:4148, Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2008-12-09 18:30:10.436, Address=130.35.99.248:8088,
MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:4148, R
ole=CoherenceServer)
    Member(Id=2, Timestamp=2008-12-09 19:35:57.046, Address=130.35.99.248:8089,
MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:4716, R
ole=CoherenceConsole)
    )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

Services
  (
  TcpRing{TcpSocketAccepter{State=STATE_OPEN, ServerSocket=130.35.99.248:8089},
Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Vers
ion=3.4, OldestMemberId=1}
  )

Map (?):
2008-12-09 19:35:58.280/4.047 Oracle Coherence GE 3.4.1/407 <D5> (thread=TcpRing
Listener, member=2): TcpRing: connecting to member 1 using TcpSocket{State=STATE
_OPEN, Socket=Socket[addr=/130.35.99.248,port=2115,localport=8089]}
```

```
Map (?):
```

7. Exercise the client application by executing the following Coherence commands in the Coherence shell:

   - Enter `help` to see the list of commands that are available.
   - Enter `cache mycache`.

     The cache `mycache` implements the `com.tangosol.net.NamedCache` interface. Each `NamedCache` can be thought of as a table. A cluster can have many named caches. Each `NamedCache` holds one type of object. It can be a simple object, such as a String, or a complex object that you define.

     Example 1–5 illustrates that using the default configuration files (`coherence-cache-config.xml`) within the supplied `coherence.jar` file, a `NamedCache` called `mycache` is created using the distributed scheme:

Example 1–5 Output from Starting a Coherence Cache

```
Map (?): cache mycache
2008-12-09 19:46:02.889/608.656 Oracle Coherence GE 3.4.1/407 <Info> (thread=mai
n, member=2): Loaded cache configuration from resource "jar:file:/C:/oracle/prod
uct/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2008-12-09 19:46:03.421/609.188 Oracle Coherence GE 3.4.1/407 <D5> (thread=Distr
ibutedCache, member=2): Service DistributedCache joined the cluster with senior
service member 1
2008-12-09 19:46:03.452/609.219 Oracle Coherence GE 3.4.1/407 <D5> (thread=Distr
ibutedCache, member=2): Service DistributedCache: received ServiceConfigSync con
taining 258 entries
<distributed-scheme>
  <!--
  To use POF serialization for this partitioned service,
  uncomment the following section
  <serializer>
  <class-
  name>com.tangosol.io.pof.ConfigurablePofContext</class-
  name>
  </serializer>
  -->
  <scheme-name>example-distributed</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>example-binary-backing-map</scheme-ref>
    </local-scheme>
  </backing-map-scheme>
  <autostart>true</autostart>
</distributed-scheme>

Map (mycache):
```

     Execute the following commands at the `Map(?)` prompt in the Coherence shell:

   - `get message`—retrieves the value for the given key from the named cache
   - `put message "hello"`—puts a name/value pair (key and value) into the cache. This command always returns the last value assigned to the key.
   - `list`—displays the contents of the named cache

- `size`
- `get message`
- `put message "second message"`
- `get message`
- `remove message`—removes a key value from the cache
- `size`
- `get message`
- `put message "hi"`
- `bye`

Example 1–6 illustrates the output of each of these commands.

Example 1–6 Exercising Coherence Commands

```
Map (mycache): get message
null

Map (mycache): put message "hello"
null

Map (mycache): list
message = hello

Map (mycache): size
1

Map (mycache): get message
hello

Map (mycache): put message "second message"
hello

Map (mycache): get message
second message

Map (mycache): remove message
second message

Map (mycache): size
0

Map (mycache): get message
null

Map (mycache): put message "hi"
null

Map (mycache): bye
2008-12-10 11:17:18.921/56484.688 Oracle Coherence GE 3.4.1/407 <D5> (thread=Dis
tributedCache, member=2): Service DistributedCache left the cluster
2008-12-10 11:17:19.061/56484.828 Oracle Coherence GE 3.4.1/407 <D5> (thread=Clu
ster, member=2): Service Cluster left the cluster

C:\oracle\product\coherence\bin>
```

8. Open another terminal window and set the `PATH` environment variable to include `\oracle\product\jdk160_05\bin`. and the `JAVA_HOME` variable to `\oracle\product\jdk160_05` (See Step 4).
9. Start a second instance of `coherence.cmd` in the new terminal window. Note that a message similar to the following will be displayed in the terminal window where the first client is running:

```
Map (mycache):
2008-12-10 11:51:15.468/66.407 Oracle Coherence GE 3.4.1/407  (thread=Cluster,
member=3): Member 4 joined Service DistributedCache with senior member 1
```

10. Use the `cache mycache` command in the new terminal to connect to the `NamedCache` called `mycache`. Try to get and put values in different sessions. notice that each client can observe changes made by the other client.
11. Close down one of the `coherence.cmd` consoles. Note that a message indicating that the member has left the cluster will appear on the other console. For example:

```
Map (mycache):
2008-12-10 12:00:09.139/600.078 Oracle Coherence GE 3.4.1/407 <D5>
(thread=Cluster, member=3): Member 4 left service DistributedCache with senior
member 1

Map (mycache):
2008-12-10 12:00:09.249/600.188 Oracle Coherence GE 3.4.1/407 <D5>
(thread=Cluster, member=3): MemberLeft notification for Member 4 received from
Member(Id=4, Timestamp=2008-12-10 11:50:51.601, Address=130.35.99.248:8090,
MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:5356,
Role=CoherenceConsole)
2008-12-10 12:00:09.249/600.188 Oracle Coherence GE 3.4.1/407 <D5>
(thread=Cluster, member=3): Member(Id=4, Timestamp=2008-12-10 12:00:09.249,
Address=130.35.99.248:8090, MachineId=49912,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:5356,
Role=CoherenceConsole) left Cluster with senior member 1
```

12. If you close each of the Coherence shells (`bye`), and then restart them, note that data from the previous session still available. This is because the data is held in the cache server.
13. If you run another Coherence cache server, and then kill the initial one that was started (using `Ctrl+C` or by closing the Command Console window), is data still available from the Coherence shells? Why?
14. Shut down both the `coherence.cmd` consoles and `cache-server.cmd`. Start `coherence.cmd`. Create a `NamedCache` called `test` using `cachetest`. Try to put in a value as before. What happens and why?
15. Start up a cache server by running the `cache-server.cmd` file from the `coherence\bin` directory. Try to put in a value again. What happens and why?
16. Shut down all the cache servers.

Troubleshooting Cache Server Clustering

If the value of Member ID in the `cache-server.cmd` output is anything other than 1, then this indicates that the cache server has clustered with one or more other cache servers or processes running Coherence. These servers or processes may be running on the network or running locally on your host. Though this is default behavior for Coherence—to cluster with other processes running Coherence locally or on a network—it is strongly advised that you restrict Coherence to your own host.

Restricting Coherence to Your Own Host

The output of `cache-server.cmd` indicates whether you have just one member in your cluster. The value of Member ID should be equal to 1:

```
...
In MasterMemberSet
ThisMember=Member(Id should be equal to 1)
...
```

If Member ID is greater than 1, it means that multiple clusters are being formed in your subnet. For the purposes of the exercises in this document, there should only be one member in the cluster.

To restrict Coherence to your own host:

1. Stop cache server by pressing `Ctrl+C`.
2. Create a directory called `backup` under `oracle\product\coherence\lib`.

   ```
   cd C:\oracle\product\coherence\lib
   mkdir backup
   ```

3. Copy `coherence.jar` from the `oracle\product\coherence\lib` directory to the `backup` directory.

   ```
   cp coherence.jar C:\oracle\product\coherence\lib\backup\coherence.jar
   ```

4. Navigate to the `oracle\product\coherence\lib\backup` directory.
5. Extract `tangosol-coherence.xml` from `coherence.jar`.

   ```
   jar -xvf coherence.jar tangosol-coherence.xml
   ```

6. Edit the `tangosol-coherence.xml` file.

   Change the multicast listener port to a unique value, for example, `34408`. Be sure that your multicast listener port value that you are setting is unique. For example, if you share the port value `34407` with another user, then change your port value to 34408 and the other user can change to `34409`. When you are finished, save the file and quit the editor.

   Example 1–7 illustrates the `multicast-listener` fragment of `tangosol-coherence.xml` with its original port value of `34407`.

Example 1–7 Multicast-Listener Fragment of tangosol-coherence.xml File

```
...
    <multicast-listener>
      <address
system-property="tangosol.coherence.clusteraddress">224.3.4.1</address>
      <port system-property="tangosol.coherence.clusterport">34407</port>

      <!--
      Note: For production use, this value should be set to the lowest integer
      value that works. On a single server cluster, it should work at "0"; on
      a simple switched backbone, it should work at "1"; on an advanced backbone
      with intelligent switching, it may require a value of "2" or more. Setting
      the value too high can utilize unnecessary bandwidth on other LAN segments
      and can even cause the OS or network devices to disable multicast traffic.
      -->
      <time-to-live system-property="tangosol.coherence.ttl">4</time-to-live>
```

7. Append the modified `tangosol-coherence.xml` file to `coherence.jar` in the `backup` directory. Replace `coherence.jar` in the `lib` directory with the one in the `backup` directory.

```
jar -uvf coherence.jar tangosol-coherence.xml
cp coherence.jar \oracle\product\coherence\lib\coherence.jar
```

   When you execute `cache-server.cmd`, you should be able to view the modified port. Also, your cluster should now be restricted to your own host. If Membership ID still displays something other than `1`, then there may be additional issues with the installation. See "Advanced Steps to Restrict Coherence to Your Own Host" for more information.

Advanced Steps to Restrict Coherence to Your Own Host

If you follow the steps in the previous section and `cache-server.cmd` still fails to return a Member Id value of 1, then there may be additional issues you must resolve.

Disconnect from the network or disable networking on your host. If errors or exceptions occur when starting the cache server, your network settings may need to be modified. Try each of the following one at a time, restarting the cache server between each attempt:

- If connected to a VPN, disconnect from it. By default, most VPN networks are not configured to permit multicast and some unicast traffic. In this environment, Coherence may not work as it is configured out-of-the-box. Coherence can be configured to run across a VPN, but this requires some advanced settings.
- If you are running a firewall, configure it to allow the specified addresses and ports.
- If you still experience problems, unplug or disconnect from all the networks. This includes wireless and wired networks.
- If all the preceding options fail, set up Coherence to run on a single host.

2

Using JDeveloper with Coherence

This chapter describes how to set up JDeveloper to build and run Coherence-based Java applications.

Configuring Oracle JDeveloper for Coherence

To start up and configure JDeveloper for use with Coherence:

1. Open a terminal window and verify that the `PATH` environment variable is set to include `\oracle\product\jdk160_05\bin`. If the `PATH` environment variable does not include `jdk160_05\bin` directory, then set the variable as follows:

   a. Set the `JAVA_HOME` environment variable to the base of the JDK installation.

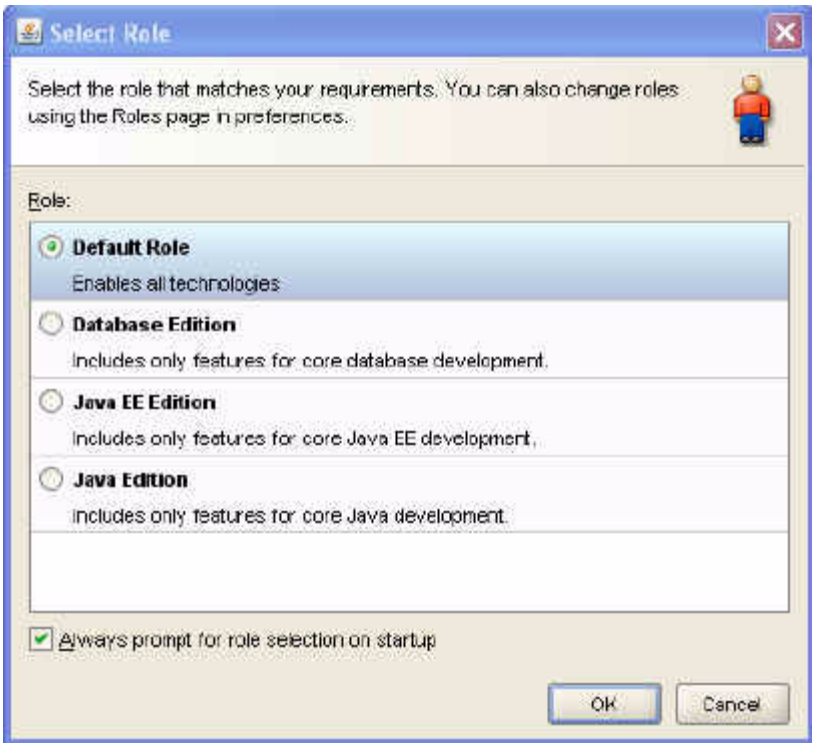
   ```
   set JAVA_HOME=\oracle\product\jdk160_05
   ```

   b. Include `%JAVA_HOME%\bin` in the PATH environment variable.

   ```
   set PATH=%JAVA_HOME%\bin;%PATH%
   ```

   c. Start JDeveloper. If you get a message asking you to select a role, select **Default Role**. If you get a message asking whether you want to migrate from previous versions of JDeveloper, select **No**. Close any window that displays the **Tip of the Day**."

Figure 2–1 Select Role Dialog Box

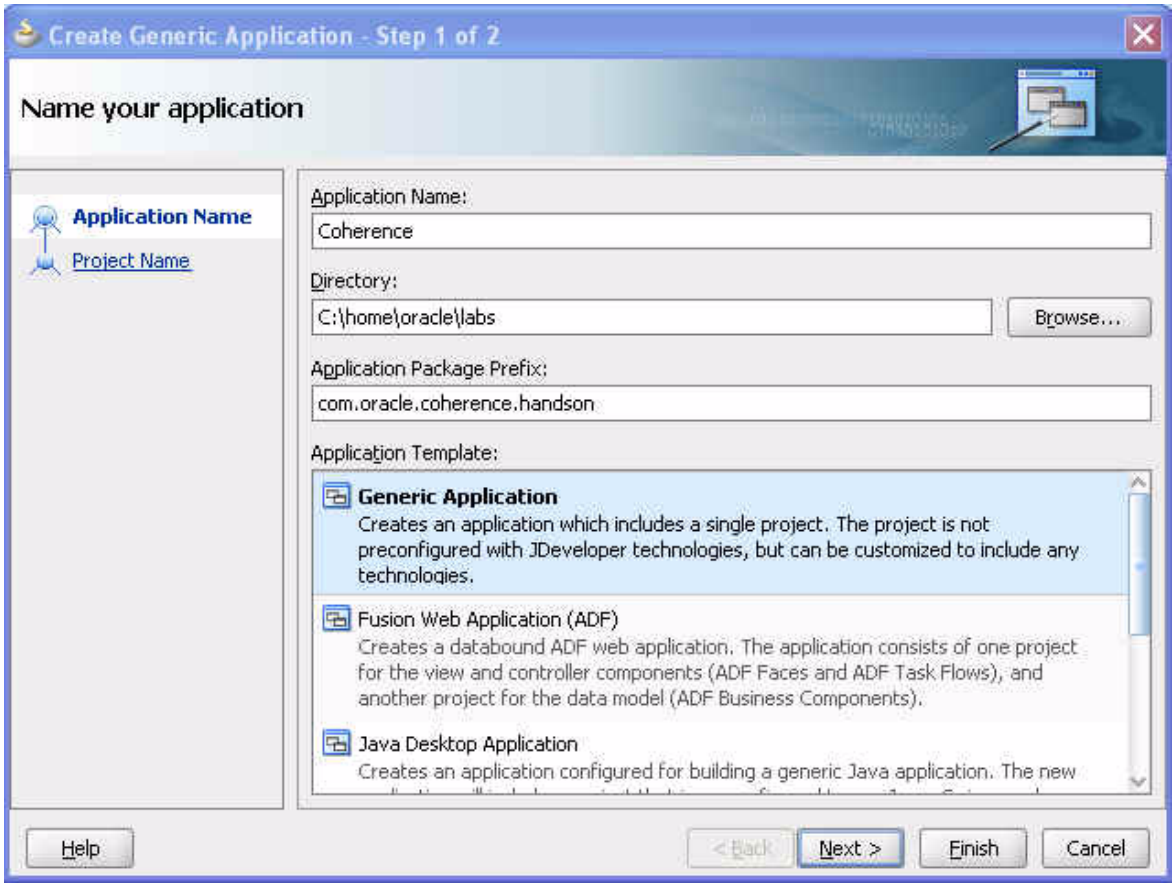


2. After JDeveloper starts up, click **New Application** to create an application.

   An application is a way of grouping projects or source code. This new application will hold the Coherence projects.

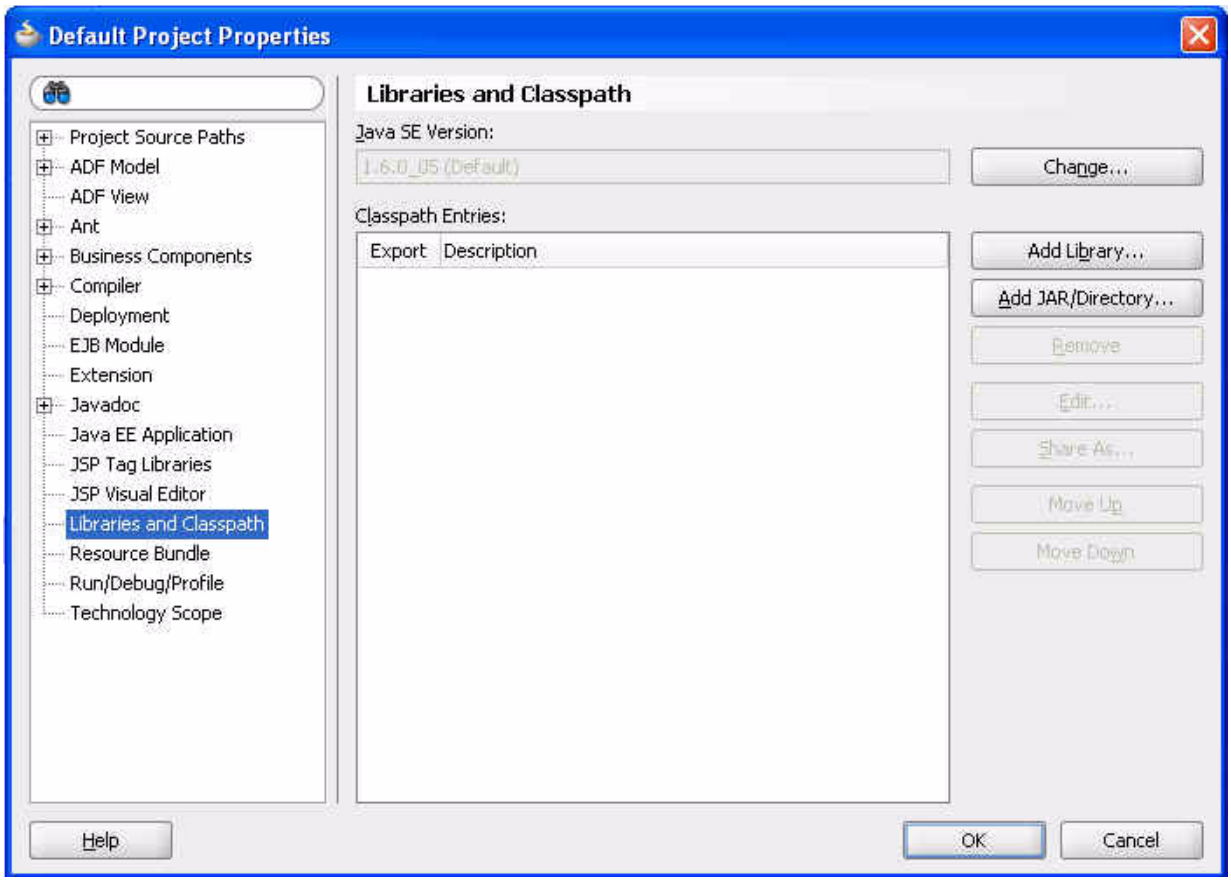
3. In the dialog box, change the application name to Coherence, the directory to `\home\oracle\labs`, and the application package prefix to `com.oracle.coherence.handson`. Click **OK**, and then click **Cancel** when prompted to create a new project.

Figure 2–2 Creating an Application in JDeveloper



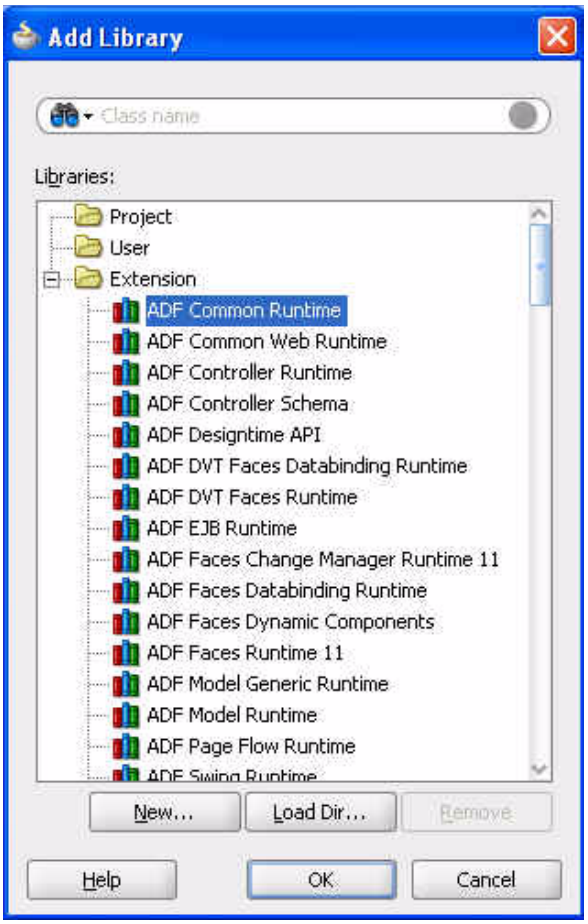
4. Add the Coherence JAR files to the default project properties. From the **Tools** menu, select **Default Project Properties**. The **Default Properties** dialog box opens.

Figure 2–3 Default Properties Dialog Box



5. Click **Libraries and Classpath**, and click the **Add Library** button. The **Add Library** dialog box opens.

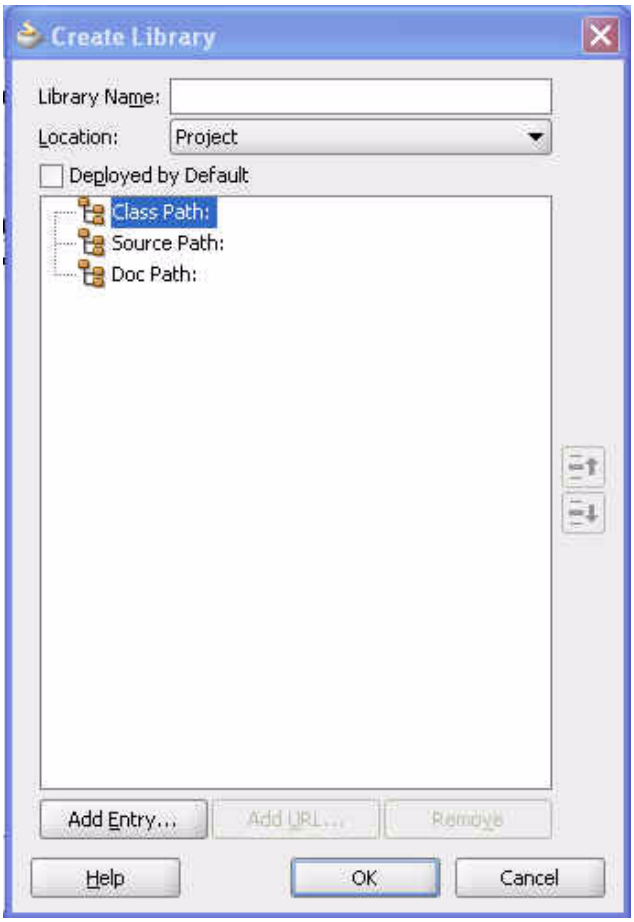
Figure 2–4 Add Library Dialog Box



6. In the **Add Library** dialog box, click **New**.

   The **Create Library** dialog box opens.

Figure 2–5 Create Library Dialog Box

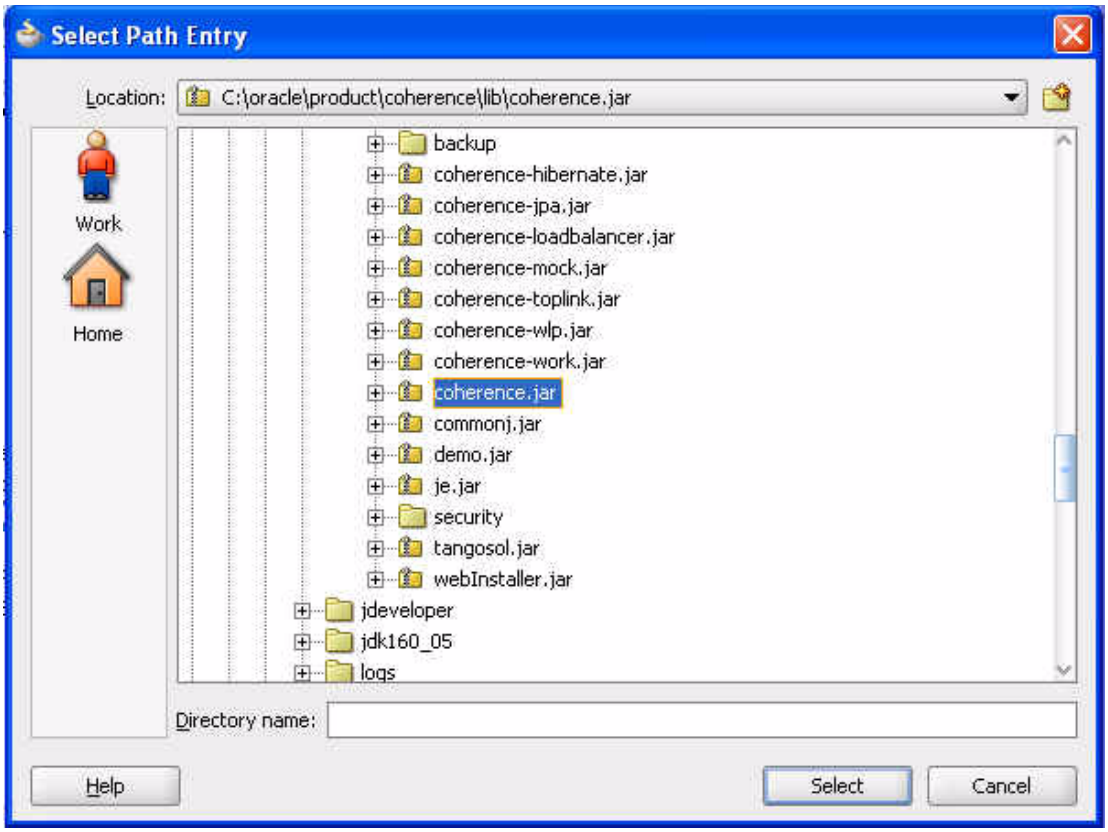


7. Click **Add Entry** to open the **Select Path Entry** dialog box.

   Find and expand the `coherence\lib` directory, which is under the `\oracle\product` directory.

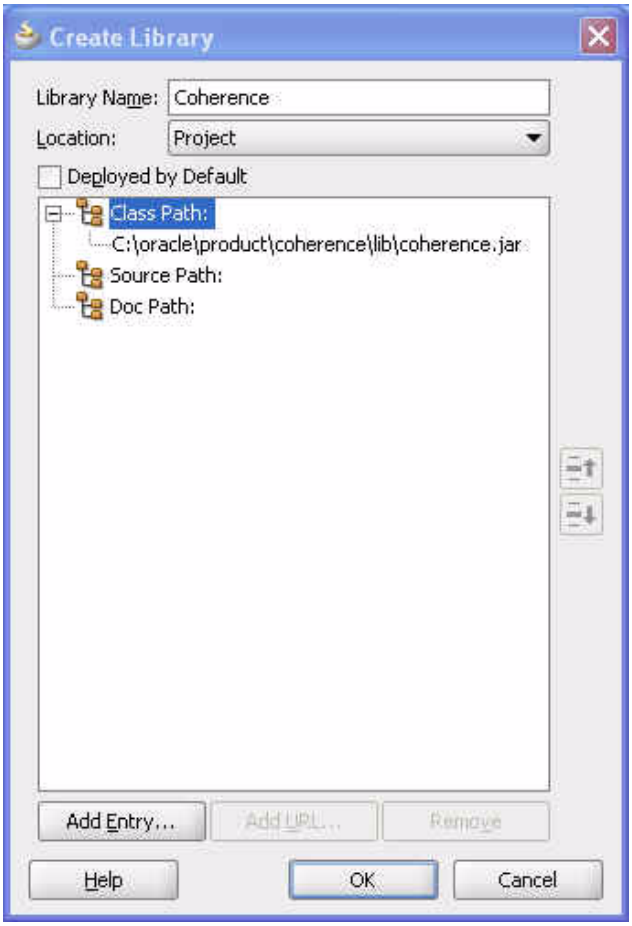
8. Highlight and select the `coherence.jar` file add it to the Classpath. Then, click **OK** to return to the **Create Library** dialog box.

Figure 2–6 Select Path Entry Dialog Box



9. In the **Create Library** dialog box, change the name in the **Library Name** field to **Coherence**. The contents of the dialog box should look like this:

Figure 2–7 Create Library Dialog Box with the Coherence Jar on the Classpath



10. Click **OK** in the **Create Library**, **Add Library**, and **Default Project Properties** dialog boxes to return to the JDeveloper IDE. Oracle JDeveloper is now set up with the correct Coherence libraries.

Accessing the Data Grid from Java

In this exercise, you develop a simple, Java console-based application to access, update, and remove simple types of information from a Coherence clustered cache. You also get familiar with using the Coherence Java APIs. In this practice, using JDeveloper, you perform the following:

- Create a new project
- Create a new `NamedCache`
- Put information into the cache and then retrieve it
- Retrieve information about the cache

All Coherence caches are named, have a lifetime scoped by the cluster instance in which they exist, and implement the `com.tangosol.net.NamedCache` interface. The `NamedCache` interface is an extension of `java.util.Map` and holds data and resources that are shared among the cluster members. Each `NamedCache` holds data as key/value pairs. Keys and values can be both simple and complex object types. The `NamedCache` interface provides extensions to the `Map` interface, such as locking and synchronization, storage integration, queries, event aggregations, and transactions.

The cache topology (or implementation) can be swapped or changed without code changes. Table 2–1 describes some of the more commonly used methods within the `NamedCache` interface:

Table 2–1 Methods in the NamedCache Interface

Method Name	Description
`voidclear()`	Removes all entries from the `NamedCache`
`boolean containsKey(Object key)`	Returns `true` if `NamedCache` contains an entry for the key
`boolean containsValue(Objectvalue)`	Returns `true` if there is at least one entry with this value in `NamedCache`
`Object get(Object key)`	Gets the entry from `NamedCache` for that key
`Object put(Object key,Objectvalue)`	Puts an object in `NamedCache` (blocking call)
`Object remove(Object key)`	Removes the mapping for this `key` from this map if present. Inherited from `ConcurrentMap`.
`Set entrySet()`	Returns a set of key/value pairs
`Collection values()`	Gets all values back as a collection
`CacheService getCacheService()`	Returns the `CacheService` that this `NamedCache` is a part of.

The `com.tangosol.net.CacheFactory` class is typically used to obtain an instance of a `NamedCache`. Table 2–2 describes some of the more commonly used methods in the `CacheFactory` class.

Table 2–2 Methods in the CacheFactory Class

Method Name	Description
`static Cluster ensureCluster()`	Obtains a cluster object running Coherence services
`static void shutdown()`	Shuts down all clustered services
`static NamedCache getCache(String cache)`	Returns an instance of a cache. Either joins an existing cache in the cluster or creates the cache if this is the first member.

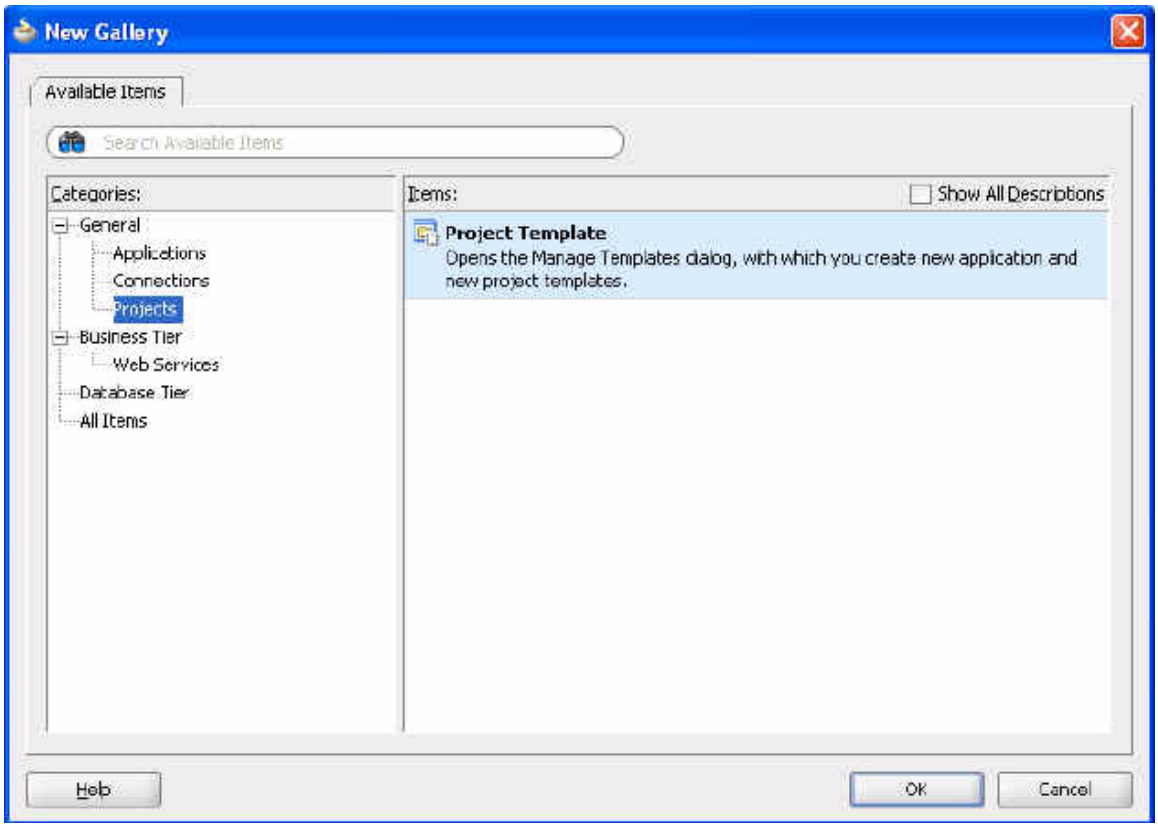
For a full list of methods, in the `NamedCache` interface and the `CacheFactory` class, see the Javadoc in `\oracle\product\coherence\doc`.

To create an application to access, update, and remove simple types of information from a Coherence clustered cache:

1. Create a new project in JDeveloper.
   a. Choose **File > New...** in the JDeveloper IDE. This will open the **New Gallery**.

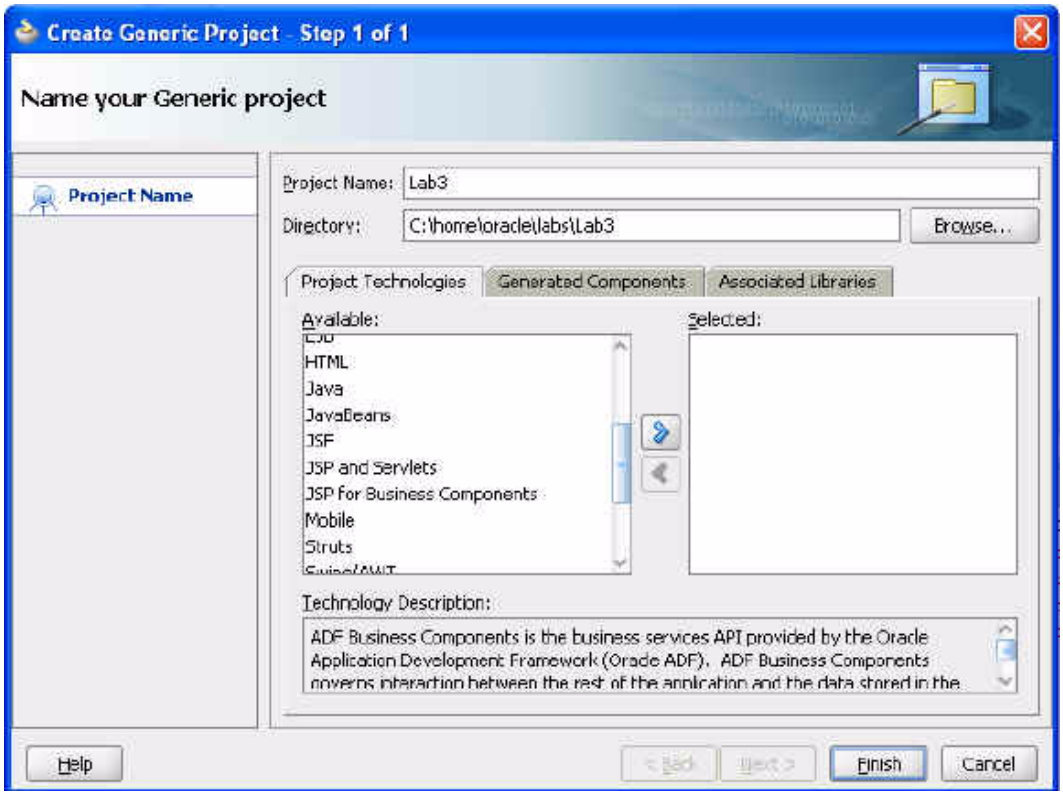
      Choose **General > Projects** in the **Categories** section of the **New Gallery** dialog box and **Generic Project** in the **Items** section. Click **OK**.

Figure 2–8 Choosing Projects in the New Gallery



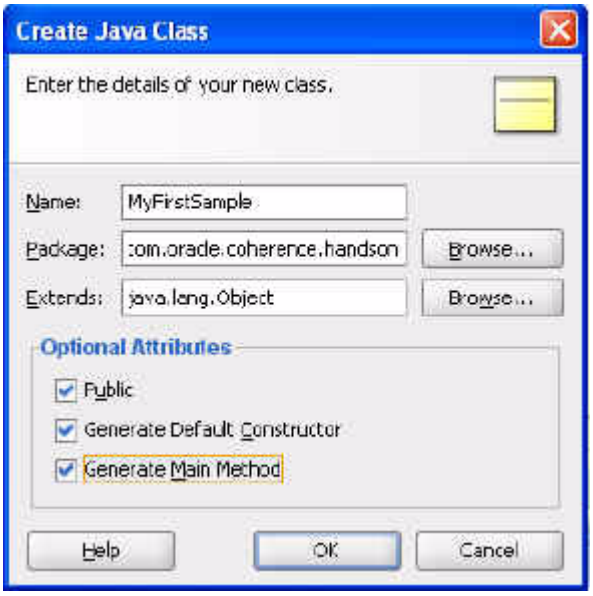
b. In the **Create Generic Project** dialog box, name the project `Lab3`. Ensure that directory is `C:\home\oracle\labs\Lab3`. Click **Finish**.

Figure 2–9 Providing Project Details



2. Create your first Coherence Java program.
   a. Right-click the `Lab3` project entry and select New.
   b. Select the **General** category and select **Java Class**. Click **OK**.
   c. Enter `MyFirstSample` as the class name and select the **Generate Main Method** check box. The dialog box should appear as follows. Click **OK**.

Figure 2–10 Providing Java Class Details



d. Right-click the `Lab3` project and choose Project Properties. Select **Libraries and Classpath** in the **Project Properties** dialog box. You should see the Coherence library that you defined earlier included here by default. Click **Cancel** to dismiss the dialog box.

e. In the editor for `MyFirstSample`, you must import the following components into your Java class:

```
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
```

f. Write the code to create a `NamedCache`, put in a value, and then verify the value that you put in. Save the file after you finish coding. Example 2–1 illustrates a sample program.

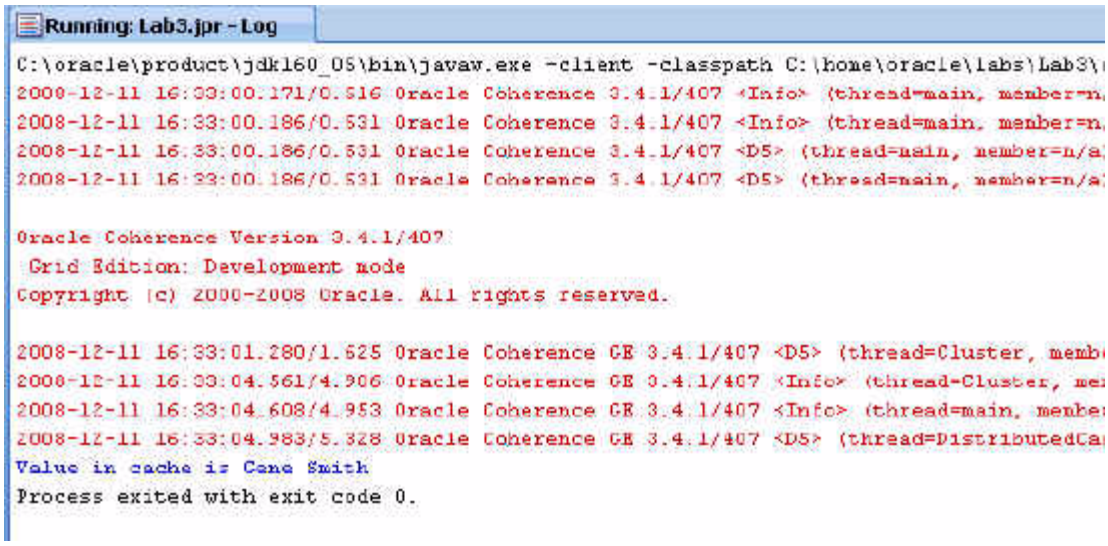
Example 2–1 Creating a NamedCache; Inserting and Verifying Values

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class MyFirstSample {
    public MyFirstSample() {
    }

    public static void main(String[] args) {

       // ensure we are in a cluser
       CacheFactory.ensureCluster();

       // create or get a named cache called mycache
       NamedCache myCache = CacheFactory.getCache("mycache");

       // put key, value pair into the cache.
       myCache.put("Name","Tim Middleton");

       System.out.println("Value in cache is " + myCache.get("Name"));

    }
}
```

g. Run the program in JDeveloper: right-click the `MyFirstSample.java` class in the editor and choose **Run**.

h. You should see messages similar to the following:

Figure 2–11 Output of Program to Create a NamedCache and to Store and Retrieve Values



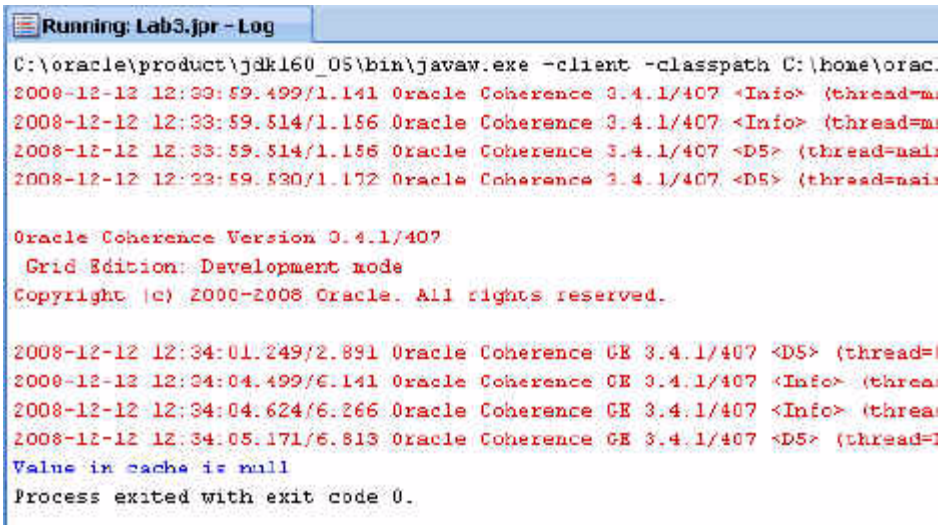
3. Now create another Java class only to get the value from your cache, rather than doing a put, and then a get. Name this Java class `MyFirstSampleReader`. Example 2–2 illustrates a sample program.

Example 2–2 Getting a Value from the Cache

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class MyFirstSampleReader {

    public MyFirstSampleReader() {
    }

    public static void main(String[] args) {
        // ensure we are in a cluser
        CacheFactory.ensureCluster();

        // create or get a named cache called mycache
        NamedCache myCache = CacheFactory.getCache("mycache");

        System.out.println("Value in cache is " + myCache.get("Name"));
    }
}
```

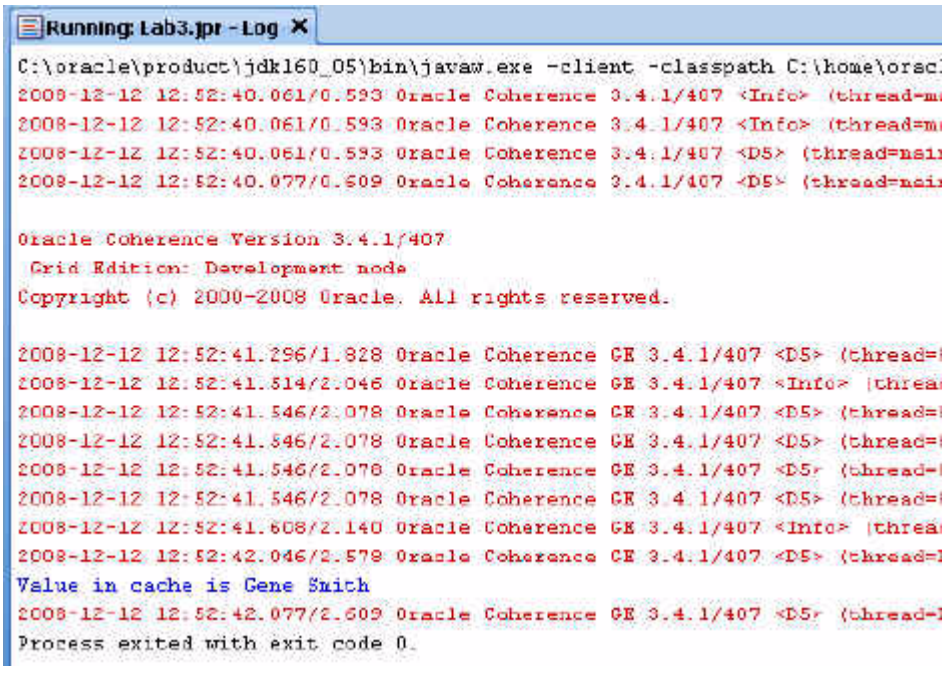
Run the `MyFirstSampleReader` class. Figure 2–12 illustrates the output from the program. Note that a null value returned. When the `MyFirstSample` class runs, it is storage-enabled by default unless otherwise specified. So, it creates a `NamedCache` when it starts, puts the value in, and when the class completes, the `NamedCache` disappears.

Figure 2–12 Output from the Sample Reader Program



4. Start up the `cache-server.cmd` that you used in "Testing a Coherence Installation" on page 1-2, run `MyFirstSample` to put the value in the `NamedCache`, and then run `MyFirstSampleReader` to read the value from the cache. Note the output illustrated in Figure 2–13: the "`Gene Smith`" value stored by `MyFirstSample` is returned by `MyFirstSampleReader`. By default, unless specified otherwise, all processes start up as storage-enabled. This means that the process can store data as part of the cluster.

Figure 2–13 Output from the Sample Reader Program with a Running Cache Server



This should not be the case for a process that joins the cluster only to perform an operation, such as putting in a value and then leaving, like `MyFirstSample`. You must modify the process so that it will not be storage-enabled.

This is done using the following Java command-line parameter.

```
-Dtangosol.coherence.distributed.localstorage
```

The value `-Dtangosol.coherence.distributed.localstorage=true` specifies that the process is storage-enabled. Setting the parameter to `false` specifies that the process is not storage-enabled.

a. Right-click the `Lab3` project and select **Project Properties**.

b. Choose **Run/Debug/Profile** and click **Edit**.

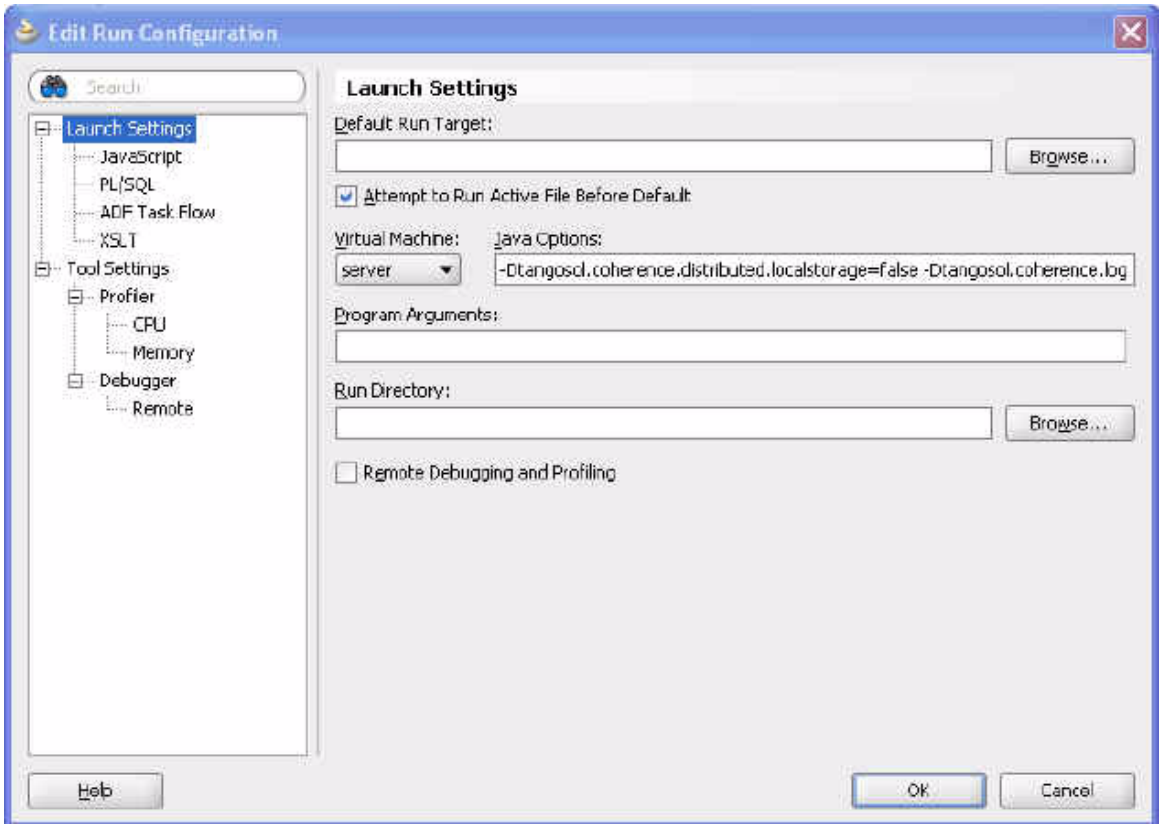
c. Ensure the **Virtual Machine** field is set to server.

d. Enter the following value in the Java Options field:

```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

The screen should look similar to the illustration in Figure 2–14.

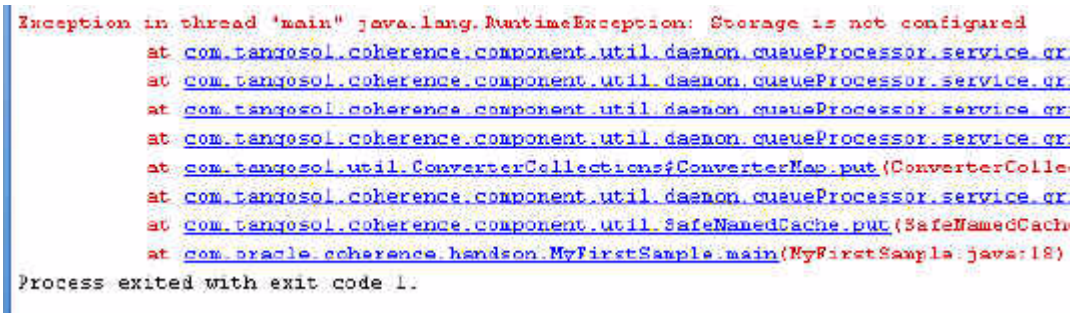
Figure 2–14 Editing Run Configuration Properties



e. Click **OK**, and then **OK** in the **Project Properties** dialog box save.

5. Shut down any running cache servers and rerun your `MyFirstSample` class. What happens? You receive a message similar to the one in Figure 2–15 indicating that storage is not enabled on the cluster, because you have set this member to be storage-disabled.

Figure 2–15 Output from JDeveloper if Storage is Disabled



6. Start the cache server again and retest. You should now see that the data is persisted between running the two Java examples.

Creating Your First Coherence-Based Java Application

In this practice you develop a simple Java console-based application to access update, and remove simple types of information from a Coherence clustered cache.

This practice assumes you have completed "Testing a Coherence Installation" on page 1-2. Make sure you have a cache server and a Coherence console up and running.

Unlike client/server applications, in which 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. Coherence-based applications typically do not "connect" to a cluster of applications; they become part of the cluster.

To change the level of logging produced by Coherence, use the following JVM parameter where `level` is a number between 0 and 9:

```
-Dtangosol.coherence.log.level=level
```

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.

To create a Java console-based application to access update, and remove simple types of information from a Coherence clustered cache:

1. In the Coherence Java documentation (Javadoc) that is shipped in the `\oracle\product\coherence\doc` directory, investigate the methods in the `CacheFactory` class.
2. Develop a simple Java console application (Java class) called `YourFirstCoherenceApplication` that uses the `CacheFactory` class to join a cluster (using the `ensureCluster` method), and then leave the cluster (using the `shutdown` method).
3. Using the Javadoc, investigate the methods that are available in the `NamedCache` interface.
4. Extend your application to use the `CacheFactory` method `getCache` to acquire a `NamedCache` for the cache called `mycache` (the same cache name used in the exercise: "Testing a Coherence Installation" on page 1-2).
5. 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" on page 1-2).

6. Output the value to the standard out using the `System.out.println(....)` method.

   Example 2–3 illustrates a sample Coherence-based Java application:

Example 2–3 Sample Coherence-Based Java Application

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class YourFirstCoherenceApplication {
public YourFirstCoherenceApplication() {
}

   public static void main(String[] args) {

      CacheFactory.ensureCluster();

      NamedCache myCache = CacheFactory.getCache("mycache");

      String message = (String)myCache.get("message");

      System.out.println(message);

      CacheFactory.shutdown();
YourFirstCoherenceApplication yourfirstcoherenceapplication = new
YourFirstCoherenceApplication();
   }

}
```

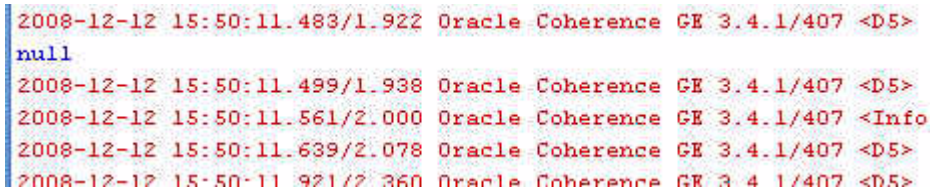
7. Ensure that you have the Coherence cache server and console up and running. In the console, connect to the `mycache` cache. For example:

   ```
   Map(?): cache mycache
   ```

8. Execute `YourFirstCoherenceApplication` from the JDeveloper IDE and view the result.

   Figure 2–16 illustrates the output from `YourFirstCoherenceApplication`. The output indicates that there is no data in the cache for the `message` key.

Figure 2–16 Output from Coherence-Based Java Application—No Value for the Key

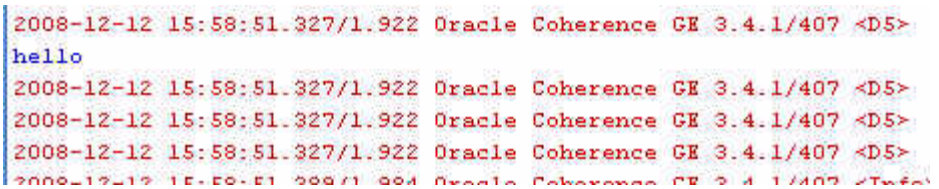


9. Using the running Coherence shell, change the key "`message`." For example, enter

   ```
   Map <mycache>: put message "hello"
   ```

   Rerun `YourFirstCoherenceApplication` from the JDeveloper IDE to see the changed values. Figure 2–17 illustrates that cache now holds the value `hello` for the key `message`.

Figure 2–17 Output from Coherence-Based Java Application—A Data Value for the Key



10. Rerun `YourFirstCoherenceApplication` with the following JVM parameter. Notice that the output is the same as the previous run.

    ```
    -Dtangosol.coherence.distributed.localstorage=false
    ```

11. Shut down your cache server and Coherence shell instances. Restart the cache server and then rerun `YourFirstCoherenceApplication` (with the preceding JVM parameter set). Note the output is now `null`.

12. If you change the value of the `message` key in your application (using the `put` method), is the new value available through the Coherence shell?

    a. For example, comment out the `get` method and add the `put` method.

    ```
    //String message = (String)myCache.get("message");
      String message = (String)myCache.put("message", "bye");
    ```

    b. Run `YourFirstCoherenceApplication`.

    c. Run the get command in the Coherence console.

    ```
    Map (mycache): get message
    ```

    You should see `bye` as the output.

13. Try setting the following JVM parameter and rerunning `YourFirstCoherenceApplication`. Note that it causes the application to return a null value since local storage is not enabled.

    ```
    -Dtangosol.coherence.distributed.localstorage=false
    ```

3

Leaving Strings Behind

In this chapter, you work with complex objects residing in the cache. Using JDeveloper, you will create a new `Person` class that is serializable, store and retrieve `Person` objects in the cache, and serialize your own objects.

This chapter contains the following sections:

Introduction

Until now you have been putting and getting `String` objects as the value in a `NamedCache`. Many of the implementations of the `get` and `put` methods in the Coherence Java API define the values and keys to be of type `Object`. For example:

```
public java.lang.Object get(java.lang.Object oKey)
public void put(java.lang.Object oKey, java.lang.Object oValue)
```

Any object can potentially be used as a value or key. This enables you to store complex objects as values in the cache.

Because Coherence may need to send the object across the wire, the object must be serializable. Object serialization is the process of saving an object's state into a sequence of bytes, and then rebuilding them into a live object at some future time. For example, objects that implement the `java.io.Serializable` interface are serializable.

Coherence supplies its own class for high-performance serialization called `com.tangosol.io.pof.PortableObject`. This is up to six times faster than the standard `Serializable` and the serialized result set is smaller.

Creating and Caching Complex Objects

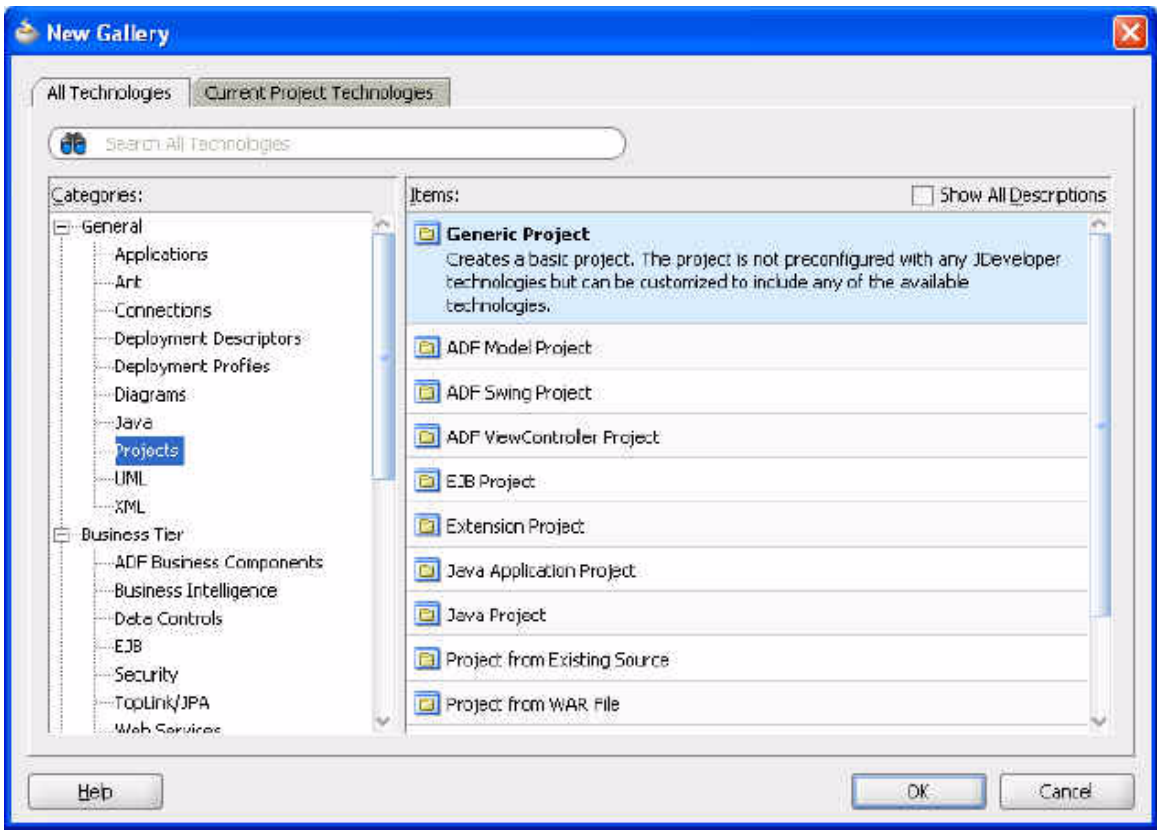
In this exercise you will create a `Person` object that has attributes of various data types, and a program that uses Java Serialization to put the object into the cache and retrieve it.

To create complex objects and place them in the cache:

1. Create a `Person` class to store in the cache.

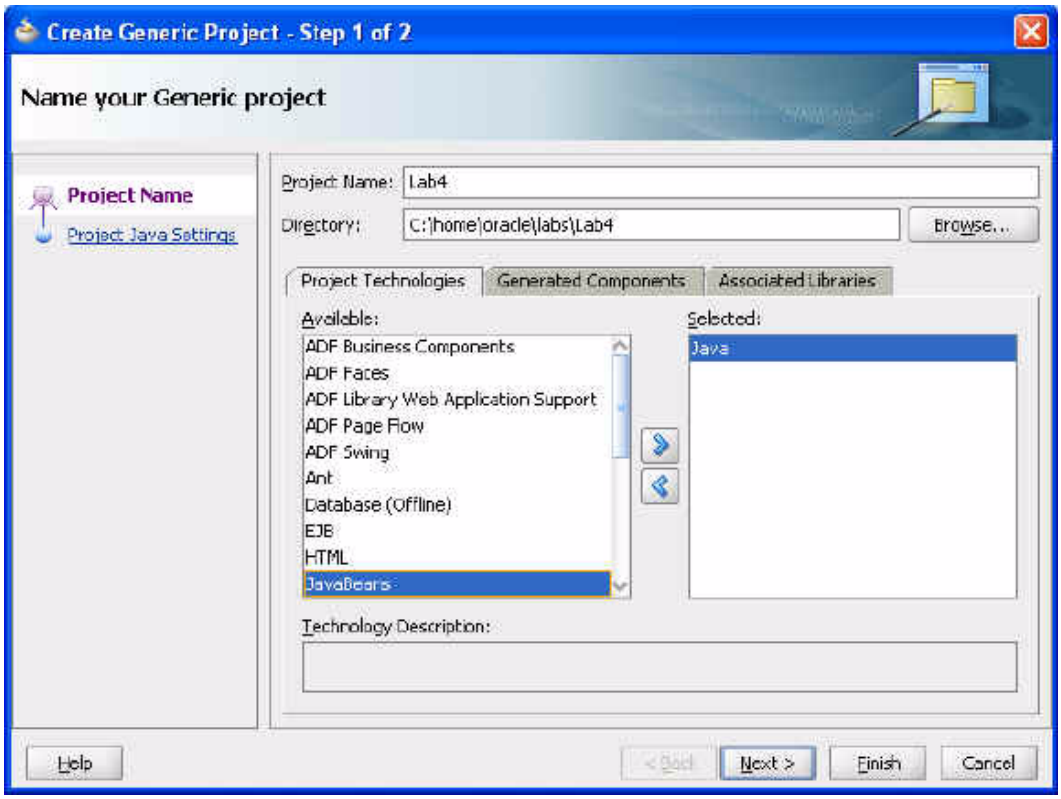
a. Create a new project in JDeveloper called `Lab4`. Hint: Right-click the Coherence application and choose **New...**. In the **New Gallery** choose **Projects** under **Categories** and **Generic Project** under **Items**. Click **OK**.

Figure 3–1 Creating a New Project in the New Gallery



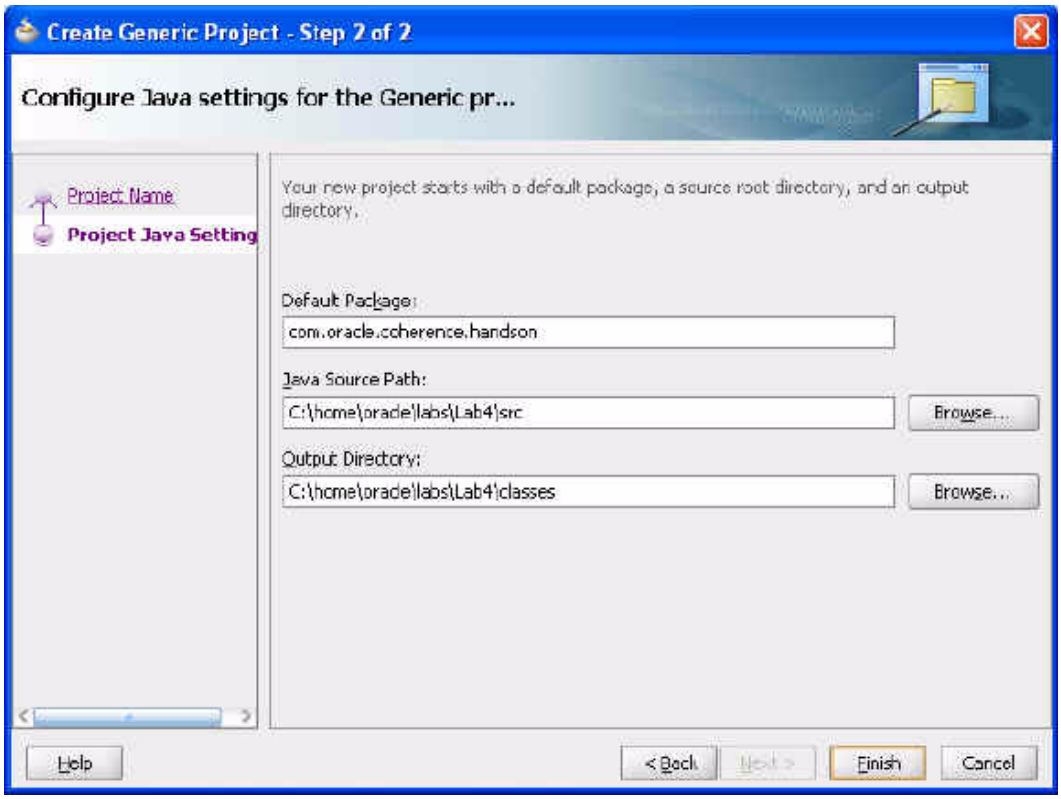
b. Name the project `Lab4` and select **Java** from **Project Technologies**. Click **Next**.

Figure 3–2 Naming the New Project



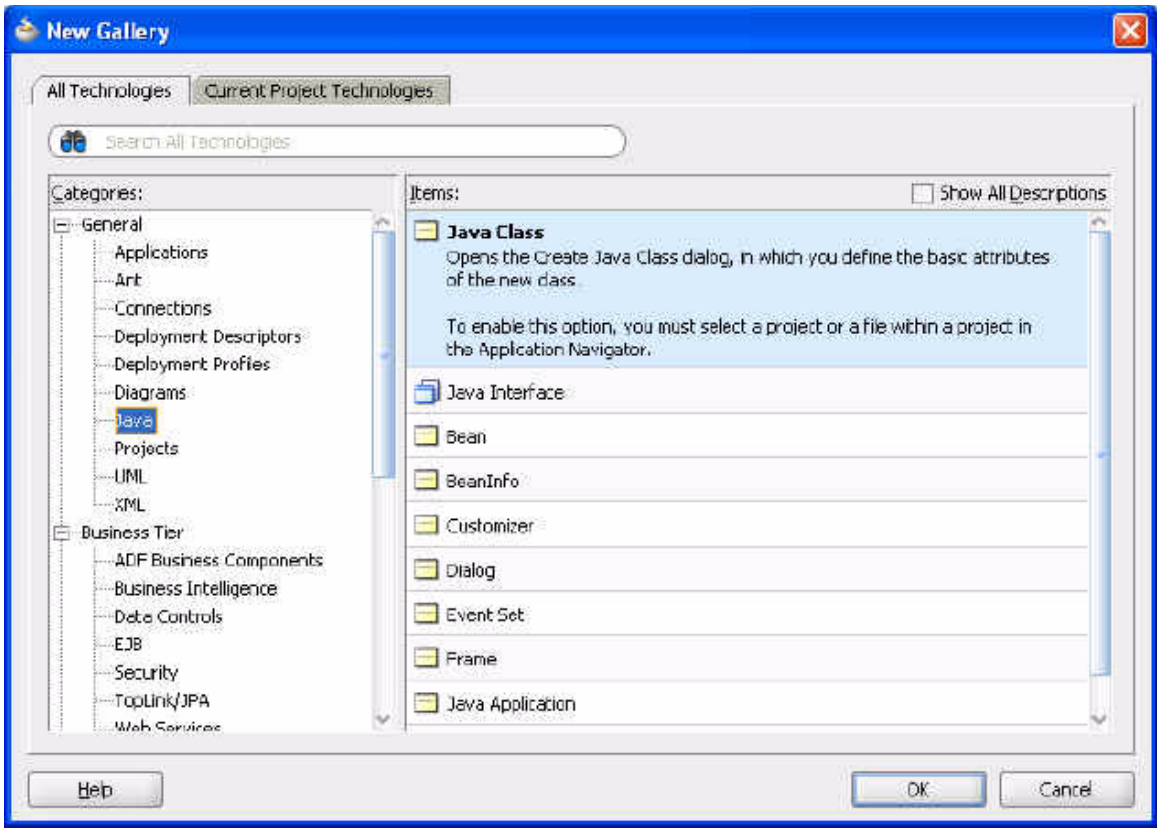
c. Ensure that the **Default Package** is `com.oracle.coherence.handson`, the **Java Source Path** is `C:\home\oracle\labs\Lab4\src` and the **Output Directory** is `C:\home\oracle\labs\Lab4\classes`. Click **Finish**.

Figure 3–3 Configuring Java Settings for the New Project



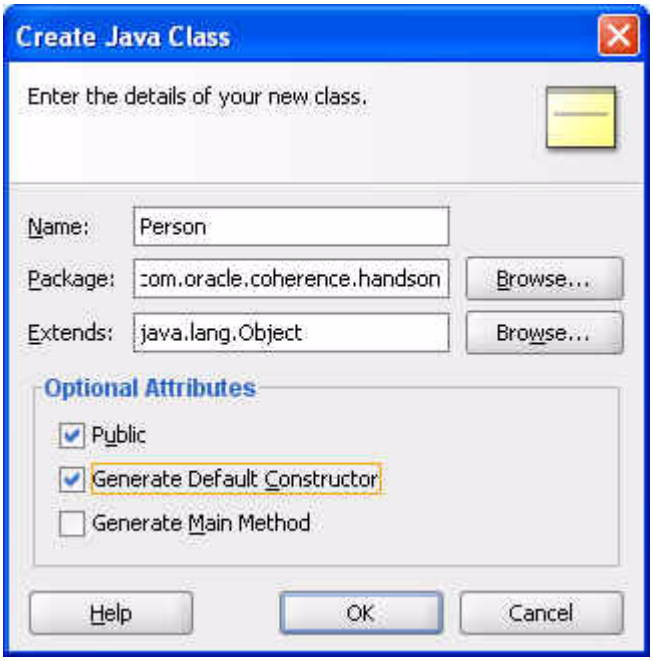
d. Create a new Java class called `Person`. Right click the `Lab4` project and select **New...**. In the **New Gallery**, choose **Java** under **General** and **Java Class** under **Items**. Click **OK**.

Figure 3–4 Creating the Java Class



e. Name the Java class `Person`. *Do not* select the **Generate Main Method** check box. Click OK.

Figure 3–5 Naming the Java Class



f. In the JDeveloper code editor, change your class to implement `java.io.Serializable`. Hint: Use `implements java.io.Serializable`. Add an `import` statement for the `java.io.Serializable` class.

g. Enter the following private attributes for your `Person` class. You can add others if you like.

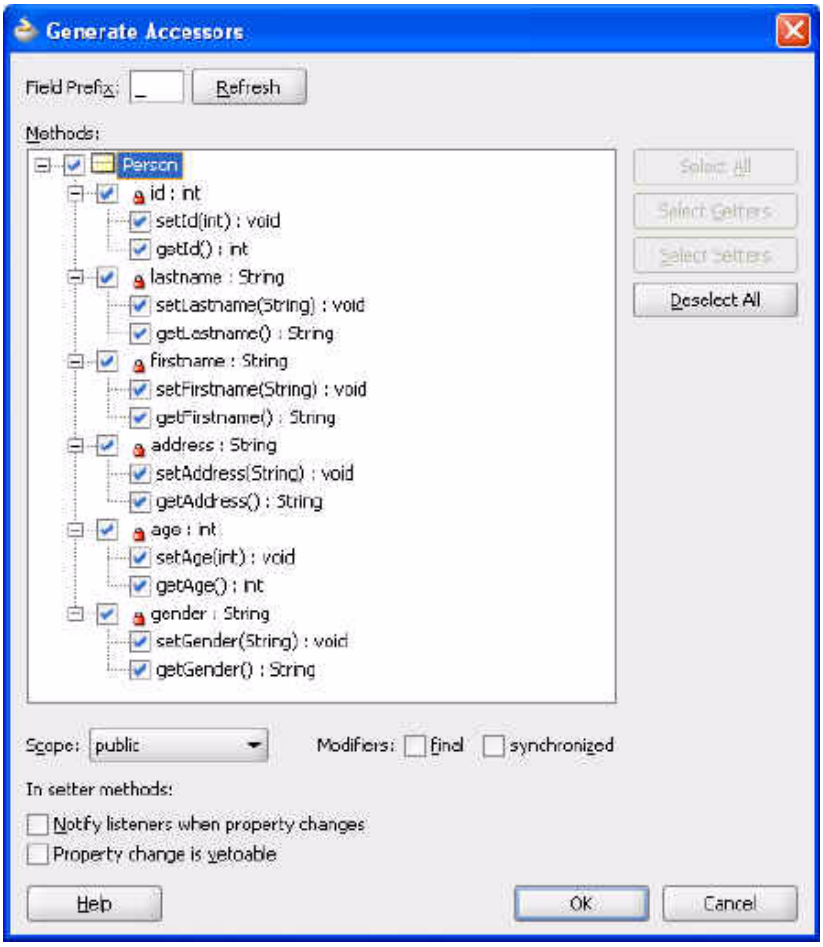
- `int id`
- `String lastname`
- `String firstname`
- `String address`
- `int age`
- `String gender`

The `Person` class should look similar to the image in Figure 3–6.

Figure 3–6 Person Class in the JDeveloper Code Editor

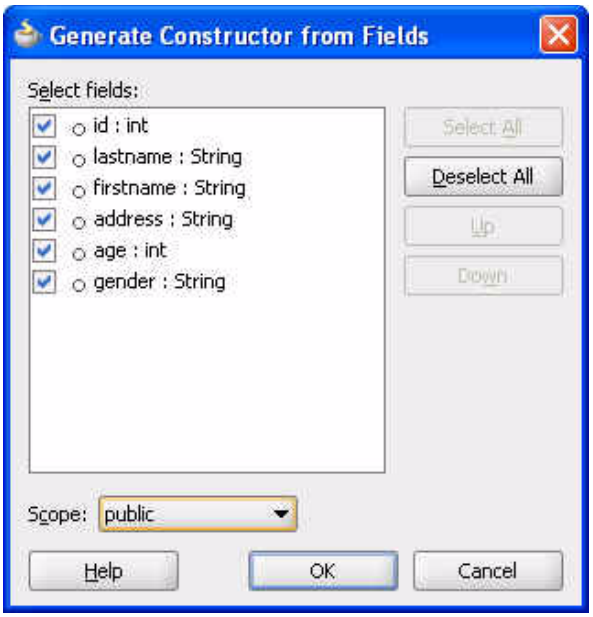
h. JDeveloper can generate the default get/set methods for your attributes. From the **Source** menu, select **Generate Accessors**. Select the check box next to the `Person` class. All of the attributes are now automatically selected. Click **OK** to continue.

Figure 3–7 Generating Accessors for the Java Class Attributes



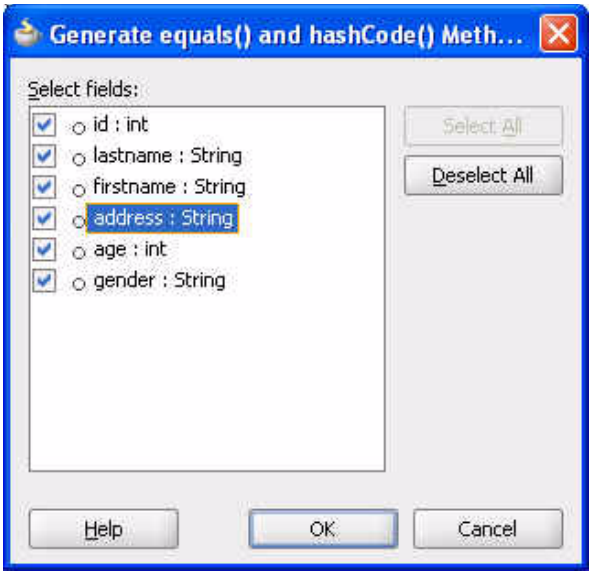
i. You can also generate the default constructor and equals methods automatically. From the **Source** menu, select **Generate Constructor from Fields**, click **Select All**, and then click **OK**.

Figure 3–8 Generating Constructors for the Java Class



j. From the **Source** menu, select **Generate equals () and hashCode()**, click **Select All**, and then click **OK**.

Figure 3–9 Generating equals() and hashCode() Methods

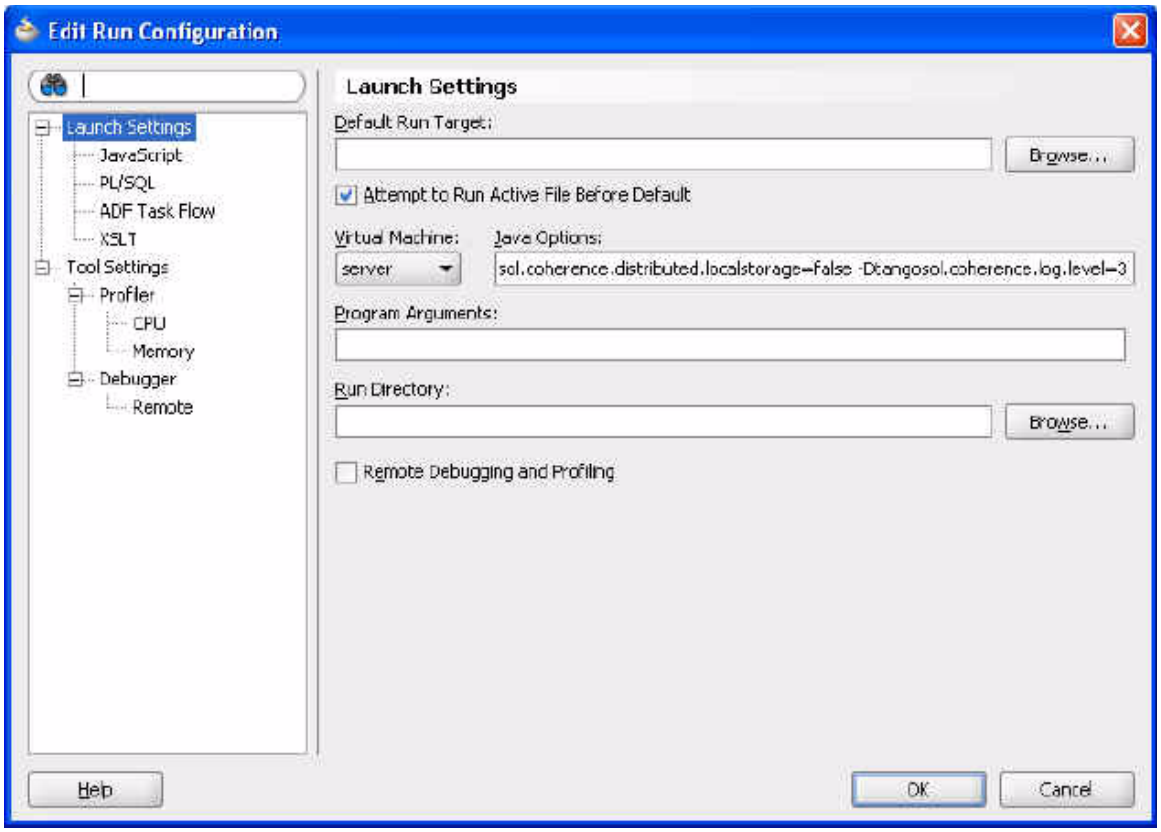


k. Change the Java process to be storage-disabled when you run it. right click the project and select **Project Properties.** In the **Project Properties** dialog box select **Run/Debug/Profile** and click **Edit**. Set **Virtual Machine** to **Server** and enter the following values for local storage and log level in the **Java Options** field of the **Edit Run Configuration** dialog box.

```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

Click **OK** in the **Edit Run Configuration** dialog box and **OK** in the **Project Properties** dialog box to return to the JDeveloper IDE.

Figure 3–10 Setting Java Runtime Options



2. Create a driver class called `PersonExample` to `put` and `get` your `Person` class.

   a. Create a new Java class called `PersonExample` in the Lab4 project. Right click the `Lab4` project and select **New...**. In the **New Gallery**, choose **Java** under **General** and **Java Class** under **Items**. Click **OK**.

   b. Name the Java class `PersonExample`. Select the **Generate Main Method** check box. Click **OK**.

Figure 3–11 Creating a Java Class with a main() Method

c. Create a new `NamedCache` called `person` and put a new instance of the `Person` class in it.

d. Get the `person` object from the cache and ensure that the two objects are identical. Example 3–1 illustrates a sample program.

Example 3–1 Comparing an Object from the Cache with the Original

```
public static void main(String[] args) {
   NamedCache person = CacheFactory.getCache("person");

   Person p1 = new Person(1,"Middleton","Tim",
   "Level 2, 66 Kings Park Road, West Perth",
   39,Person.MALE);

   person.put(p1.getId(),p1);

   Person p2 = (Person)person.get(p1.getId());

   if (p2.equals(p1)) {
   System.out.println("They are the same!!");
   }
}
```

Note: You will observe the following error: `Type or field 'MALE' not found in Person`. You will resolve this in the next step.

---

e. Open `Person.java` and add the following lines to initialize the gender variable.

```
static String MALE="M";
static String FEMALE="F";
```

Add these two lines right after where you declared these attributes:

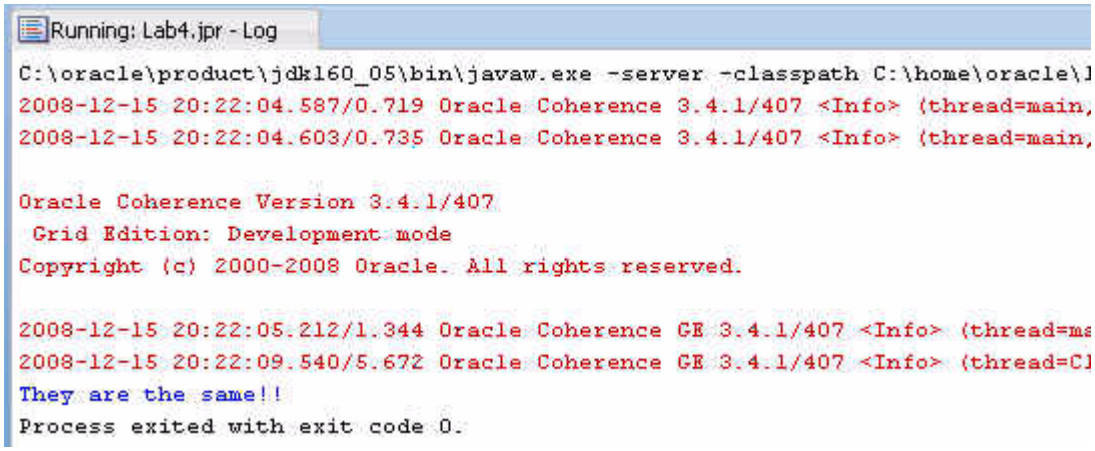
```
private int id;
private String lastname;
private String firstname;
private Stringaddress;
private intage;
private String gender;
```

f. Start the cache server.

```
C:\oracle\product\coherence\bin>cache-server.cmd
```

g. Execute `PersonExample` and view the result.

Figure 3–12 Results of Running PersonExample.java File



3. Implement faster serialization: change the `Person` class to implement the Coherence `PortableObject` class.

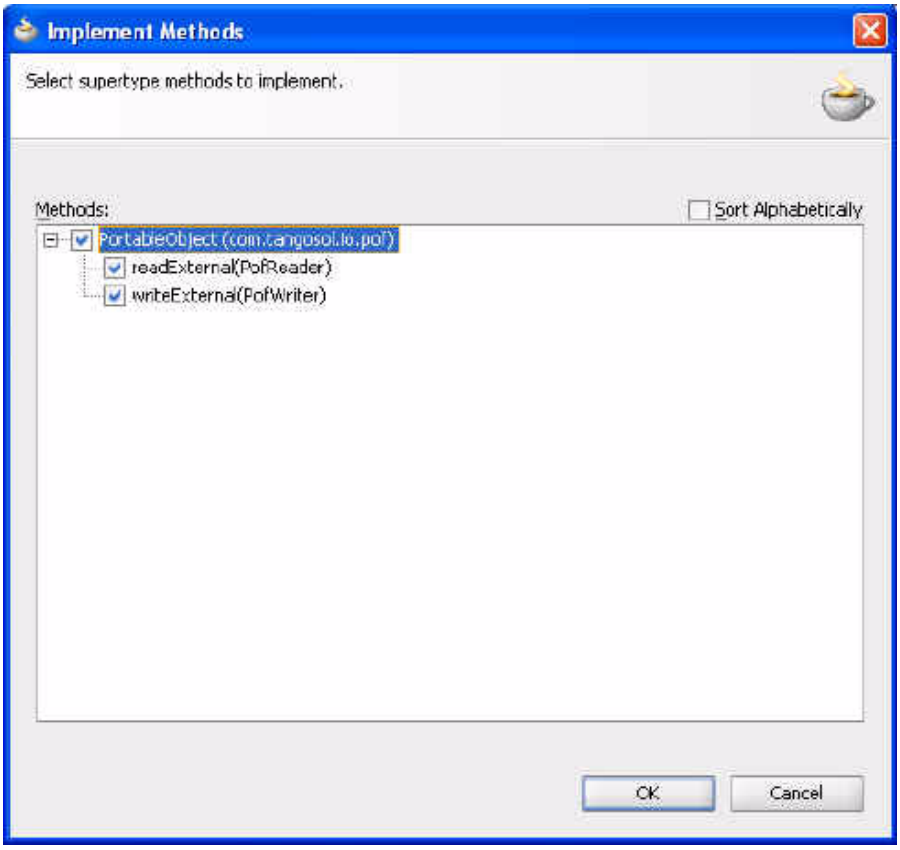
   a. Edit the `Person` class in JDeveloper and change `import java.io.Serializable` to `import com.tangosol.io.pof.PortableObject`. Change `implements Serializable` to `implements PortableObject`. Note that the `Person` class now appears with a red underline. If you place your cursor on this, JDeveloper tells you what the problem is.

   ```
   Methods not implemented: readExternal(PofReader) writeExternal(PofWriter)
   ```

   Using the `PortableObject` class requires that you implement its `readExternal(PofReader)` and `writeExternal(writeExternal)` methods.

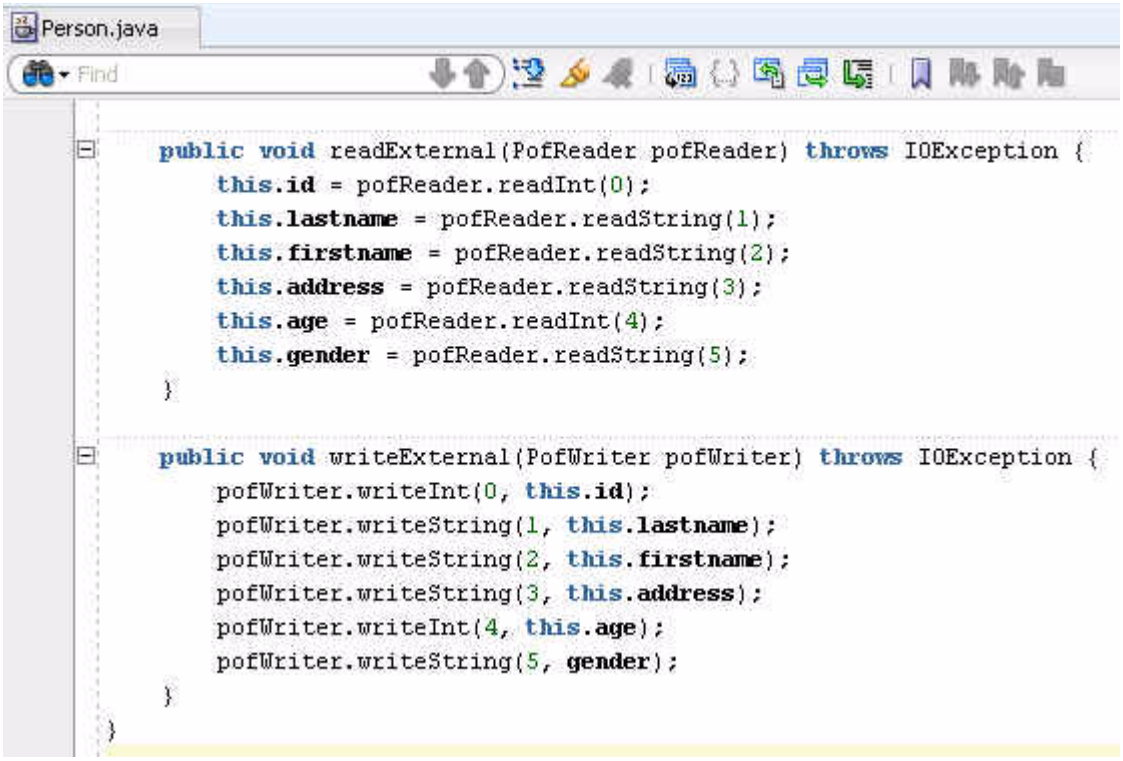
   b. Right-click `Person` and choose **Code Assist**. Select **Implement Methods**. The **Implement Methods** dialog box should appear. This generates the required **readExternal** and **writeExternal** methods. Accept the defaults and click **OK**.

Figure 3–13 Implement Methods Dialog Box



c. Make the following changes to the `Person` class. (You may need to import `java.io.IOException`.)

Figure 3–14 Implementing the PortableObject.readExternal and writeExternal Methods



d. Save the changes and rebuild your application.

4. Create a POF configuration file for `person`.

a. Create a copy of `coherence-pof-config.xml` and name it `lab4-pof-config.xml`.

Note: If the `coherence-pof-config.xml` file is not already present in your file system, you can find a copy of it in the `coherence.jar` file.

```
cp coherence-pof-config.xml lab4-pof-config.xml
```

b. Edit `lab4-pof-config.xml` and add a `<user-type>` element for `Person` to the `<user-type-list>` section. Use `1001` as the value for the `<type-id>` subelement and the full class name for `Person` in the `<class-name>` subelement. This is illustrated in Example 3–2.

Example 3–2 New <user-type> Element in POF Configuration File

```
 ...
 <user-type-list>
    ...
    <user-type>
      <type-id>1001</type-id>
      <class-name>com.oracle.coherence.handson.Person</class-name>
    </user-type>
</user-type-list>
...
```

c. Save the `lab4-pof-config.xml` file in the `\home\oracle\labs` directory.

d. Create a copy of `coherence-cache-config.xml` and name it `lab4-cache-config.xml`.

Note: If the `coherence-cache-config.xml` file is not already present in your file system, you can find a copy of it in the `coherence.jar` file.

```
cp coherence-cache-config.xml lab4-cache-config.xml
```

e. Edit `lab4-cache-config.xml` and un-comment the `serializer` section for the `example-distributed` caching scheme. This is illustrated in Example 3–3.

Example 3–3 Serializer Section in the Cache Configuration File

```
...
<caching-schemes>
    <!--
    Distributed caching scheme.
    -->
    <distributed-scheme>
      <scheme-name>example-distributed</scheme-name>
      <service-name>DistributedCache</service-name>

      <!-- To use POF serialization for this partitioned service,
           uncomment the following section -->

      <serializer>
        <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
      </serializer>

...
```

f. Add an `<init-param>` element to the serializer passing the name `lab4-pof-config.xml`.

Example 3–4 <init-param> Section in the Cache Configuration File

```
   ...
      <serializer>
        <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
        <init-params>
          <init-param>
            <param-value>lab4-pof-config.xml</param-value>
            <param-type>String</param-type>
          </init-param>
        </init-params>
      </serializer>
...
```

5. Stop all cache servers.

6. Copy `cache-server.cmd` to `pof-cache-server.cmd` and save the file in the `\oracle\product\coherence\bin` directory.

```
cd /oracle/product/coherence/bin
```

```
cp cache-server.cmd pof-cache-server.cmd
```

7. Edit `pof-cache-server.cmd` to point to the `lab4-cache-config.xml` file and `\home\oracle\labs` directory to the classpath. This is to ensure that the cache can find `lab4-pof-config.xml` file at runtime.
   a. Add `\home\oracle\labs` to the classpath.
   b. Add the location of `Person.class` to your classpath (`\home\oracle\labs\Lab4\classes`)
   c. Add `COHERENCE_OPTS="-Dtangosol.coherence.cacheconfig=\home\oracle\labs\lab4-cache-config.xml"`
   d. Add `COHERENCE_OPTS` to the `JAVAEXEC` command line. The resulting `pof-cache-server.cmd` file should look similar to this:

Example 3–5 Sample pof-cache-config.cmd File

```
@echo off
@
@rem This will start a cache server
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify the JVM heap size
set memory=512m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch

set java_opts="-Xms%memory% -Xmx%memory%"

set coherence_opts="-Dtangosol.coherence.log.level=9

-Dtangosol.coherence.cacheconfig=C:/home/oracle/labs/lab4-cache-config.xml"

"%java_exec%" -server -showversion "%java_opts%" "%coherence_opts%" -cp

"%coherence_
home%\lib\coherence.jar;C:\home\oracle\labs\Lab4\classes;C:\home\oracle\labs"

com.tangosol.net.DefaultCacheServer %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\pof-cache-server.cmd
goto exit
```

```
:exit
endlocal
@echo on
```

8. Start the cache server by executing the following command:

```
C:\oracle\product\coherence\bin>pof-cache-server.cmd
```

Example 3–6 lists sample output from running `pof-cache-server.cmd`.

Example 3–6 Starting the POF Cache Server

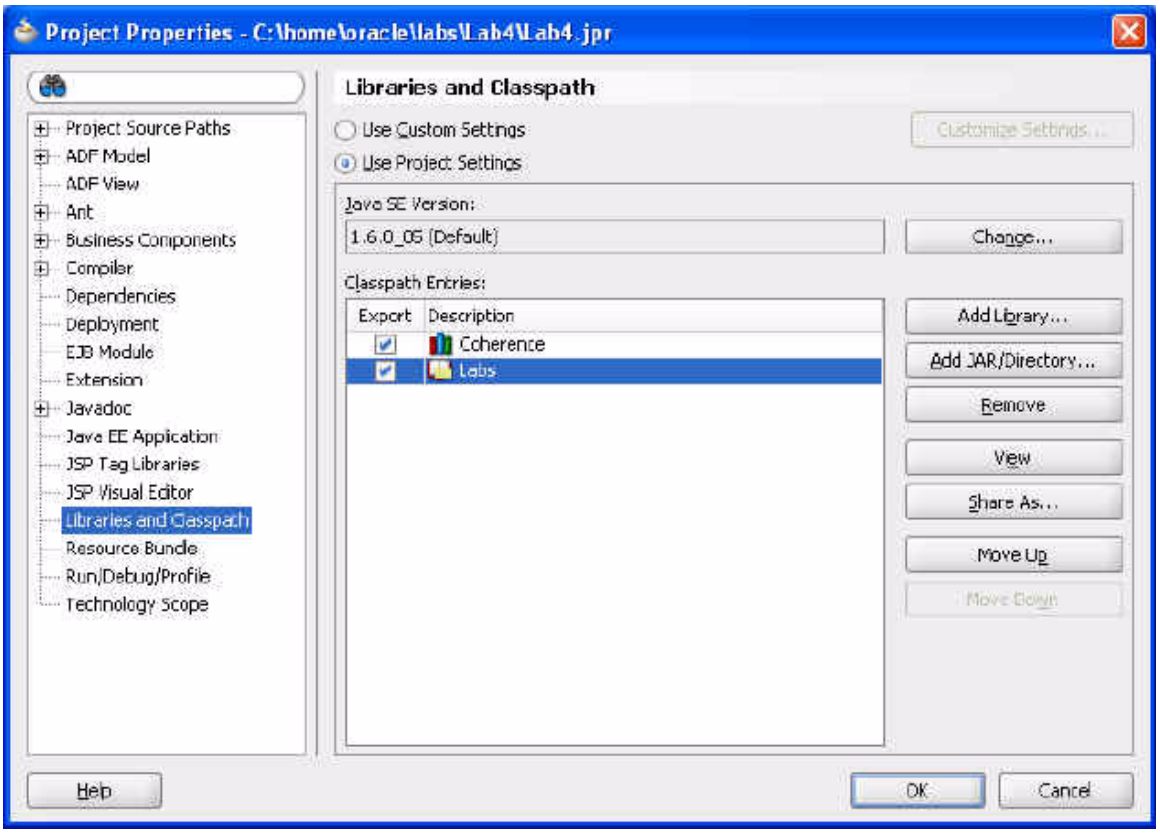
```
C:\oracle\product\coherence\bin>pof-cache-server.cmd
java version "1.6.0_05"
Java(TM) SE Runtime Environment (build 1.6.0_05-b13)
Java HotSpot(TM) Server VM (build 10.0-b19, mixed mode)

2008-12-16 18:20:05.665/0.547 Oracle Coherence 3.4.1/407 <Info> (thread=main, me
mber=n/a): Loaded operational configuration from resource "jar:file:/C:/oracle/p
roduct/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2008-12-16 18:20:05.681/0.563 Oracle Coherence 3.4.1/407 <Info> (thread=main, me
mber=n/a): Loaded operational overrides from resource "jar:file:/C:/oracle/produ
ct/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2008-12-16 18:20:05.681/0.563 Oracle Coherence 3.4.1/407 <D5> (thread=main, memb
er=n/a): Optional configuration override "/tangosol-coherence-override.xml" is n
ot specified
2008-12-16 18:20:05.697/0.579 Oracle Coherence 3.4.1/407 <D5> (thread=main, memb
er=n/a): Optional configuration override "/custom-mbeans.xml" is not specified
2008-12-16 18:20:05.697/0.579 Oracle Coherence 3.4.1/407 <D6> (thread=main, memb
er=n/a): Loaded edition data from "jar:file:/C:/oracle/product/coherence/lib/coh
erence.jar!/coherence-grid.xml"

Oracle Coherence Version 3.4.1/407
 Grid Edition: Development mode
Copyright (c) 2000-2008 Oracle. All rights reserved.

2008-12-16 18:20:06.400/1.282 Oracle Coherence GE 3.4.1/407 <Info> (thread=main,
 member=n/a): Loaded cache configuration from file "C:\home\oracle\labs\lab4-cac
he-config.xml"
2008-12-16 18:20:07.322/2.204 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluster
, member=n/a): Service Cluster joined the cluster with senior service member n/a

2008-12-16 18:20:07.540/2.422 Oracle Coherence GE 3.4.1/407 <Info> (thread=Clust
er, member=n/a): This Member(Id=3, Timestamp=2008-12-16 18:20:07.322, Address=13
0.35.99.248:8089, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-
pc,process:5564, Role=CoherenceServer, Edition=Grid Edition, Mode=Development, C
puCount=1, SocketCount=1) joined cluster "cluster:0x23CB" with senior Member(Id=
1, Timestamp=2008-12-16 13:01:46.572, Address=130.35.99.248:8088, MachineId=4991
2, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:1592, Role=CoherenceC
onsole, Edition=Grid Edition, Mode=Development, CpuCount=1, SocketCount=1)
2008-12-16 18:20:07.837/2.719 Oracle Coherence GE 3.4.1/407 <D5> (thread=Distrib
utedCache, member=3): Service DistributedCache joined the cluster with senior se
rvice member 3
2008-12-16 18:20:07.868/2.750 Oracle Coherence GE 3.4.1/407 <Info> (thread=Distr
ibutedCache, member=3): Loading POF configuration from resource "file:/C:/home/o

racle/labs/lab4-pof-config.xml"
2008-12-16 18:20:07.884/2.766 Oracle Coherence GE 3.4.1/407 <Info> (thread=Distr
ibutedCache, member=3): Loading POF configuration from resource "jar:file:/C:/or
acle/product/coherence/lib/coherence.jar!/coherence-pof-config.xml"
```

```
2008-12-16 18:20:08.337/3.219 Oracle Coherence GE 3.4.1/407 <D5> (thread=Replica
tedCache, member=3): Service ReplicatedCache joined the cluster with senior serv
ice member 3
2008-12-16 18:20:08.353/3.235 Oracle Coherence GE 3.4.1/407 <D5> (thread=Optimis
ticCache, member=3): Service OptimisticCache joined the cluster with senior serv
ice member 3
2008-12-16 18:20:08.384/3.266 Oracle Coherence GE 3.4.1/407 <D5> (thread=Invocat
ion:InvocationService, member=3): Service InvocationService joined the cluster w
ith senior service member 3
2008-12-16 18:20:08.384/3.266 Oracle Coherence GE 3.4.1/407 <Info> (thread=main,
 member=3): Started DefaultCacheServer...

SafeCluster: Name=cluster:0x23CB

Group{Address=224.3.4.1, Port=34407, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=3, Timestamp=2008-12-16 18:20:07.322, Address=130.35.99.2
48:8089, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proces
s:5564, Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2008-12-16 13:01:46.572, Address=130.35.99
.248:8088, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proc
ess:1592, Role=CoherenceConsole)
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2008-12-16 13:01:46.572, Address=130.35.99.248:8088,
MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:1592, R
ole=CoherenceConsole)
    Member(Id=3, Timestamp=2008-12-16 18:20:07.322, Address=130.35.99.248:8089,
MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:5564, R
ole=CoherenceServer)
    )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

Services
  (
  TcpRing{TcpSocketAccepter{State=STATE_OPEN, ServerSocket=130.35.99.248:8089},
Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Vers
ion=3.4, OldestMemberId=1}
  DistributedCache{Name=DistributedCache, State=(SERVICE_STARTED), LocalStorage=
enabled, PartitionCount=257, BackupCount=1, AssignedPartitions=257, BackupPartit
ions=0}
  ReplicatedCache{Name=ReplicatedCache, State=(SERVICE_STARTED), Id=2, Version=3
.0, OldestMemberId=3}
  Optimistic{Name=OptimisticCache, State=(SERVICE_STARTED), Id=3, Version=3.0, O
ldestMemberId=3}
  InvocationService{Name=InvocationService, State=(SERVICE_STARTED), Id=4, Versi
on=3.1, OldestMemberId=3}
  )

2008-12-16 18:20:09.087/3.969 Oracle Coherence GE 3.4.1/407 <D5> (thread=TcpRing
Listener, member=3): TcpRing: connecting to member 1 using TcpSocket{State=STATE
_OPEN, Socket=Socket[addr=/130.35.99.248,port=4044,localport=8089]}
```

9. Try to run the POF cache server in JDeveloper. To do this, add additional `CLASSPATH` entries to the existing project properties. Navigate to **Tools > Project**

Properties > Libraries and Classpath. Click **Add JAR/Directory**. In your project, add `\home\oracle\labs` to your `CLASSPATH`.

Figure 3–15 Adding Labs and Configuration Files to the Classpath

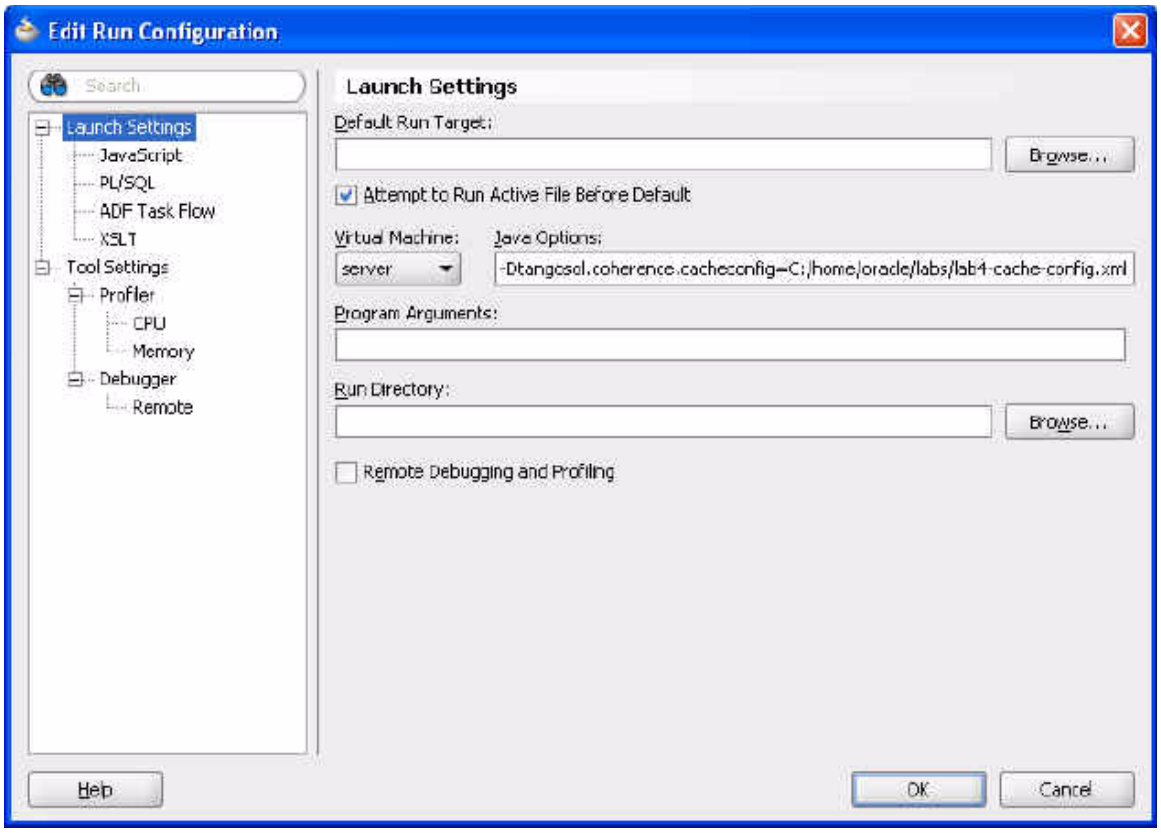


10. Click **Run/Debug/Profile** to edit the runtime properties. Append the following line to the existing **Java Options**.

    ```
    -Dtangosol.coherence.cacheconfig=C:\home\oracle\labs\lab4-cache-config.xml
    ```

    Click **OK** to save your changes to the runtime configuration and **OK** again to dismiss the **Project Properties** dialog box.

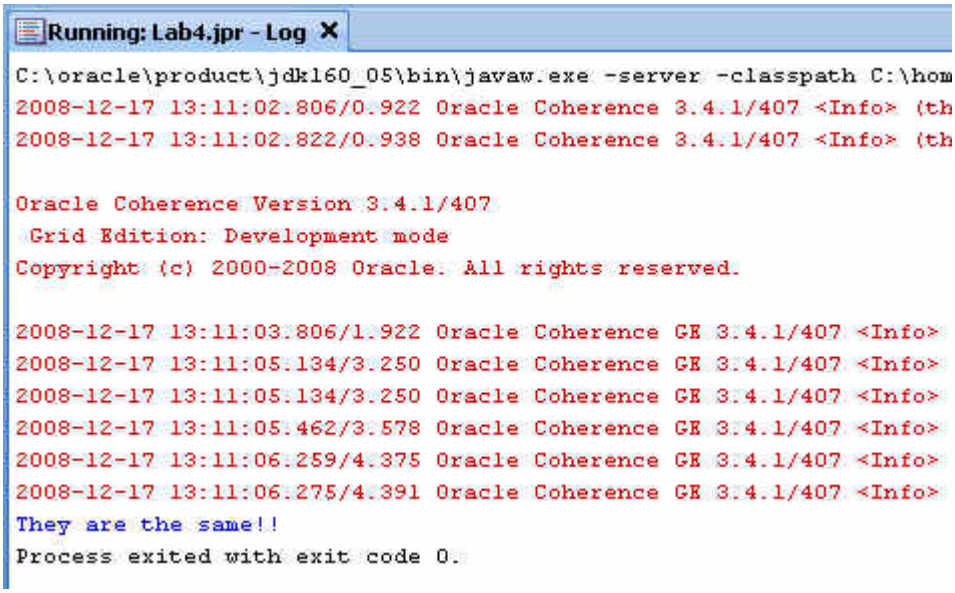
Figure 3–16 Setting the Runtime Profile



11. Rerun `PersonExample.java` from the JDeveloper IDE to ensure that it works.

    In this example, the `Person` object is converted to POF at run time instead of the standard Java serialized format. Using POF should provide significant performance improvements in terms of CPU time and the size of the binary generated.

Figure 3–17 PersonExample Output Run from JDeveloper



Caching a Complex Object using Java Serialization

In this practice, you create a simple domain object that can be placed into a Coherence cache. This practice assumes that you have a good understanding of Java language serialization rules.

Coherence requires that any non-primitive object placed into its cache be Java-Serializable. This is because the objects may need to be transported across process boundaries, that is, between JVMs, across networks, or processes on the same physical machine. The standard way to transport an object is to serialize, transmit, and then deserialize it. Coherence requires objects to be placed in a cache for them to be serializable.

1. Create a Java class called `EndOfDayStockSummary` that implements the `java.io.Serializable` interface.

   The purpose of this file is to capture the open, high, low, close, and adjusted close prices (in dollars) of a stock (called a symbol) for a specific date, with the trading volume. The variables in Example 3–7 can be the attribute definitions for the class:

Example 3–7 Attributes for a Sample Serializable Class

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

   a. Define a method called `getKey()` to return a `String` that is a combination of the symbol and date attributes.

   b. Declare `serialVersionUID`.

   c. Define a public default constructor `EndOfDayStockSummary()`.

   d. Define a public constructor that accepts values for all the attributes specified.

   Example 3–8 illustrates a sample solution.

Example 3–8 Sample Domain Object for the Coherence Cache

```
package com.oracle.coherence.handson;


import java.io.Serializable;
import java.util.Date;

public class EndOfDayStockSummary implements Serializable {

    private static final long serialVersionUID = -200684451000777011L;

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

```
public EndOfDayStockSummary() {
    //for Java Serialization.  not to be called directly by application code
}


public EndOfDayStockSummary(String symbol,
                            long date,
                            double openPrice,
                            double highPrice,
                            double lowPrice,
                            double closePrice,
                            double adjustedClosePrice,
                            long volume) {
    this.symbol = symbol;
    this.date = date;
    this.openPrice = openPrice;
    this.highPrice = highPrice;
    this.lowPrice = lowPrice;
    this.closePrice = closePrice;
    this.adjustedClosePrice = adjustedClosePrice;
    this.volume = volume;
}


public String getKey() {
    return symbol + date;
}


public String getSymbol() {
    return symbol;
}

public long getDate() {
    return date;
}


public double getOpenPrice() {
    return openPrice;
}

public double getHighPrice() {
    return highPrice;
}

public double getLowPrice() {
    return lowPrice;
}

public double getClosePrice() {
    return closePrice;
}


public double getAdjustedClosePrice() {
    return adjustedClosePrice;
}
```

```
    public long getVolume() {
        return volume;
    }

    public String toString() {
        return String.format("EndOfDayPrice{symbol=%s, date=%s, open=%f, high=%f,
low=%f, close=%f, adj-close=%f, volume=%d}",
                             symbol,
                             new Date(date),
                             openPrice,
                             highPrice,
                             lowPrice,
                             closePrice,
                             adjustedClosePrice,
                             volume);
    }
}
```

2. Create a console application called `CacheAnObject` which contains a `main` method.

   In the application, create an instance of the `EndOfDayStockSummary` class and use the `NamedCache` `put` method to place it into a Coherence cache called `dist-eodStockSummaries`. The application should then retrieve the summery and display it in the console.

   Example 3–9 illustrates a sample solution.

Example 3–9 Sample Console Application

```
import java.util.Date;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class CacheAnObject {

    public static void main(String[] args) {
        CacheFactory.ensureCluster();

        NamedCache namedCache = CacheFactory.getCache("dist-eodStockSummaries");

        EndOfDayStockSummary eodStockSummary = new EndOfDayStockSummary("orcl",
                                                    new Date().getTime(),
                                                    25.00,
                                                    27.00,
                                                    24.00,
                                                    26.00,
                                                    26.00,
                                                    1000);

        namedCache.put(eodStockSummary.getKey(), eodStockSummary);

        System.out.println(namedCache.get(eodStockSummary.getKey()));

        CacheFactory.shutdown();
    }

}
```

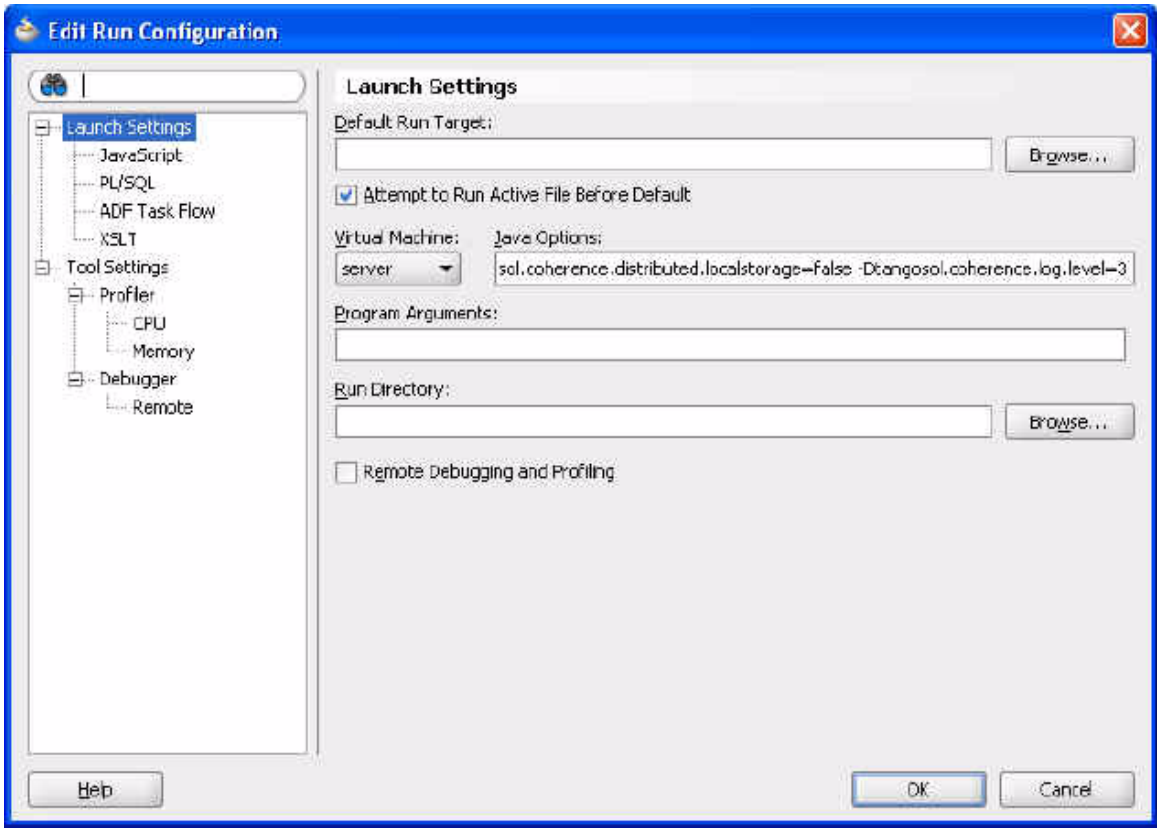
3. Edit the **Lab4 Project Properties** and modify the **Run/Debug/Profile** configuration. Remove the following line from the existing **Java Options** if it is present.

```
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\lab4-cache-config.xml
```

This field should still contain the following lines:

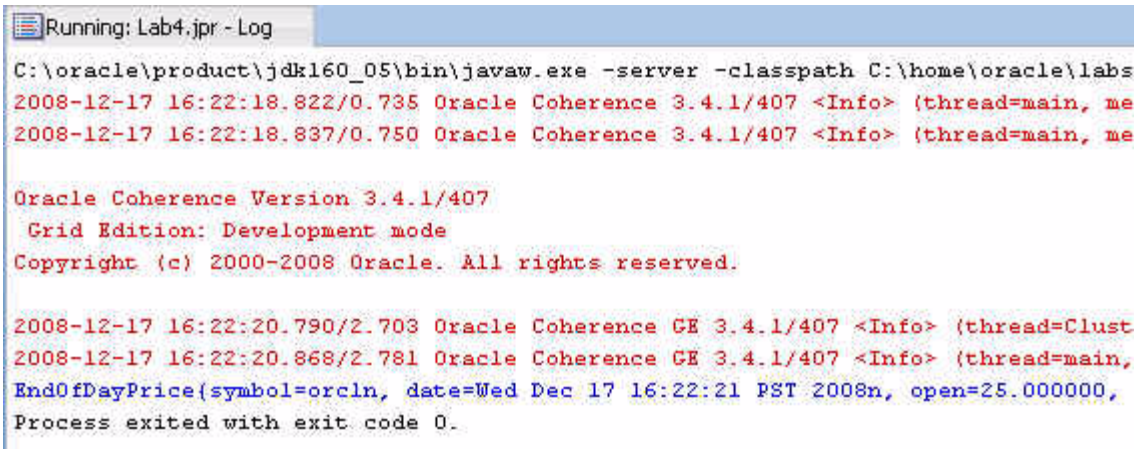
```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

Figure 3–18 Project Properties for the EndOfDayStockSummary Application



4. Edit `cache-server.cmd` and add the following:
   a. Add `\home\oracle\labs` to the classpath.
   b. Add the location of `Person.class` to your classpath (`\home\oracle\labs\Lab4\classes`)
5. Ensure that all other cache servers are shut down and execute `cache-server.cmd`. Ensure that the cache server starts successfully, then shut it down.
6. Execute `CacheAnObject` from the JDeveloper IDE and observe the results.

Figure 3–19 Output from the CacheAnObject Console Application



Caching a Complex Object using Coherence PortableObject

In this exercise, you improve the serialization performance of domain objects placed in the Coherence cache using the `com.tangosol.io.pof.PortableObject` interface.

This exercise assumes that you have successfully completed the previous section, "Caching a Complex Object using Java Serialization".

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 in their binary format, typically containing much more information than required by an application. It is also possible that the streams are encoded in such a manner that it is inefficient to send across a network, or to construct and destruct in a Java heap (memory). This consequently leads to increased garbage collection requirements.

To resolve some of these issues, including dramatically improving serialization performance, reducing the binary format size, and reducing the impact on garbage collection, Coherence provides its own proprietary serialization known as the `PortableObject` format. The `PortableObject` interface introduces two simple methods, `readExternal` and `writeExternal`, that permit a developer to explicitly read and write serialized object attributes from the provided `PofReader` and `PofWriter` streams respectively. By taking control over the serialization format, Coherence provides a way to dramatically improve the performance of the process. Using POF will dramatically reduce the size of the resulting binary. The size of the binary is often 5 to 10x smaller, and the conversion to-or-from the binary can be between 5 and 20 times faster, depending on the size of the object.

Note: Properties in POF are indexed and they should be read from the same index they were written to.

To implement `PortableObject` serialization format:

1. Modify the `EndOfDayStockSummary` class from the previous exercise to implement the `com.tangosol.io.pof.PortableObject` interface.
2. Using the methods defined in `com.tangosol.io.pof.PofReader` class and `com.tangosol.io.pof.PofWriter` the non-static methods defined in the

`DataInput` and `DataOutput` streams, implement the `readExternal` and `writeExternal` methods.

Example 3–10 Sample PortableObject Implementation

```
package com.oracle.coherence.handson;

import java.io.IOException;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;
import java.util.Date;

public class EndOfDayStockSummary implements PortableObject {

    private static final long serialVersionUID = -20068445100077701lL;

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


    public EndOfDayStockSummary() {
        //for Java Serialization.  not to be called directly by application code
    }


    public EndOfDayStockSummary(String symbol,
                                long date,
                                double openPrice,
                                double highPrice,
                                double lowPrice,
                                double closePrice,
                                double adjustedClosePrice,
                                long volume) {
        this.symbol = symbol;
        this.date = date;
        this.openPrice = openPrice;
        this.highPrice = highPrice;
        this.lowPrice = lowPrice;
        this.closePrice = closePrice;
        this.adjustedClosePrice = adjustedClosePrice;
        this.volume = volume;
    }


    public String getKey() {
        return symbol + date;
    }


    public String getSymbol() {
        return symbol;
    }
```

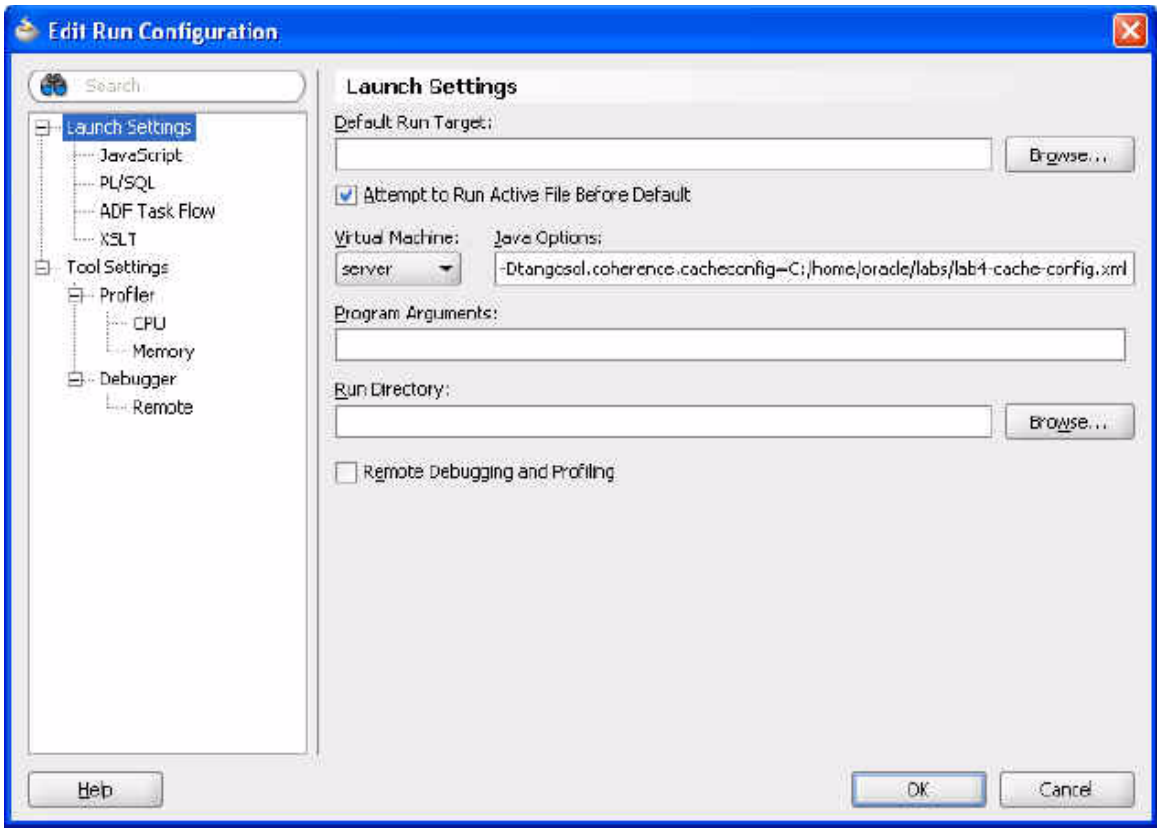
```
    public long getDate() {
        return date;
    }

    public double getOpenPrice() {
        return openPrice;
    }

    public double getHighPrice() {
        return highPrice;
    }

    public double getLowPrice() {
        return lowPrice;
    }

    public double getClosePrice() {
        return closePrice;
    }


    public double getAdjustedClosePrice() {
        return adjustedClosePrice;
    }

    public long getVolume() {
        return volume;
    }

    public String toString() {
        return String.format("EndOfDayPrice{symbol=%s, date=%s, open=%f, high=%f,
low=%f, close=%f, adj-close=%f, volume=%d}",
                             symbol,
                             new Date(date),
                             openPrice,
                             highPrice,
                             lowPrice,
                             closePrice,
                             adjustedClosePrice,
                             volume);
    }

    public void readExternal(PofReader pofReader) throws IOException {
        symbol=pofReader.readString(0);
        date=pofReader.readLong(1);
        openPrice=pofReader.readDouble(2);
        lowPrice=pofReader.readDouble(3);
        highPrice=pofReader.readDouble(4);
        closePrice=pofReader.readDouble(5);
        adjustedClosePrice=pofReader.readDouble(6);
        volume=pofReader.readLong(7);

    }

    public void writeExternal(PofWriter pofWriter) throws IOException {
        pofWriter.writeString(0, symbol);
        pofWriter.writeLong(1, date);
        pofWriter.writeDouble(2, openPrice);
        pofWriter.writeDouble(3, highPrice);
```

```
        pofWriter.writeDouble(4, lowPrice);
        pofWriter.writeDouble(5, closePrice);
        pofWriter.writeDouble(6, adjustedClosePrice);
        pofWriter.writeLong(7, volume);

    }
}
```

3. Modify `lab4-pof-config.xml` in `\home\oracle\labs` to have an entry for `EndOfDayStockSummary`. Hint: You can change `<type-id>` to `1002` and `<class-name>` to `com.oracle.coherence.handson.EndOfDayStockSummary`.
4. Edit the Lab4 **Project Properties** and modify the **Run/Debug/Profile** configuration. Append the following line to the existing **Java Options**:

```
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\lab4-cache-config.xml
```

Figure 3–20 Setting Java Options for the PortableObject Implementation



5. Save the changes and rebuild your application.
6. Ensure that all other cache servers are shut down and execute `pof-cache-server.cmd` on the command line

```
C:\oracle\product\coherence\bin>pof-cache-server.cmd
```

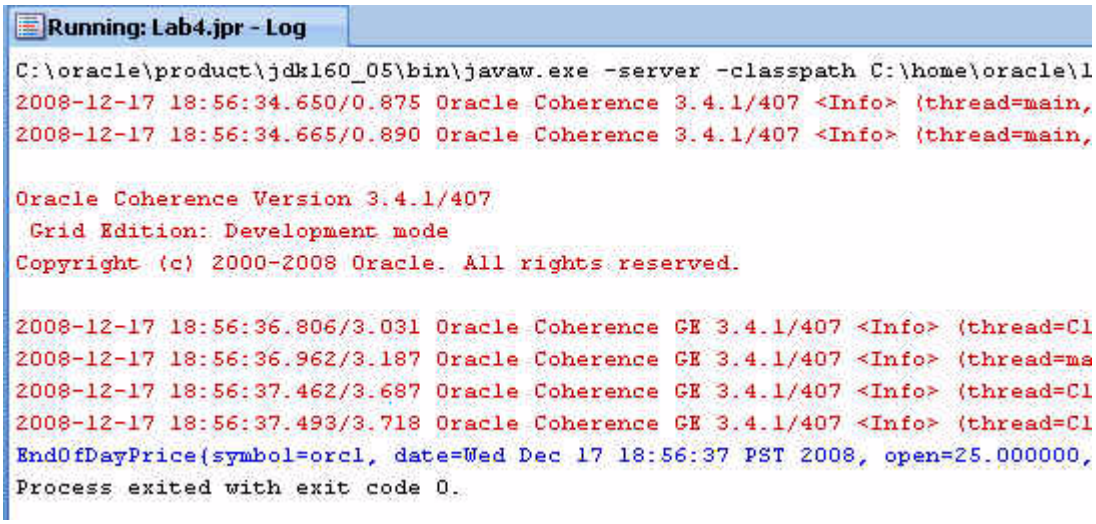
In the output from the server startup, you should see lines similar to those in Example 3–11 that indicate that a POF configuration is being employed.

Example 3–11 Output from Starting the Server with a POF Configuration

```
...
2008-12-17 18:47:12.900/3.407 Oracle Coherence GE 3.4.1/407 <Info> (thread=Distr
ibutedCache, member=2): Loading POF configuration from resource "file:/C:/home/o
racle/labs/lab4-pof-config.xml"
2008-12-17 18:47:12.931/3.438 Oracle Coherence GE 3.4.1/407 <Info> (thread=Distr
ibutedCache, member=2): Loading POF configuration from resource "jar:file:/C:/or
acle/product/coherence/lib/coherence.jar!/coherence-pof-config.xml"
2008-12-17 18:47:13.431/3.938 Oracle Coherence GE 3.4.1/407 <D5> (thread=Replica
tedCache, member=2): Service ReplicatedCache joined the cluster with senior serv
ice member 2
2008-12-17 18:47:13.462/3.969 Oracle Coherence GE 3.4.1/407 <D5> (thread=Optimis
ticCache, member=2): Service OptimisticCache joined the cluster with senior serv
ice member 2
2008-12-17 18:47:13.478/3.985 Oracle Coherence GE 3.4.1/407 <D5> (thread=Invocat
ion:InvocationService, member=2): Service InvocationService joined the cluster w
ith senior service member 2
2008-12-17 18:47:13.478/3.985 Oracle Coherence GE 3.4.1/407 <Info> (thread=main,
 member=2): Started DefaultCacheServer...
...
```

7. Verify that your `CacheAnObject.java` class from the previous exercise uses the `EndOfDayStockSummary` that implemented `PortableObject`. Run `CacheAnObject` from the JDeveloper IDE.

Figure 3–21 Running EndOfDayStockSummary with a PortableObject Implementation



It is important to keep the order of the attributes and their indexes consistent between the `readExternal` and `writeExternal` methods. As an experiment, try changing the index of the object attributes written in the `writeExternal` method to be different from the index of the reading object attributes in the `readExternal` method.

For example, change the order of the `openPrice` and `lowPrice` attributes and indexes as they currently appear in the `writeExternal` method:

```
...
pofWriter.writeDouble(2, openPrice);
pofWriter.writeDouble(3, highPrice);
pofWriter.writeDouble(4, lowPrice);
...
```

to this order:

```
...
pofWriter.writeDouble(2, lowPrice);
pofWriter.writeDouble(3, highPrice);
pofWriter.writeDouble(4, openPrice);
...
```

Run `CacheAnObject.java` again. Notice that the assigned values in the output are transposed.

4

Loading Data Into a Cache

In this chapter, you learn how to populate a Coherence cache with domain objects that are read from text files. You also learn the most efficient method of loading data into a Cache (in sequence).

This chapter contains the following sections:

Introduction

This chapter assumes that you have completed "Caching a Complex Object using Java Serialization" on page 3-20 and "Caching a Complex Object using Coherence PortableObject" on page 3-24.

This chapter also assumes that you are familiar with using `java.io.BufferedReader` to read text files, `java.lang.String.split` method to parse text files, and `java.text.SimpleDateFormat` to parse dates.

The following examples provide brief illustrations of these classes.

- The code in Example 4–1 illustrates how to use `BufferedReader` to open and close a text file called `myFile`:

Example 4–1 Sample Code to Open and Close a Text File

```
BufferedReader in = new BufferedReader(new FileReader(myFile));
...
   // your reading code here
...
in.close();
...
```

- The sample code in Example 4–2 illustrates how to use the `String.split` method to split a string into an array of substrings based on a delimiter. In this case, the delimiter is a comma:

Example 4–2 Sample Code to Split a String

```
...
String[] parts = stringToSplit.split(",");
...
```

- The sample code in Example 4–3 illustrates how to parse a date expressed as a `yyyy-MM-dd` formatted string into a `java.util.Date` instance:

Example 4–3 Sample Code to Parse a Date

```
...
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
Date date = sdf.parse(dateAsAString);
...
```

Populating a Cache with Domain Objects

This exercise demonstrates two ways to create a console application that will populate a Coherence cache with domain objects. The application can use either the Java `Serialization` or the Coherence `com.tangosol.io.pof.PortableObject` implementation. The exercise also includes a stopwatch program so you can observe the time savings that the `PortableObject` implementation provides.

To populate a cache with domain objects:

1. Create a new project called `Lab5`.

   See "Creating and Caching Complex Objects" on page 3-1 for information on creating a new project.

2. Download the `samples.zip` file from the Coherence Library main page:

   The zip file contains the `endofdaystocksummaries` folder which, in turn, contains several text files with stock price data.

   a. Click the **Tutorial Sample Data** link on the Oracle Coherence Library main page to download the `samples.zip` file.

   b. Unzip `samples.zip` and place the `endofdaystocksummaries` folder in the `\home\oracle\labs\Lab5` directory.

3. Write a console application (Java class) called `CacheLoading` to load the entire end-of-day-stock summaries contained in the `endofdaystocksummaries` folder into a single Coherence cache called "`eodss`."

   Example 4–4 illustrates a possible solution.

Example 4–4 Sample Cache Loading Program

```
package com.oracle.coherence.handson;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class CacheLoading {

    public static void loadPricesFor(String symbol, NamedCache namedCache) throws
IOException, NumberFormatException, ParseException {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
```

```
        String fileName = "./endofdaystocksummaries/" + symbol + ".CSV";

        BufferedReader in = new BufferedReader(new FileReader(fileName));
        in.readLine();

        String line = null;
        while ((line = in.readLine()) != null) {
            String[] parts = line.split(",");

            EndOfDayStockSummary eodss =
                new EndOfDayStockSummary(symbol,
                                         dateFormat.parse(parts[0]).getTime(),
                                         Double.parseDouble(parts[1]),
                                         Double.parseDouble(parts[2]),
                                         Double.parseDouble(parts[3]),
                                         Double.parseDouble(parts[4]),
                                         Double.parseDouble(parts[6]),
                                         Long.parseLong(parts[5]));

            namedCache.put(eodss.getKey(), eodss);
        }

        in.close();
    }


    /**
     * @param args
     * @throws IOException
     * @throws ParseException
     * @throws NumberFormatException
     */
    public static void main(String[] args) throws IOException,
NumberFormatException, ParseException {

        NamedCache namedCache = CacheFactory.getCache("eodss");

        String[] symbols = {"AAPL", "BT", "DELL", "GOOG", "HPQ", "JAVA", "MSFT",
"ORCL", "YHOO"};
StopWatch st=new StopWatch();
st.start();

        for (String symbol : symbols) {
            System.out.printf("Loading Symbol %s\n", symbol);
            loadPricesFor(symbol, namedCache);
        }
st.stop();
System.out.println("elapsed time is:"+st.getElapsedTime());
    }
}
```

4. Create a Java class called `EndOfDayStockSummary` and copy the code from the `EndOfDayStockSummary` class from the previous exercise (see "Caching a Complex Object using Java Serialization" on page 3-20).

5. Create a `StopWatch` Java class to determine how long it takes to `put` the summaries into the cache and how many may be `put` per second.

   Example 4–5 illustrates a possible solution.

Example 4–5 Sample Stopwatch Program

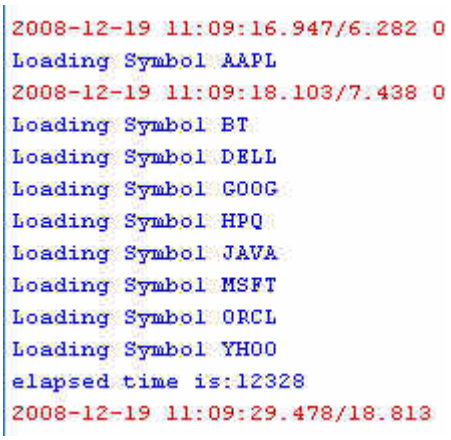
```
package com.oracle.coherence.handson;

public class StopWatch {

    private long startTime;       //time in ms since the EPOC
    private long finishTime;    //time in ms since the EPOC
    private boolean isRunning;

    public StopWatch() {
        this.startTime = -1;
        this.finishTime = -1;
        isRunning = false;
    }


    public synchronized void start() {
        if (!isRunning) {
            this.startTime = System.currentTimeMillis();
            isRunning = true;
        }
    }


    public synchronized void stop() {
        if (isRunning) {
            this.finishTime = System.currentTimeMillis();
            isRunning = false;
        }
    }


    public boolean isRunning() {
        return isRunning;
    }


    public long getElapsedTime() {
        if (isRunning) {
            return System.currentTimeMillis() - startTime;
        } else if (startTime != -1) {
            return finishTime - startTime;
        } else {
           throw new IllegalStateException("StopWatch hasn't been started as yet.
Not possible to determine the elapsed time.");
        }
    }

    public String toString(int count) {
        return String.format("StopWatch{startTime=%d, finishTime=%d,
elapsedTime=%d ms (%f sec), rate=%f per sec}",
                             startTime,
                             finishTime,
                             getElapsedTime(),
                             getElapsedTime() / 1000.0,
                             (double)count * 1000.0 / getElapsedTime());
    }

    public String toString() {
        return String.format("StopWatch{startTime=%d, finishTime=%d,
```

```
elapsedTime=%d ms (%f sec)}",
                                startTime,
                                finishTime,
                                getElapsedTime(),
                                getElapsedTime() / 1000.0);
    }
}
```

6. Run your application using the following cluster configurations:

   a. Run `CacheLoader` with no cache servers running.

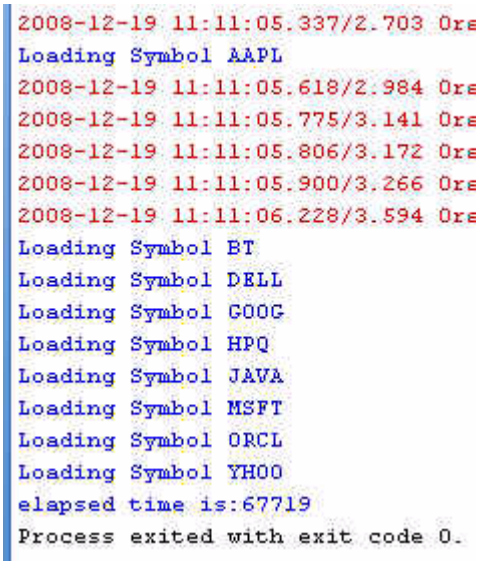
Figure 4–1 Sample Program Run without a Cache Server



```
2008-12-19 11:09:16.947/6.282 O
Loading Symbol AAPL
2008-12-19 11:09:18.103/7.438 O
Loading Symbol BT
Loading Symbol DELL
Loading Symbol GOOG
Loading Symbol HPQ
Loading Symbol JAVA
Loading Symbol MSFT
Loading Symbol ORCL
Loading Symbol YHOO
elapsed time is:12328
2008-12-19 11:09:29.478/18.813
```

   b. Run `CacheLoader` with a single cache server running (that is, run `cache-server.cmd` in a shell).

Figure 4–2 Sample Program Run with One Cache Server



```
2008-12-19 11:11:05.337/2.703 Ora
Loading Symbol AAPL
2008-12-19 11:11:05.618/2.984 Ora
2008-12-19 11:11:05.775/3.141 Ora
2008-12-19 11:11:05.806/3.172 Ora
2008-12-19 11:11:05.900/3.266 Ora
2008-12-19 11:11:06.228/3.594 Ora
Loading Symbol BT
Loading Symbol DELL
Loading Symbol GOOG
Loading Symbol HPQ
Loading Symbol JAVA
Loading Symbol MSFT
Loading Symbol ORCL
Loading Symbol YHOO
elapsed time is:67719
Process exited with exit code 0.
```

   c. Run `CacheLoader` with two cache servers running (that is, run `cache-server.cmd` in a second shell).

Figure 4–3 Sample Program Run with Two Cache Servers

```
2008-12-19 11:13:37.306/25.313 Or
2008-12-19 11:13:37.478/25.485 Or
2008-12-19 11:13:37.665/25.672 Or
2008-12-19 11:13:37.822/25.829 Or
2008-12-19 11:13:37.993/26.000 Or
2008-12-19 11:13:38.150/26.157 Or
Loading Symbol BT
Loading Symbol DELL
Loading Symbol GOOG
Loading Symbol HPQ
Loading Symbol JAVA
Loading Symbol MSFT
Loading Symbol ORCL
Loading Symbol YHOO
elapsed time is:106797
Process exited with exit code 0.
```

d. Run the `CacheLoading` application configured with the following **Java Options** and with both cache servers running.

```
-Dtangosol.coherence.distributed.localstorage=false
```

Figure 4–4 Sample Program Run with Two Cache Servers and Local Storage Set to False

```
2008-12-19 14:13:15.286/4.375 Oracle
2008-12-19 14:13:15.301/4.390 Oracle
Loading Symbol AAPL
Loading Symbol BT
Loading Symbol DELL
2008-12-19 14:14:05.582/54.671 Oracle
2008-12-19 14:14:05.582/54.671 Oracle
2008-12-19 14:14:05.582/54.671 Oracle
2008-12-19 14:14:06.582/55.671 Oracle
Loading Symbol GOOG
Loading Symbol HPQ
Loading Symbol JAVA
Loading Symbol MSFT
Loading Symbol ORCL
Loading Symbol YHOO
elapsed time is:90625
Process exited with exit code 0.
```

7. Modify your application to use the `PortableObject` implementation of the `EndOfDayStockSummary` class (You can copy the code from "Caching a Complex Object using Coherence PortableObject" on page 3-24).

a. Save the changes and rebuild your application.

b. Stop all running cache servers.

c. Edit the `lab4-pof-congig.xml` file to comment-out the user type `Person`.

```
...
<!--
```

```
    <user-type>
      <type-id>1001</type-id>
      <class-name>com.oracle.coherence.handson.Person</class-name>
    </user-type>
-->
...
```

d. Edit the `CLASSPATH` environment variable in `pof-cache-server.cmd` to add the `\home\oracle\labs\Lab5\classes` directory. Start `pof-cache-server.cmd`.

```
C:\oracle\product\coherence\bin>pof-cache-server.cmd
```

e. Edit the **Run/Debug/Profile** configuration. Change the Java process to be storage-disabled and also append the following line to the **Java Options**.

```
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\lab4-cache-config.xml
```

f. Add additional `CLASSPATH` entries to the existing project properties.

Navigate to **Tools > Project Properties > Libraries** and **Classpath > Add JAR/Directory**. In your project, add `\home\oracle\labs` to your `CLASSPATH`. Execute the application. What effect does this have on throughput?

Figure 4–5 Sample Program Run with Pof Serialization

```
2008-12-19 14:59:40.536/3.219 Oracle
2008-12-19 14:59:40.614/3.297 Oracle
Loading Symbol AAPL
2008-12-19 14:59:41.098/3.781 Oracle
Loading Symbol BT
Loading Symbol DELL
Loading Symbol GOOG
Loading Symbol HPQ
Loading Symbol JAVA
Loading Symbol MSFT
Loading Symbol ORCL
Loading Symbol YHOO
elapsed time is:44250
Process exited with exit code 0.
```

8. Use Table 4–1 to enter your results. Observe that the throughput is more efficient when using `PortableObject` as compared to `Serialization`.

Table 4–1 Throughput Calculations: Using PortableObject versus Serialization

CacheLoading Application Configuration	No Cache Servers: Throughput (per sec)	One Cache Servers: Throughput (per sec)	Two Cache Servers: Throughput (per sec)
Using the Java serialization version of EndOfDayStockSummary			

Table 4–1 (Cont.) Throughput Calculations: Using PortableObject versus Serialization

CacheLoading Application Configuration	No Cache Servers: Throughput (per sec)	One Cache Servers: Throughput (per sec)	Two Cache Servers: Throughput (per sec)
Using the Java serialization version of EndOfDayStockSummary with local storage disabled	N/A		
Using the PortableObject version of EndOfDayStockSummary			
Using the PortableObject version of EndOfDayStockSummary with local storage disabled	N/A		

Querying and Aggregating Data in the Cache

This exercise introduces the concept of querying and aggregating data in a cache. In this exercise:

- Create a Java class to populate the cache with 10,000 `Person` objects
- Query the cache for specific data
- Aggregate information within the cache and observe the changes in query times when you add cache members (On dual core machines only)

After putting complex objects in the named caches, you look at querying and aggregating information within the grid. The `com.tangosol.util.QueryMap` interface provides methods for managing the values or keys within a cache. You can use filters to restrict your results. You can also define indexes to optimize your queries.

Because Coherence serializes information when storing, you will have the overhead of deserializing when querying. When indexes are added, the values identified in the index are not serialized and therefore, offer quicker access time. Some of the more useful methods in the `QueryMap` interface are:

- `Set entrySet(Filterfilter)`—Returns a set of entries that are contained in the map that satisfy the filter
- `addIndex(ValueExtractorextractor,booleanfOrdered, Comparator comparator)`—Adds an index
- `Set keySet(Filter filter)`—Similar to `entrySet`, but returns keys, not values

It is important to note that filtering occurs at Cache Entry Owner level. In a partitioned topology, filtering can be done in parallel because it is the primary partitions that do the filtering. The `QueryMap` interface uses the `Filter` classes. You can find more information on these classes in the API for the `com.tangosol.util.filter` package.

All Coherence `NamedCaches` implement the `com.tangosol.util.QueryMap` interface. This allows `NamedCaches` to support the searching for keys or entries in a cache that satisfy some condition. The condition can being represented as an object that implements the `com.tangosol.util.Filter` interface.

The `com.tangosol.util.filter` package contains several predefined classes that provide implementations of standard query expressions. Examples of these classes include `GreaterFilter`, `GreaterEquals`, `LikeFilter`, `NotEqualsFilter`, `InFilter`, and so on. You can use these filters to construct and compose object-based equivalents of almost all SQL `WHERE` clause expressions.

> **Note:** Coherence does not provide a `SQLFilter` because it is unlikely that the objects placed in a cache are modeled in a relational manner, that is, using rows and columns (as they are typically represented in a database). Additionally, it is common that objects placed in a cache are not easily modeled using relational models, for example, large blobs.

The `Filter` classes use standard Java method reflection to represent test conditions. For example, the following filter represents the condition where the value returned from the `getSymbol` method on an object in a cache (for example, `endofdaystocksummary`) is for Oracle (`ORCL`):

```
new EqualsFilter("getSymbol", "ORCL");
```

If the object tested with this filter fails to have a `getSymbol` method, then the test will fail.

A couple of examples will make things clearer:

- Return a set of people where their last name begins with `Sm`:

  ```
  Set macPeople = people.entrySet( new LikeFilter("getLastName", "Sm"));
  ```

- Return a set containing all open trades:

  ```
  Set openTrades = trades.entrySet( new EqualsFilter("isOpen", BOOLEAN.TRUE));
  ```

In addition to the `entrySet` and `keySet` methods defined by the `QueryMap` interface, Coherence supports the definition of indexes, using the `addIndex` method, to improve query performance. Unlike relational database systems, where indexes are defined according to well-known and strictly enforced collections of named columns (that is, a schema), Coherence does not have a schema. Though lacking a formal schema for data allows for significant flexibility and polymorphism, within applications, it means that an approach different from that of traditional database systems is required to define indexes and therefore, increase query performance.

To define the values that are to be indexed for each object placed in a cache, Coherence introduces the concept of a `ValueExtractor`. A `com.tangosol.util.ValueExtractor` is a simple interface that defines an "`extract`" method. If given an object parameter, a `ValueExtractor` returns some value based on the parameter.

A simple example of a `ValueExtractor` implementation is the `com.tangosol.util.extractor.ReflectionExtractor`, which uses reflection to return the result of a method call on an object. For example:

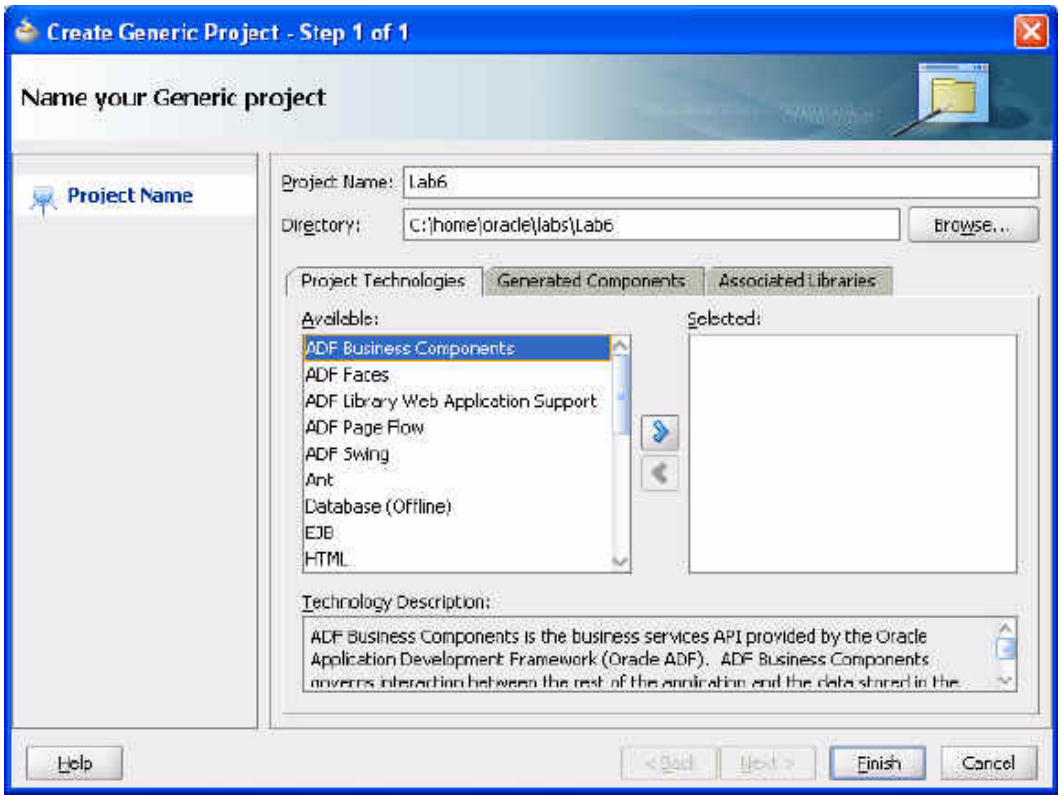
```
new ReflectionExtractor("getSymbol")
```

`ValueExtractors` may be used throughout the Coherence API. Typically, however, they are used to define indexes.

To aggregate and query data in the cache:

1. Create a Java class to populate the cache with 10,000 random `People` objects.
   a. Create a JDeveloper project and call it `Lab6`.

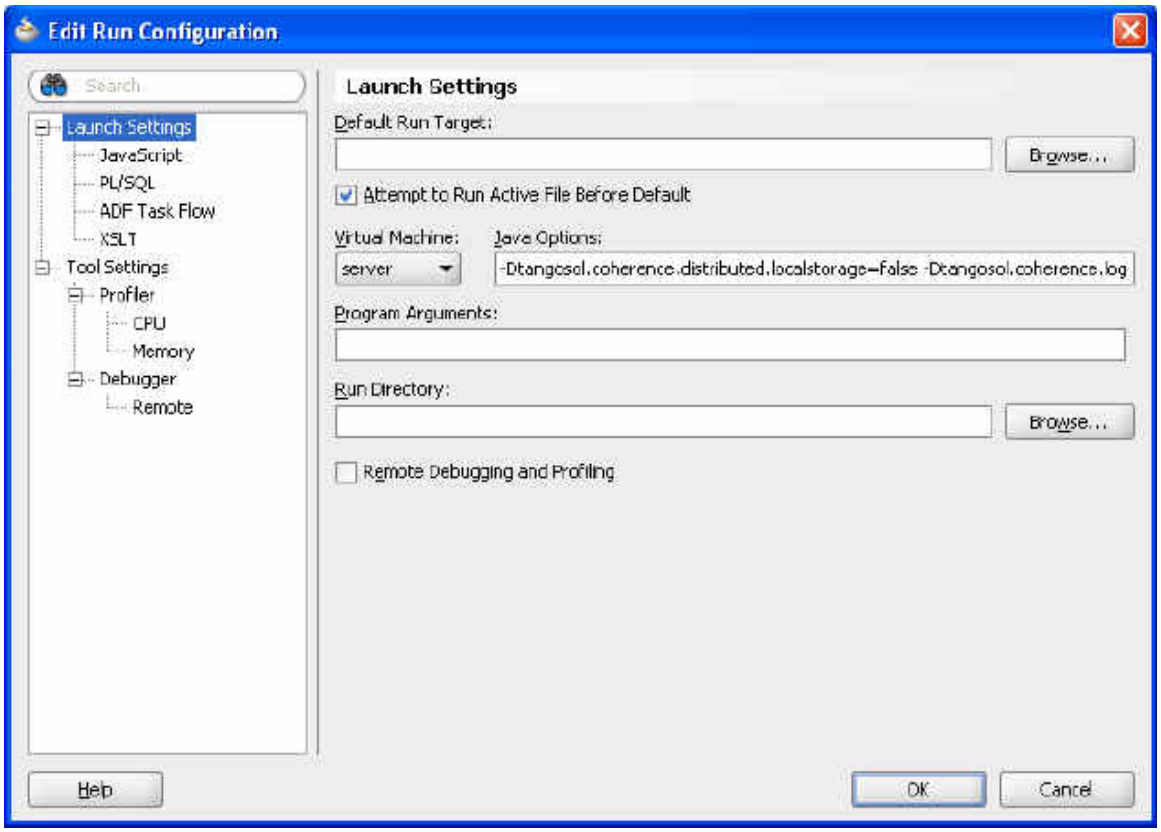
Figure 4–6 Creating a New Project



b. Change the Java process to be storage-disabled when it is run. Right click the new project and select **Project Properties**. In the project properties dialog box, select **Run/Debug/Profile**. In the **Java Options** field, enter these parameters:

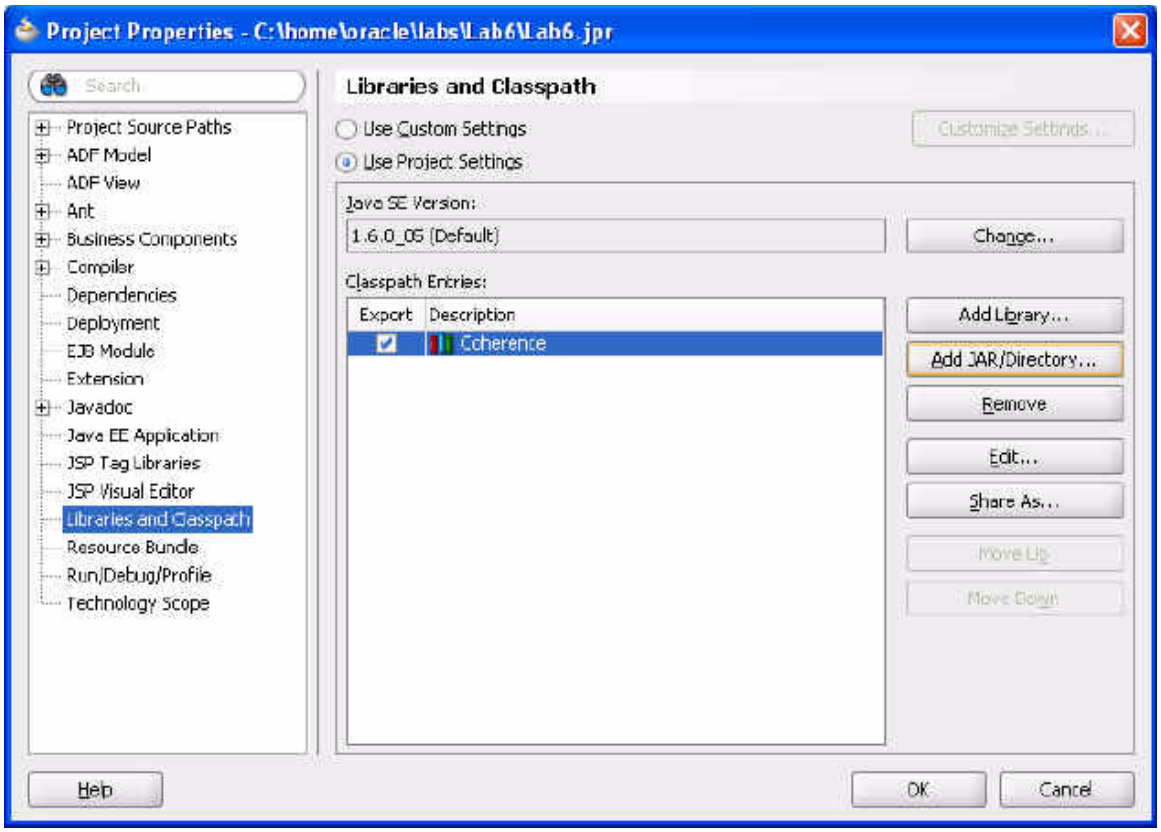
```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

Figure 4–7 Setting Runtime Parameters



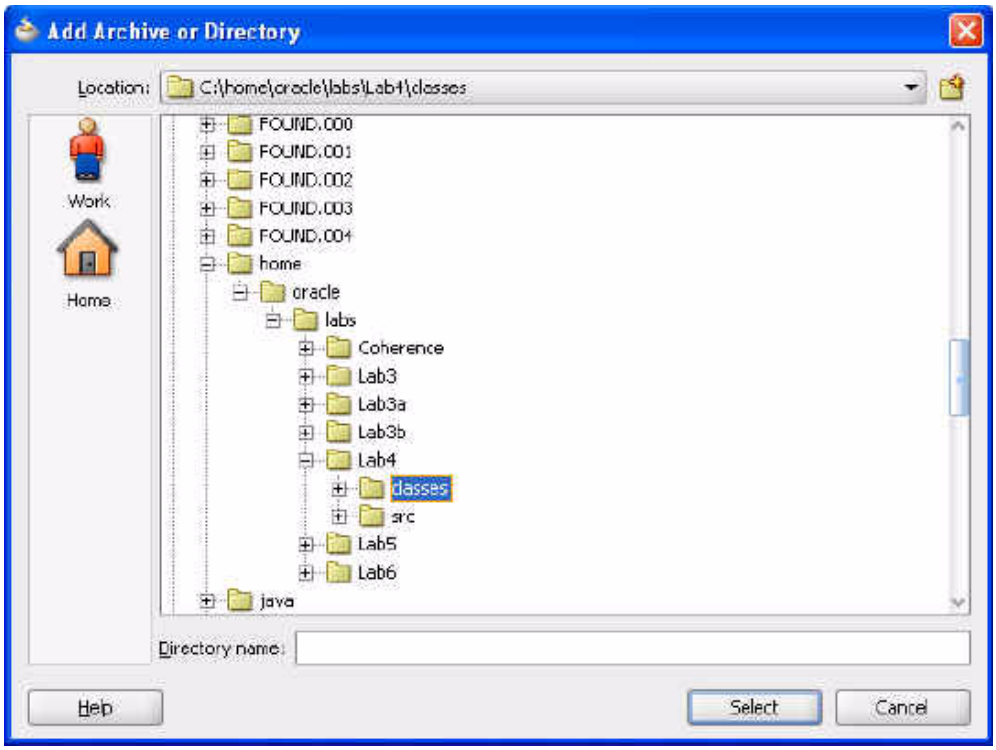
c. Use the `Person` class that you created in the earlier lab (`Lab4`). If the `Person` class in `Lab4` implements `PortableObject`, then modify the `Person.java` in `Lab4` to use `Serialization`. Save and rebuild `Lab4`. Right-click `Lab6` and select **Project Properties**. Select **Libraries and Classpath**. If you do not have `\home\oracle\labs\Lab4\classes` in your `CLASSPATH`, click **Add Jar/Directory**.

Figure 4–8 Adding Jars and Directories to Classpath



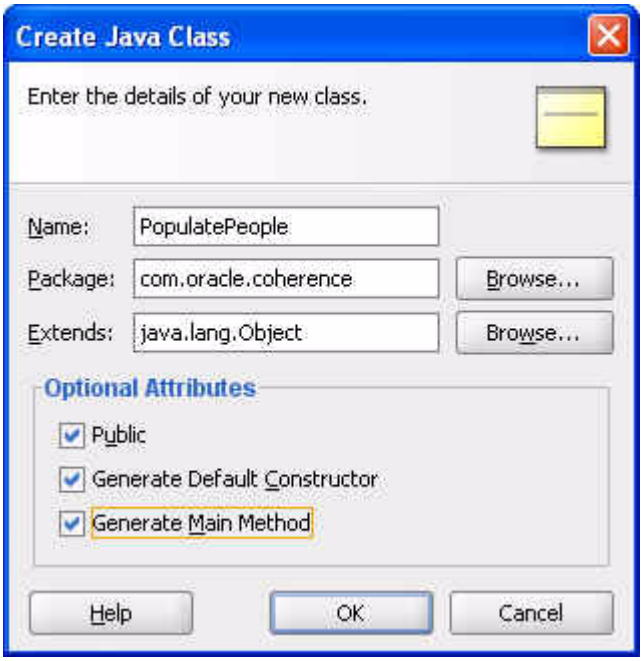
d. Select the `\home\oracle\labs\Lab4\classes` directory.

Figure 4–9 Adding Classes to the Classpath



e. Create a new Java class called `PopulatePeople`. Ensure that it has a `main` method.

Figure 4–10 Creating a Java Class



f. Create the code in the new class to connect to the cache and create 10,000 random `Person` objects. Hint: Use `java.util.Random` to generate some random ages.

```
Random generator = new Random(); int age = generator.nextInt(100);
```

Example 4–6 illustrates a possible solution.

Example 4–6 Sample PopulatePeople Class

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import com.tangosol.util.extractor.ReflectionExtractor;

import java.io.IOException;

import java.util.Random;

public class PopulatePeople {
    public PopulatePeople() {
    }

    public static void main(String[] args) throws IOException {

        NamedCache person = CacheFactory.getCache("person");

        // add indexes
        //person.addIndex(new ReflectionExtractor("getGender"), true, null);
        //person.addIndex(new ReflectionExtractor("getAgeDouble"), false, null);

        Random generator = new Random();

        for (int i = 1; i <= 10000; i++) {
            Person p = new Person(i, "Surname" + i, "Firstname" + i,
                                  "Address" + i, generator.nextInt(100) + 1,
                                  (generator.nextInt(2) == 1 ? Person.FEMALE :
Person.MALE) ) ;
            person.put(p.getId(),p);

        }

        System.out.println("The entry set size is " +  person.entrySet().size());

    }
}
```

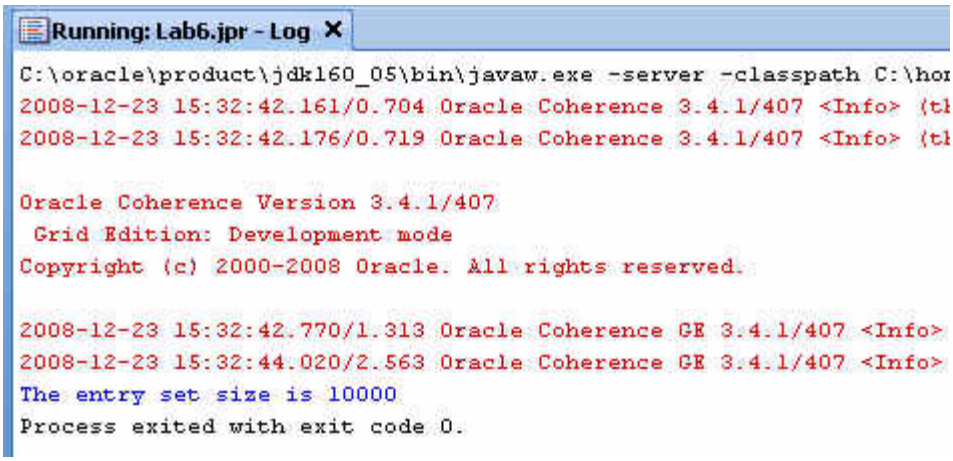
g. Stop all other cache servers and run `cache-server.sh`.

```
C:\oracle\product\coherence\bin>cache-server.cmd
```

h. Run `PopulatePeople` from the JDeveloper IDE.

You should see results similar to the following:

Figure 4–11 Results of Populating the Cache

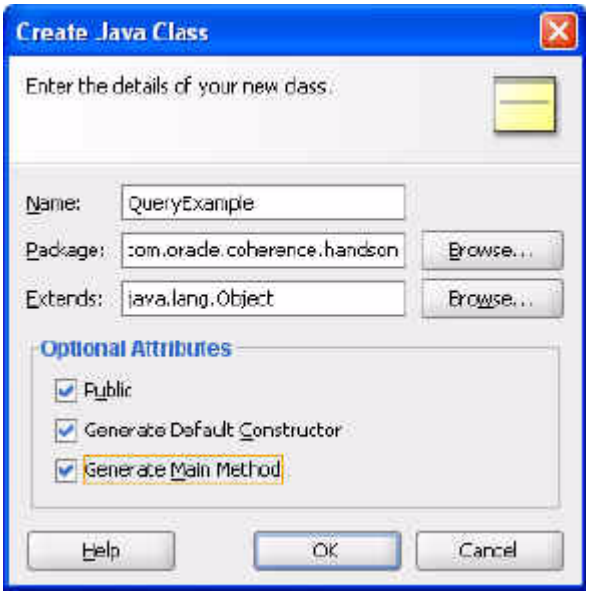


Can you think of a more efficient way of performing the 10,000 puts? Hint: See the information on bulk loading in *Pre-Loading the Cache* chapter of the *Developer's Guide for Oracle Coherence*.

2. Create a class to perform your queries.

   a. Create a new Java class called `QueryExample` with a `main` method.

Figure 4–12 Creating a Query Class



   b. Use the `entrySet` method to get the number of males, and the number of males aged 35 and above. Use the `size()` method to get the number of records returned. You will use the aggregation functions to perform this more efficiently in a later practice.

      Example 4–7 illustrates a possible solution.

Example 4–7 Sample QueryExample Class

```
package com.oracle.coherence.handson;
```

```
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import com.tangosol.util.filter.AndFilter;
import com.tangosol.util.filter.EqualsFilter;

import com.tangosol.util.filter.GreaterEqualsFilter;

import java.util.Set;

public class QueryExample {
    public QueryExample() {
    }

    public static void main(String[] args) {

        NamedCache person = CacheFactory.getCache("person");

        // get a set of all males
        Set males = person.entrySet(
                        new EqualsFilter("getGender",Person.MALE ));

        Set malesOver35 = person.entrySet(
                        new AndFilter(
                         new EqualsFilter("getGender",Person.MALE ),
                         new GreaterEqualsFilter("getAge",35)) );

        System.out.println("Total number of males is " + males.size());
        System.out.println("Total number of males > 35 " + malesOver35.size());
    }
}
```

c. Stop all running cache servers. Navigate to the `\oracle\product\coherence\bin` directory. Edit the `cache-server.cmd` file and modify the `-cp` entry on the line beginning with `$JAVAEXEC -server` to remove `\home\oracle\labs\Lab4\classes` from the `CLASSPATH` environment variable. Restart your cache server. What happens when you try to run the `QueryExample` code? You should get an error similar to the following:

```
Exception in thread "main" (Wrapped: Failed request execution for
DistributedCache service on Member(Id=1, Timestamp=2008-12-23
16:24:48.848, Address=130.35.99.248:8088, MachineId=49912,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:6024,
Role=CoherenceServer) (Wrapped) readObject failed:
java.lang.ClassNotFoundException: com.oracle.coherence.handson.Person
```

What happened and why? When a `QueryMap` is used, that is, when you use the `entrySet` method to retrieve a set of entries in the cache that map your request, the request is processed on the storage-enabled members, and then returned to the process that requested the set.

What happens here is that the Coherence cache server does not know yet about the `Person` object that you created. You will need to add the `Person` object into `CLASSPATH` of the Coherence cache server.

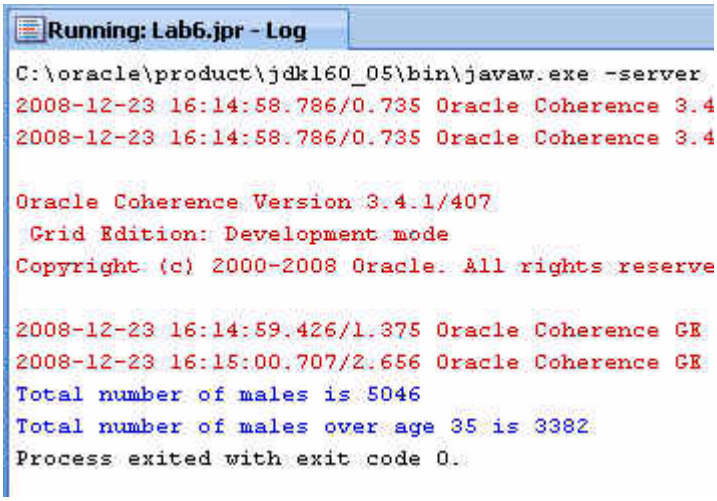
Edit `cache-server.cmd` file and add the `\home\oracle\labs\Lab4\classes` entry in the `$JAVAEXEC -server` command. The command should look similar to the following:

```
$JAVAEXEC -server -showversion $JAVA_OPTS -cp "$COHERENCE_
HOME/lib/coherence.jar:/home/oracle/labs/Lab4/classes"
```

```
com.tangosol.net.DefaultCacheServer $1
```

3. Restart your cache server, execute `PopulatePeople` and `QueryExample`, and see what happens.

Figure 4–13 Results of Executing the Query



4. Create a class to perform aggregations on the data in the cache.

   An `EntryAggregator` (`com.tangosol.util.InvocableMap.EntryAggregator`) enables you to perform operations on all or a specific set of objects and get an aggregation as a result. `EntryAggregators` are essentially agents that execute services in parallel against the data within the cluster.

   Aggregations are performed in parallel and can benefit from the addition of cluster members.

   There are two ways of aggregating: aggregate over a collection of keys or by specifying a filter. Example 4–8 illustrates the methods that perform these aggregations.

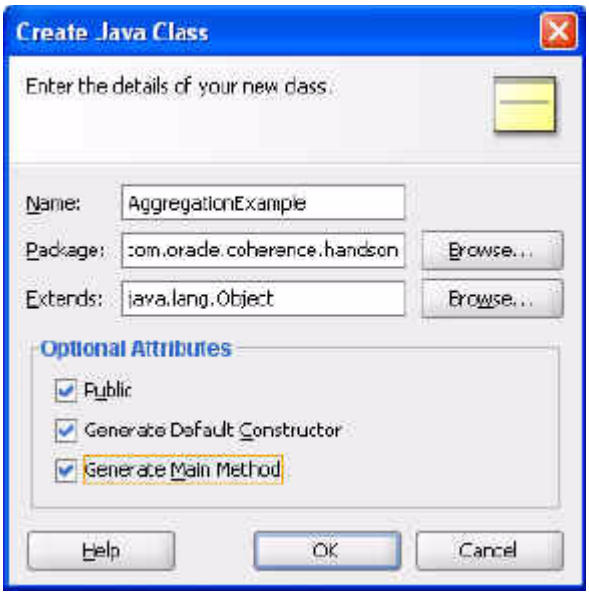
Example 4–8 Methods to Aggregate Over Keys or by Specifying Filters

```
Object aggregate(Collection keys, InvocableMap.entryAggregator agg)

Object aggregate(Filter filter, InvocableMap.entryAggregator agg)
```

   The following example uses a filter.

   a. Create a new Java class in the `Lab6` project called `AggregationExample`. Ensure that it has a `main` method.

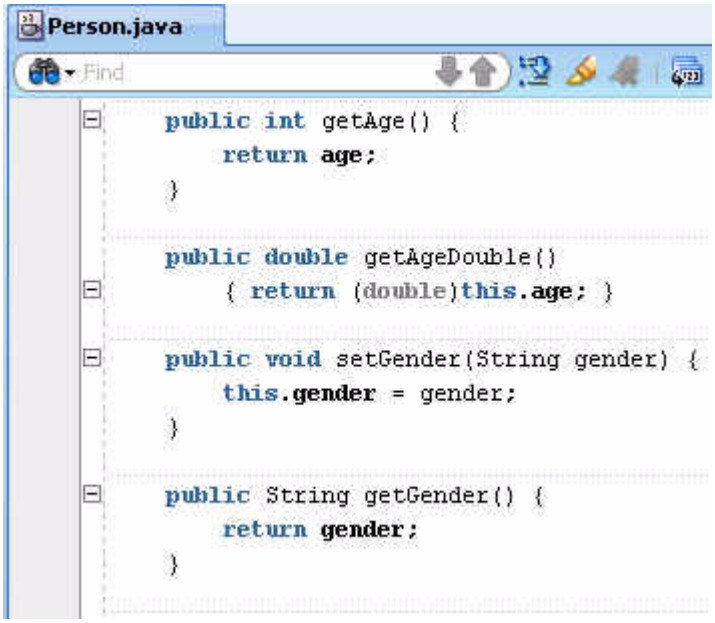
Figure 4–14 Creating an Aggregation Class



b. Create a new method in the `Person` class that you created in the earlier lab [`Lab4-Person.java`] so that you can use the correct aggregators. Call the method `getAgeDouble`.

```
public double getAgeDouble()
    { return (double)this.age; }
```

Figure 4–15 Creating the getAgeDouble Method



c. Stop and start the cache server after making this change.

d. In `AggregationExample.java`, write the code to get the following from the cache:

—Average age of all males

—Average age of all females

—Average age

—Maximum age

Hint 1: The following code gets the average age for all males:

```
Double averageAgeMales = (Double)person.aggregate( new
EqualsFilter("getGender",Person.MALE ), new DoubleAverage("getAgeDouble")
);
```

Hint 2: To query all the objects in a named cache, you can use either `(Filter)null` or a `new AlwaysFilter()`.

Example 4–9 illustrates a possible solution.

Example 4–9 Sample Data Aggregation Class

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.Filter;
import com.tangosol.util.aggregator.DoubleAverage;
import com.tangosol.util.aggregator.DoubleMax;
import com.tangosol.util.filter.AlwaysFilter;
import com.tangosol.util.filter.EqualsFilter;

import java.math.BigDecimal;

public class AggregationExample {
    public AggregationExample() {
    }

    public static void main(String[] args) {
        Double averageAgeMales = null;
        Double averageAgeFemales = null;
        Double maxAge = null;
        Double averageAge = null;
        int max = 100;

        NamedCache person = CacheFactory.getCache("person");
        long totalTime = 0;

        // create a new query

        for (int i = 0; i< max; i++) {
          long startTime = System.currentTimeMillis();

          averageAgeMales = (Double)person.aggregate(
                        new EqualsFilter("getGender",Person.MALE ),
                        new DoubleAverage("getAgeDouble")
                        );

          averageAgeFemales = (Double)person.aggregate(
                        new EqualsFilter("getGender",Person.FEMALE ),
                        new DoubleAverage("getAgeDouble")
                        );

          maxAge = (Double)person.aggregate(
                        (Filter)null,   // new AlwaysFilter()
```

```
                        new DoubleMax("getAgeDouble")
                        );

          averageAge = (Double)person.aggregate(
                        new EqualsFilter("getGender",Person.FEMALE ),
                        new DoubleAverage("getAgeDouble")
                        );


          long endTime = System.currentTimeMillis();

          System.out.println("Total time taken is " + (endTime - startTime) /
1000F + " seconds");
          totalTime += (endTime - startTime);
        }

        System.out.println("Average time is " +  (totalTime / max) / 1000F);

        //System.out.println("Average age of males is " + averageAgeMales);
        //System.out.println("Average age of femals is " + averageAgeFemales);
        //System.out.println("Max age is " + maxAge);
        //System.out.println("Average age  " + averageAge);
    }
}
```

5. Observe what happens when you add new cache members.

   a. Modify `AggregationExample` to put in timing information. Run the queries in a loop for 100 times and get an average time. This ensures that garbage collection or normal machine spikes will not influence the results.

      Hint: You can get the current time using the following code:

      ```
      long timeStart = System.currentTimeMillis();
      ```

      This line is illustrated in bold font in Example 4–9.

   b. Run your code with one cache server and note the average time.

Figure 4–16 Aggregation Process Run with One Cache Server

```
Total time taken is 0.0 seconds
Total time taken is 0.015 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.016 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.0 seconds
Total time taken is 0.078 seconds
Total time taken is 0.016 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.016 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.016 seconds
Average time is 0.018
Process exited with exit code 0.
```

Start up a second cache server, allow for the data to be redistributed, and rerun the code. What happens?

Figure 4–17 Aggregation Process Run with Two Cache Servers

```
Total time taken is 0.015 seconds
Total time taken is 0.032 seconds
Total time taken is 0.015 seconds
Total time taken is 0.032 seconds
Total time taken is 0.015 seconds
Total time taken is 0.016 seconds
Total time taken is 0.031 seconds
Total time taken is 0.016 seconds
Total time taken is 0.031 seconds
Total time taken is 0.062 seconds
Total time taken is 0.016 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.032 seconds
Average time is 0.034
Process exited with exit code 0.
```

Note: If you do not have a dual core machine, you may not see an improvement in aggregation times. Why do you think this happens? The following are some of the sample timings:

c. What happens if you increase the number of cache servers to three?

Figure 4–18 Aggregation Process Run with Three Cache Servers

```
Total time taken is 0.031 seconds
Total time taken is 0.032 seconds
Total time taken is 0.078 seconds
Total time taken is 0.031 seconds
Total time taken is 0.031 seconds
Total time taken is 0.047 seconds
Total time taken is 0.016 seconds
Total time taken is 0.062 seconds
Total time taken is 0.047 seconds
Total time taken is 0.078 seconds
Total time taken is 0.031 seconds
Total time taken is 0.032 seconds
Total time taken is 0.015 seconds
Total time taken is 0.094 seconds
Average time is 0.047
Process exited with exit code 0.
```

6. Add indexes to the `PopulatePeople` class to improve performance. Hints: Find the `addIndex` method in the Javadoc for the `QueryMap` interface. Note that when you add indexes, you should see a significant improvement in performance. Un-comment the index code lines and re-run your application.

```
// add indexes
person.addIndex(new ReflectionExtractor("getGender"), true, null);
```

```
person.addIndex(new ReflectionExtractor("getAgeDouble"), false, null);
```

Figure 4–19 Aggregation Process Run with One Cache Server and Indexing

```
Total time taken is 0.0 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.0 seconds
Total time taken is 0.016 seconds
Total time taken is 0.016 seconds
Total time taken is 0.0 seconds
Total time taken is 0.015 seconds
Total time taken is 0.016 seconds
Total time taken is 0.0 seconds
Total time taken is 0.016 seconds
Average time is 0.014
Process exited with exit code 0.
```

Figure 4–20 Aggregation Process Run with Three Cache Servers and Indexing

```
Total time taken is 0.031 seconds
Total time taken is 0.047 seconds
Total time taken is 0.016 seconds
Total time taken is 0.015 seconds
Total time taken is 0.188 seconds
Total time taken is 0.031 seconds
Total time taken is 0.047 seconds
Total time taken is 0.015 seconds
Total time taken is 0.031 seconds
Total time taken is 0.032 seconds
Total time taken is 0.031 seconds
Average time is 0.042
Process exited with exit code 0.
```

5

Observing Data Changes

In this chapter, you observe data changes within a `NamedCache`. This chapter contains the following sections:

Introduction

The `com.tangosol.util.ObservableMap` interface enables you to observe and take action on the changes made to cache entries. It extends `java.util.EventListener` and uses the standard bean event model within Java. All types of `NamedCaches` implement this interface. To listen for an event, you register a `MapListener` (`com.tangosol.util.MapListener`) on the cache.

There are three ways to listen for events:

- Listen for all events
- Listen for all events that satisfy a filter
- Listen for events on a particular object key

The methods listed in Example 5–1 (which implement the preceding list) can be used on a `NamedCache`:

Example 5–1 Listener Methods on a NamedCache

```
void addMapListener(MapListener listener)

void addMapListener(MapListener listener, Filter filter, boolean fLite)

void addMapListener(MapListener listener, Object oKey, boolean fLite)
```

The `com.tangosol.util.MapEvent` class captures the object key, and the old and new values. You can specify a "Lite" event, in which the new and old values may not be present. Example 5–2 describes a pattern for registering these methods against a `NamedCache`. This has been done as an anonymous class.

Example 5–2 Code Pattern for Registering an Event

```
namedCache.addMapListener(new MapListener() {
public void entryDeleted(MapEvent mapEvent) {
```

```
      //TODO... handle deletion event
}
public void entryInserted(MapEvent mapEvent) {
    //TODO... handle inserted event
}
public void entryUpdated(MapEvent mapEvent)
{
     //TODO... handle updated event } });
```

You can use the `getOldValue()` or `getNewValue()` methods in the preceding `MapEvent` class to get the entry for which the event gets fired.

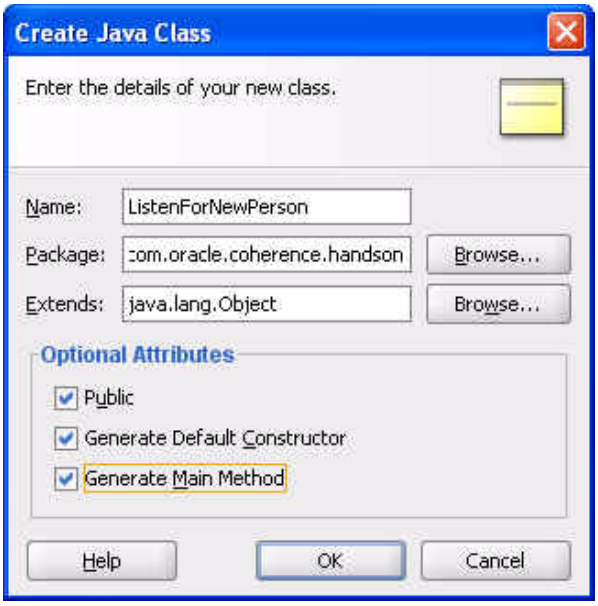
Creating a Cache Listener and Responding to Changes

This section describes how to create a Java class that listens on a `NamedCache` and responds to any changes it detects.

To create a Java class that listens to the cache and responds to changes:

1. In the `Lab6` project, create a new class that listens for a new `Person` object entry.
   a. Create a new Java class called `ListenForNewPerson`. Ensure that it has a `main` method.

Figure 5–1 Creating a Listener Class



   b. Within this class, add a listener to print out a message whenever a new `Person` is added to the cache.

      Hint: Use the following code to keep the Java process running until you read from the console; otherwise your program exits immediately.

```
BufferedReader console = new BufferedReader(new
InputStreamReader(System.in)); String text = console.readLine();
```

      Example 5–3 illustrates a possible solution.

Example 5–3 Sample Listener Class

```
package com.oracle.coherence.handson;
```

```
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.MapEvent;
import com.tangosol.util.MapListener;
import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.filter.MapEventFilter;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ListenForNewPerson {
    public ListenForNewPerson() {
    }

    public static void main(String[] args) throws IOException {
        // connect to named cache
        NamedCache person = CacheFactory.getCache("person");

        // listen for insert events on Person
        // This can be done in an easier way by using a new AbstractMapListener()
        // and then overriding only the method you want to
        //
        person.addMapListener(new MapListener()
        {
          public void entryDeleted(MapEvent mapEvent) {
            // ignore
          }
          public void entryInserted(MapEvent mapEvent) {
             Person p = (Person)mapEvent.getNewValue();
             System.out.println("New person added: " + p.getFirstname() + " " +
p.getLastname());
          }
          public void entryUpdated(MapEvent mapEvent) {
             // ignore
          }
        }
        );


        System.out.println("waiting for events");
        BufferedReader console = new BufferedReader(new
InputStreamReader(System.in));
        String text = console.readLine();

    }
}
```

c. To enable the console input, you must perform the following:

—Right-click the `Lab6` project and select **Project Properties**.

—Select **Run/Debug/Profile** at the left.

—Click the **Edit** button at the right and click **Tool Settings**. Ensure that the **Allow Program Input** check box in the **Edit Run Configuration** dialog box is selected.

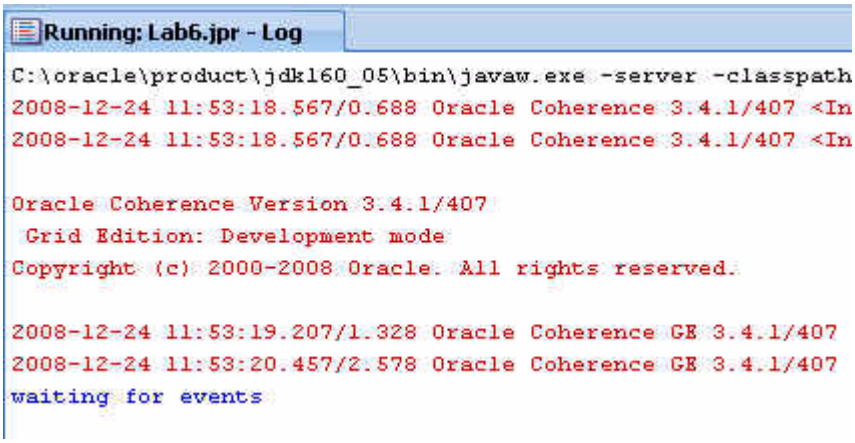
—Click **OK** in the **Edit Run Configuration** dialog box and in the **Project Properties** dialog box to save your changes.

d. Start the cache server if it is not already running.

```
C:\oracle\product\coherence\bin>cache-server.cmd
```

e. Run the `ListenForNewPerson` Java class. You should see the **Input** area at the bottom of the messages window. This is where you can input information from the console.

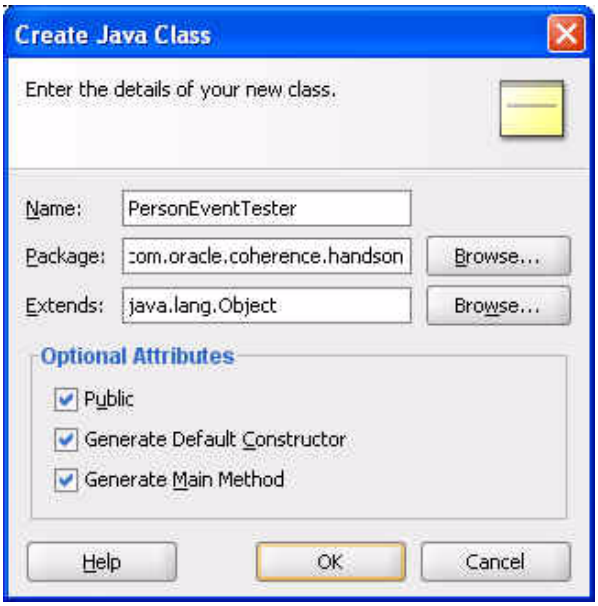
Figure 5–2 Listener Program Waiting for Events



2. In `Lab6`, create a class that adds and removes entries in a cache.

   a. In `Lab6`, create a class called `PersonEventTester` that puts a new `Person` object in the cache. Ensure that it has a `main` method.

Figure 5–3 Creating a PersonEventTester Class



Example 5–4 illustrates possible code for the `PersonEventTester` class.

Example 5–4 Sample Program to Put a New Object in the Cache

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
```

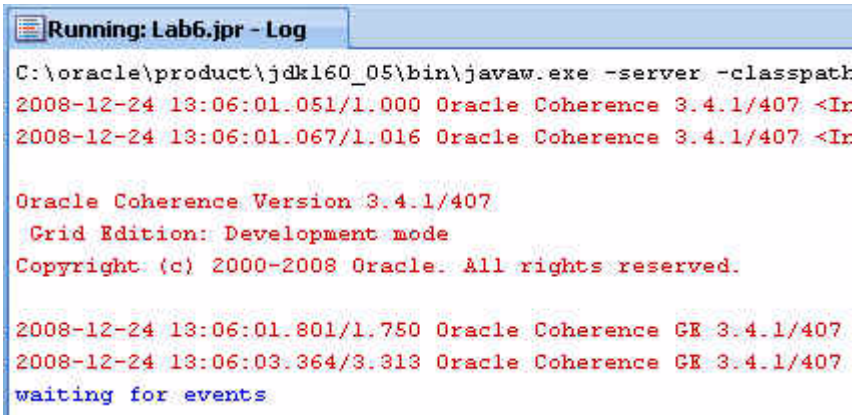
```
import com.tangosol.net.NamedCache;

public class PersonEventTester {
    public PersonEventTester() {
    }

    public static void main(String[] args) {

        NamedCache person = CacheFactory.getCache("person");

        Person p1 = new Person(1,"Middleton","Tim",
                  "Level 2, 66 Kings Park Road, West Perth",
                  39,Person.MALE);

       System.out.println("put person");
       person.put(p1.getId(),p1);

       Person p2 = (Person)person.get(p1.getId());
       p2.setFirstname("Timothy");

        System.out.println("Update person");
       person.put(p2.getId(),p2);
    }
}
```

b. Perform the following steps to test the `ListenForNewPerson` and `PersonEventTester` classes.

—Restart your cache server.

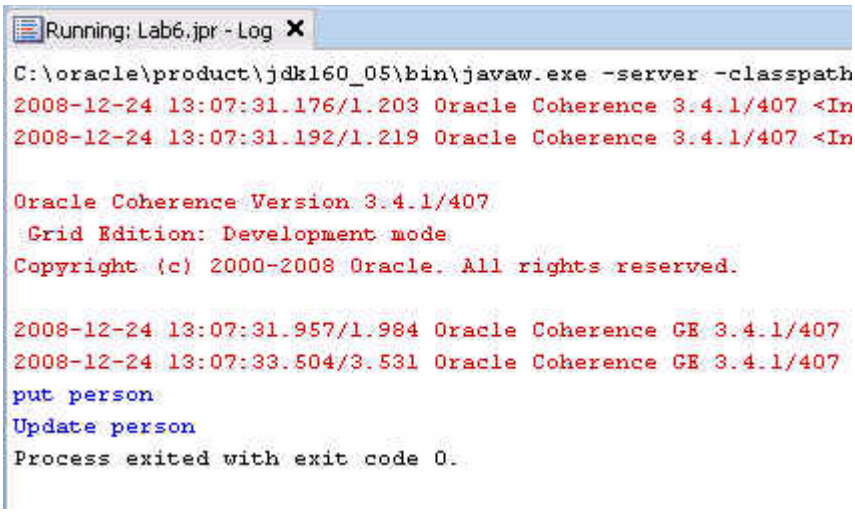
—Run your `ListenForNewPerson` class. Do not input any value through the **Input** area at the bottom of the messages window.

Figure 5–4 Output from the ListenForNewPerson Class



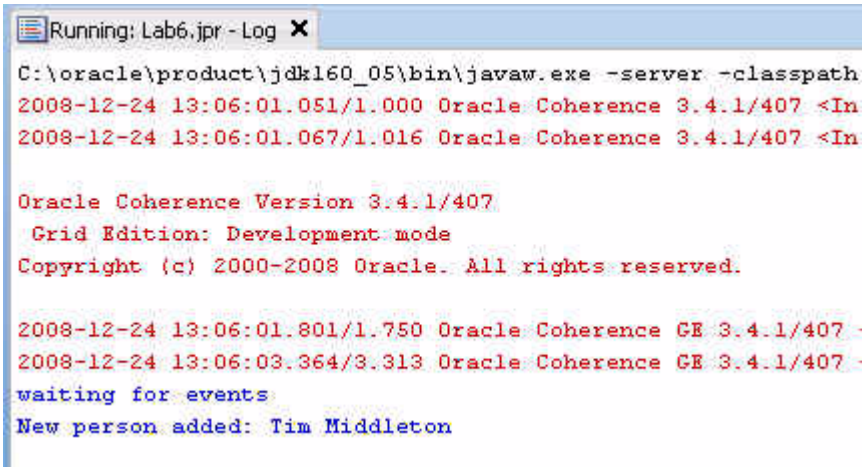
—Run the `PersonEventTester` to create a new record in the cache. What happens?

Figure 5–5 Output from the PersonEventTester Class



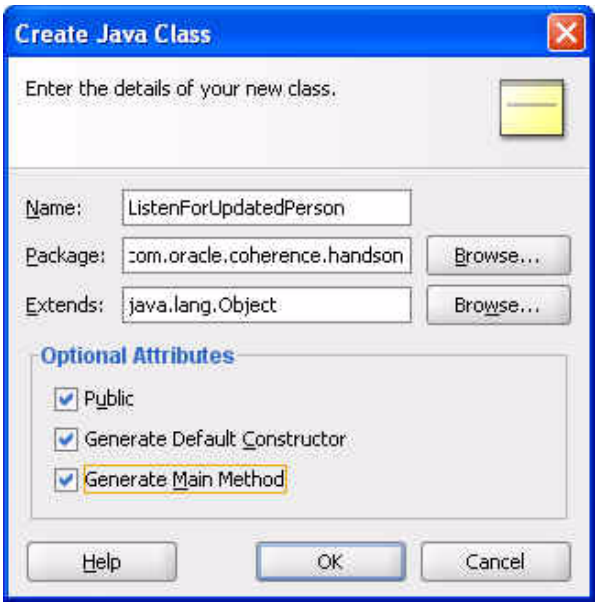
—You should see a message in the `ListenForNewPerson` messages window indicating that a new record has been added.

Figure 5–6 New Record Detected by the ListenForNewPerson Class



3. In `Lab6`, create a class that listens for an update to the `Person` object.
   a. a) In `Lab6`, create a new class called `ListenForUpdatedPerson`. Ensure that it has a `main` method.

Figure 5–7 Creating an ListenForUpdatedPerson Class



b. In the class, add a `MapListener` to the person cache that prints out the new and old values of a male `Person` when that person object is updated.

Hint: You must supply two new parameters to the `addListener` method: a new `MapEventFilter()` parameter and a parameter indicating whether you want a lite event

Example 5–5 illustrates a possible solution.

Example 5–5 Sample Code that Listens for an Update to an Object

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.AbstractMapListener;
import com.tangosol.util.MapEvent;
import com.tangosol.util.MapListener;
import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.filter.MapEventFilter;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ListenForUpdatedPerson {
    public ListenForUpdatedPerson() {
    }

    public static void main(String[] args) throws IOException {
        // connect to named cache
        NamedCache person = CacheFactory.getCache("person");

        // listen for insert events on Person
        // This can be done in an easier way by using a new AbstractMapListener()
        // and then overriding only the method you want to
        //
        person.addMapListener(new AbstractMapListener()
```

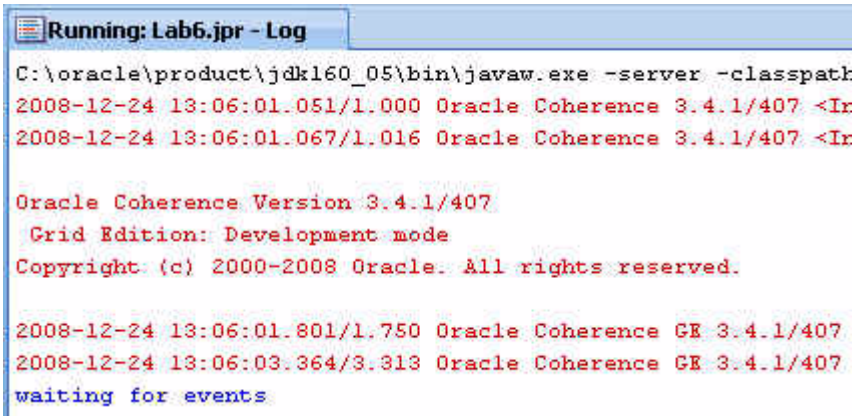
```
        {
              public void entryUpdated(MapEvent mapEvent) {
                Person oldPerson = (Person)mapEvent.getOldValue();
                 Person newPerson = (Person)mapEvent.getNewValue();

                 // better to implement toString() on the person object to display
it.. :)
                 System.out.println("Old person is " + oldPerson.getFirstname() +
" " +
                                          oldPerson.getLastname());
                 System.out.println("New person is " + newPerson.getFirstname() +
" " +
                              newPerson.getLastname());
               }
           },
          new MapEventFilter(MapEventFilter.E_UPDATED, new
EqualsFilter("getGender", Person.MALE)),
          false   // not a lite event

        );


        System.out.println("waiting for events");
        BufferedReader console = new BufferedReader(new
InputStreamReader(System.in));
        String text = console.readLine();


    }
}
```

c. Next, perform the following:

—Restart your cache server.

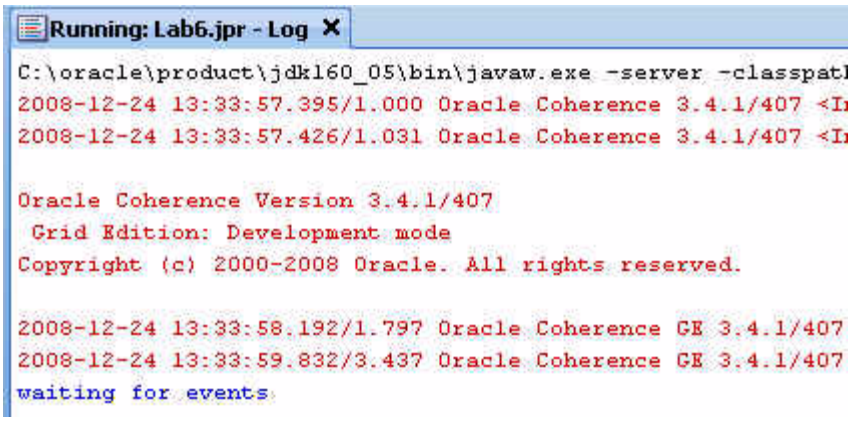
—Run the `ListenForNewPerson` class. Do not input any value through the **Input** area at the bottom of the messages window.

Figure 5–8 Output of the ListenForNewPerson Class



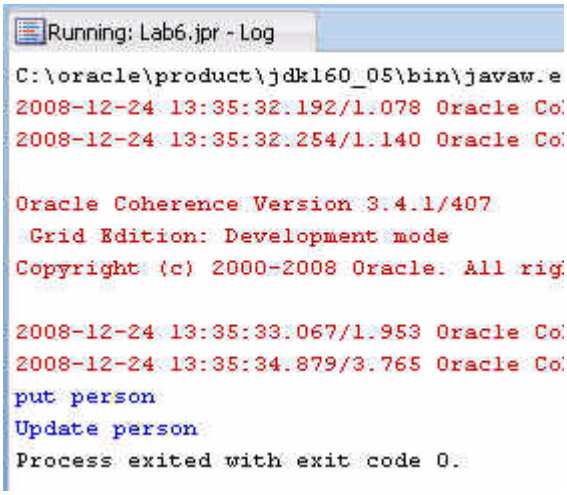
—Run the `ListenForUpdatedPerson` class. Do not input any value through the **Input** area at the bottom of the messages window.

Figure 5–9 Output of the ListenForUpdatedPerson Class



—Run the `PersonEventTester` class.

Figure 5–10 Output from PersonEventTester Class



—You should see appropriate messages in the correct windows.

Figure 5–11 Output from the ListenForNewPerson Class

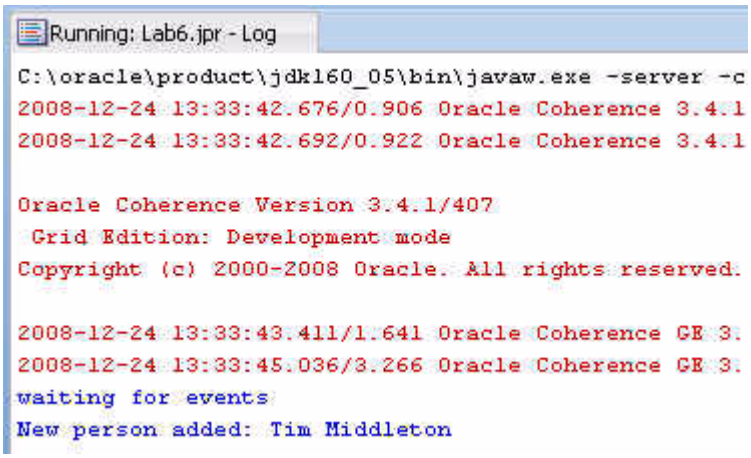
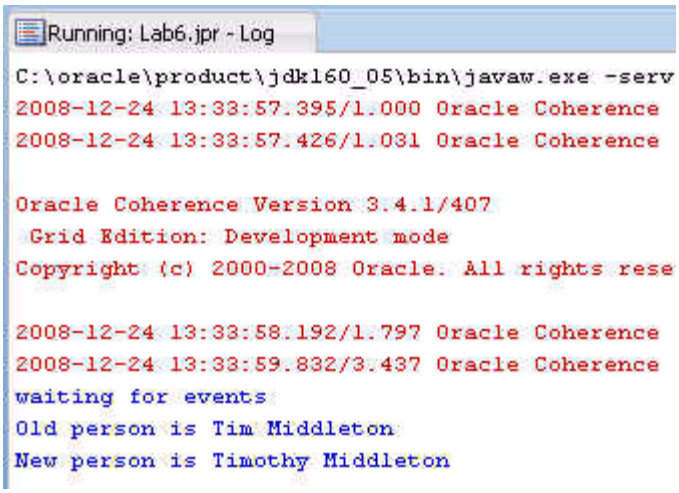


Figure 5–12 Output from the ListenForUpdatedPerson Class



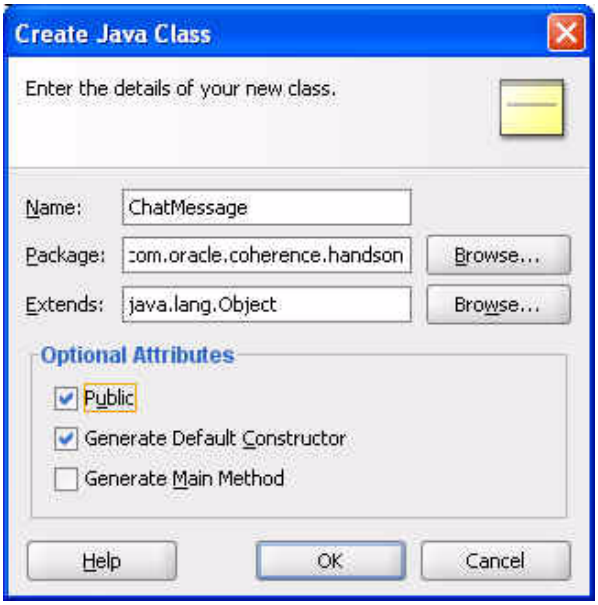
Creating a Chat Program

In this exercise, you create your own scalable and robust chat program. You should now have all the elements of a basic chat program.

To create a chat program:

1. In `Lab6`, create a `ChatMessage` Java class to store the chat messages.

Figure 5–13 Creating a Chat Program



Example 5–6 illustrates a possible solution.

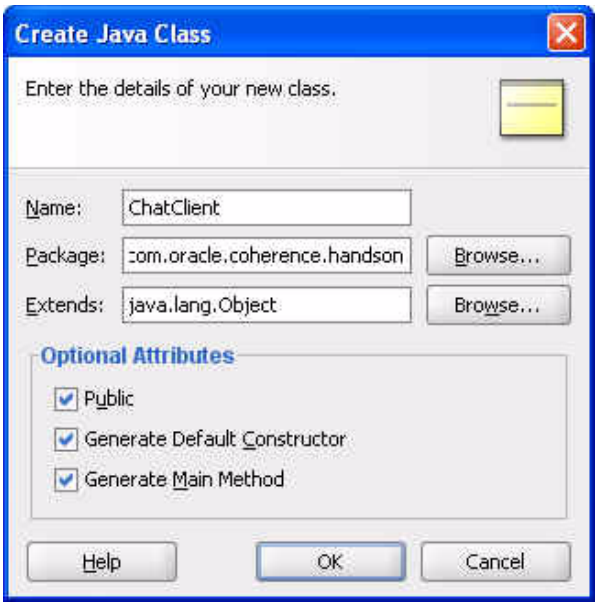
Example 5–6 Sample Chat Program

```
package com.oracle.coherence.handson;

import java.io.Serializable;
```

```
public class ChatMessage implements Serializable {

    private String from;
    private long entryTime;
    private String message;

    public ChatMessage() {
    }

    public ChatMessage(String from, String message) {
        this.from = from;
        this.message = message;
        this.entryTime = System.currentTimeMillis();
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public String getFrom() {
        return from;
    }

    public void setEntryTime(long entryTime) {
        this.entryTime = entryTime;
    }

    public long getEntryTime() {
        return entryTime;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }
}
```

2. Stop all running cache servers. Navigate to the `/oracle/product/coherence/bin` directory and edit `cache-server.cmd`. Modify the `CLASSPATH` environment variable to include `/home/oracle/labs/Lab6/classes`. Restart the cache server.
3. Create a `ChatClient` Java class. Ensure that it has a `main` method.

Figure 5–14 Creating a Chat Client Program



Write the client program so that it performs the following:

- Gets a person's name.
- Sets up a new `MapListener` to receive messages that are posted in the chat.
- Loops and reads a message from the command line and posts this to the `messages` named cache (Exit when the user enters `bye`.)
- Runs multiple copies of this and observes the behavior.
- You can add some extra features if you wish. For example:
  - Add a facility to list all users in the chat. (You would need a second named cache.)
  - Send a private message to a named individual.

  Example 5–7 illustrates a possible solution:

Example 5–7 Sample Chat Client Program

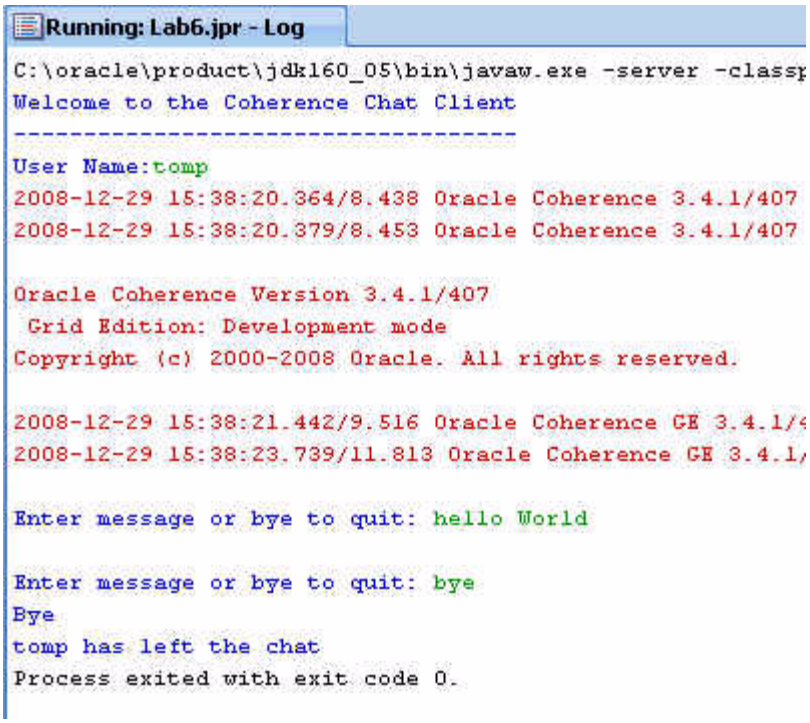
```
package com.oracle.coherence.handson;

import com.tangosol.net.*;

import java.io.*;
import com.tangosol.util.UUID;
import com.tangosol.util.filter.*;
import com.tangosol.util.MapEvent;
import com.tangosol.util.AbstractMapListener;

import java.util.Date;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

public class ChatClient {

    public static void main(String[] args) {
```

```
        String userName = null;
        String message;
        NamedCache chatmembers = null;

         try {

             System.out.println("Welcome to the Coherence Chat Client");
             System.out.println("-----------------------------------");

             BufferedReader console = new BufferedReader(new
InputStreamReader(System.in));

             System.out.print("User Name:");
             userName = console.readLine();

             // join the chatroom named cache as storage enabled = true
             NamedCache cache = CacheFactory.getCache("chatroom");
             chatmembers = CacheFactory.getCache("chatmembers");

             chatmembers.put(userName,userName);

             // register a listener to display the messages
             cache.addMapListener(new AbstractMapListener() {
                     public void entryInserted(MapEvent event) {
                        ChatMessage msg = (ChatMessage)event.getNewValue();

                        System.out.println("From: " + msg.getFrom());
                        System.out.println("Time: " + new Date(msg.getEntryTime()));
                        System.out.println("Mesg: " + msg.getMessage() );

                        System.out.println();
                      } }
                      ,
                      new MapEventFilter(MapEvent.ENTRY_INSERTED, new
NotEqualsFilter("getFrom", userName)),
    false);

             chatmembers.addMapListener( new AbstractMapListener() {
                   public void entryDeleted(MapEvent event) {
                       String who = (String)event.getOldValue();
                       System.out.println(who + " has left the chat");
                   }

                   public void entryInserted(MapEvent event) {
                       String who = (String)event.getNewValue();
                       System.out.println(who + " has entered the chat");
                   }
                }
             );

             do {
                 System.out.print("\nEnter message or bye to quit: ");
                 message = console.readLine();

                 if ("bye".equals(message))
                    break;
                 // else add this to the chat
                 else if ("help".equals(message)) {
                     System.out.println("HELP:");
                     System.out.println("bye - quit");
```

```
                    System.out.println("who - list of users in the chat\n");
                }
                else if ("who".equals(message)) {
                    System.out.println("Current chat memebers");
                    System.out.println("====================");
                    Set s = chatmembers.entrySet();
                    for (Iterator<Map.Entry> entries =
chatmembers.entrySet().iterator() ; entries.hasNext();) {
                        Map.Entry entry = entries.next();
                        String member = (String)entry.getValue();

                        System.out.println(member);
                    }
                }
                else {

                    cache.put(new UUID(),new ChatMessage(userName, message));

                }

            } while (true);

            System.out.println("Bye");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            chatmembers.remove(userName);
        }
    }
}
```

Figure 5–15 illustrates the output of the chat program.

Figure 5–15 Output of the Chat Program



Working with Partitions and Composite Keys

In this exercise, you extend objects to have composite keys, rather than a single value. Using JDeveloper, you perform the following:

- Create a new `Person` class and implement a `Person.Key` inner class
- Use this new `Person` class to put and get values
- Enable data affinity—Keep related data together in the same partition.

In all the examples so far, you have used single attributes for the key, such as person ID or name. If you want to store different versions of objects or have composite keys, there are several (bad) ways in which you "can" do this, such as storing versions as `Fred1` and `Fred-2`, or strings such as `Fred-Jones`. Using this method to get a key for a particular individual requires some string mangling, which could get messy and potentially give suboptimal performance.

Rather than using a single key, as in the preceding case, you can create a class that encapsulates the business logic of that key and implement this as your key. This helps you to future-proof your objects against changes. For example, you can create an inner class that provides you a key.

```
Person.Key(int id, int version)
```

An interesting side-effect of this is that you can then make your key implement the `KeyAssociation` interface, and use the `getAssociatedKey()` method to provide you with partition affinity on an attribute of a key. For example, you can potentially store all people with the same last name or any common field within the same partition. The code in Example 5–8 defines an inner class called `Key` for the `Person` class based on the person ID and a version number.

Example 5–8 Inner Class to Implement a Key Association for Data Affinity

```
public static class Key implements PortableObject, KeyAssociation {
    //define a key of id and version
    private int id;
    private int version;
    private String lastname;

    public Key() {
        //for serializble
    }

    public Object getAssociatedKey() {
        return lastname;
    }

    public Key(int id, int version) {
        this.id = id;
        this.version = version;
    }

    public Key(Person p) {
        this.id = p.getId();
        this.version = 1;
    // default this.lastname = p.getLastname();
    }

    public void writeExternal(PofWriter dataOutput) throws IOException {
        dataOutput.writeInt(0, this.id);
        dataOutput.writeInt(1, this.version);
        dataOutput.writeString(2, this.lastname);
    }

    public void readExternal(PofReader dataInput) throws IOException {
        this.id = dataInput.readInt(0);
        this.version = dataInput.readInt(1);
        this.lastname = dataInput.readString(2);
    }

    @Override public boolean equals(Object object) {
        . . .
    }

    @Override
    public int hashCode() {
        final int PRIME = 37;
        int result = 1;
        return result;
    }
}
```

Example 5–9 illustrates how to create and use the inner class within Coherence:

Example 5–9 Sample Code to Create and Use an Inner Class within Coherence

```
...
NamedCache person = CacheFactory.getCache("person");
Person p = new Person(1,"Middleton","Tim","Address",29,Person.MALE);
person.put(p.getKey(), p);
Person p2 = (Person)person.get(new Person.Key(1,1));
System.out.println("Person is " + p2.getLastname());
```

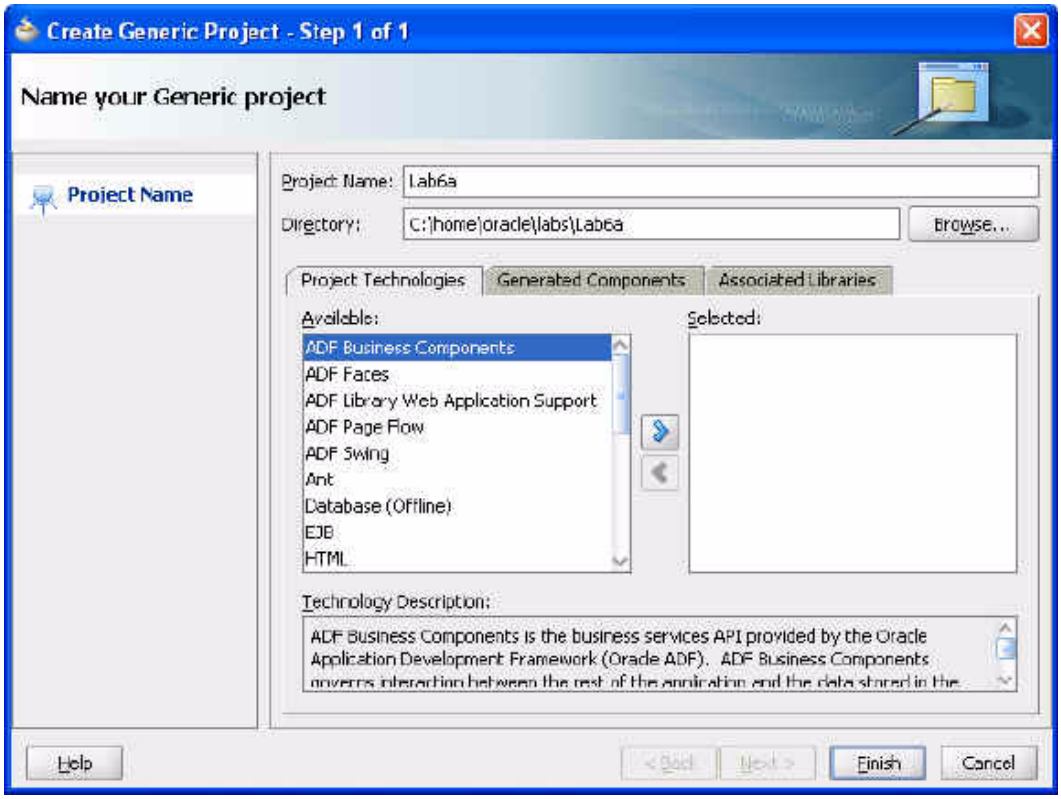
...

You can then modify the inner class to implement `KeyAssociation`, and then provide an implementation of the `getAssociatedKey` method. If `getAssociatedKey()` returns a last name, this ensures that all people with the same last name reside in the same key partition and therefore, the same member.

To create partitions and keys on data for use with Coherence:

1. Copy the existing `Person` class from `Lab4` into a new project.
   a. Create a new project called `Lab6a`.

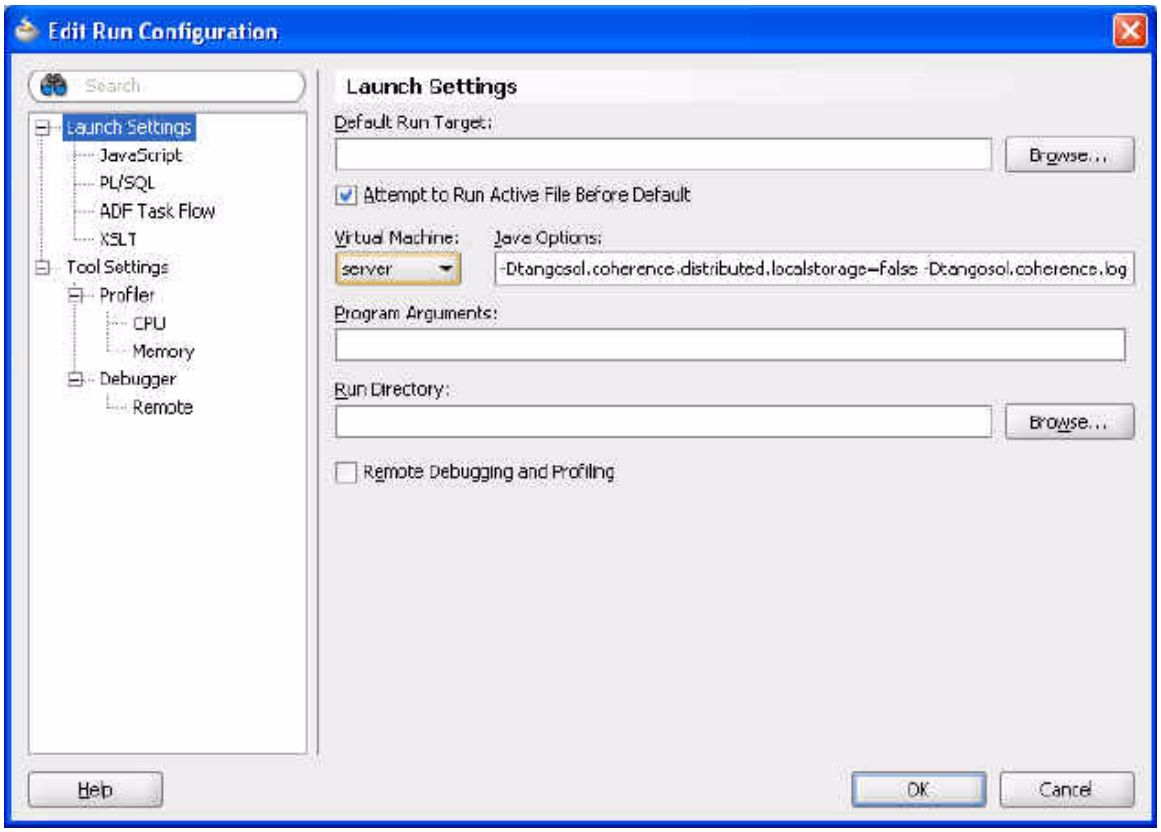
Figure 5–16 Creating a New Project



   b. Make sure that you change the default local storage properties in the **Java Options** field in the **Edit Run Configuration** dialog box.

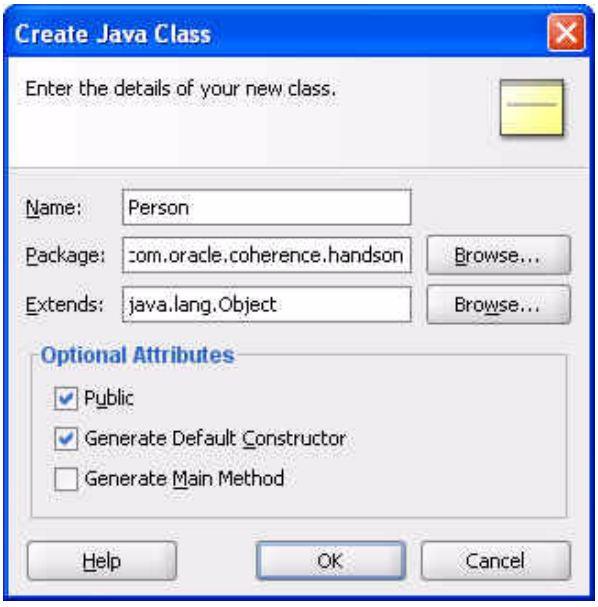
```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

Figure 5–17 Turning Off Local Storage and Setting Log Level for the Runtime Configuration



c. Create a new class called `Person`.

Figure 5–18 Creating the Person Class



d. Copy and paste the source from the `Person.java` file from `Lab4` into this class. Modify the `Person.java` file as shown in the given code snippet for Person class.

e. Implement the `Key` class.

f. Ensure that you implement the `getKey` method in the `Person` class.

```
public Key getKey() {
    return new Key(this);
}


public void setId(int param) {
    this.id = param;
}
```

2. Implement the `KeyAssociation` class for data affinity.

   a. Modify your `Person.Key` class to implement the `KeyAssociation` interface. You must do the following:

      —Implement the `getAssociatedKey` method within the `Person.Key` class

      —Add a new attribute to the `Key` class called `Lastname`

      —Initialize `Lastname` in the constructor

   b. Compile your new `Person` class.

   c. Restart your cache servers because the definition of the `Person` object has changed.

      Example 5–10 illustrates a possible implementation of the `Person` class.

Example 5–10 Implementation of the Person Object with a Key Inner Class

```
package com.oracle.coherence.handson;

import java.io.IOException;
import com.tangosol.net.cache.KeyAssociation;
import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;

public class Person implements PortableObject {

   private int id;
   private String lastname;
   private String firstname;
   private String address;
   private int age;
   private String gender;

   public static String MALE="M";
   public static String FEMALE="F";


    public Person() {
    }

    public Person(int id1, String lastname1, String firstname1, String address1,
                  int age1, String gender1) {
        super();
        this.id = id1;
        this.lastname = lastname1;
        this.firstname = firstname1;
```

```
        this.address = address1;
        this.age = age1;
        this.gender = gender1;
    }

    public void setId(int param) {
        this.id = param;
    }

    public Key getKey() {
        return new Key(this);
    }
    public int getId() {
        return id;
    }

    public void setLastname(String param) {
        this.lastname = param;
    }

    public String getLastname() {
        return lastname;
    }

    public void setFirstname(String param) {
        this.firstname = param;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setAddress(String param) {
        this.address = param;
    }

    public String getAddress() {
        return address;
    }

    public void setAge(int param) {
        this.age = param;
    }

    public int getAge() {
        return age;
    }

    public void setGender(String param) {
        this.gender = param;
    }

    public String getGender() {
        return gender;
    }

    public double getAgeDouble() {
        return this.age;
    }
```

```
    @Override
    public boolean equals(Object object) {
        if (this == object) {
            return true;
        }
        if (!(object instanceof Person)) {
            return false;
        }
        final Person other = (Person)object;
        if (id != other.id) {
            return false;
        }
        if (!(lastname == null ? other.lastname == null :
lastname.equals(other.lastname))) {
            return false;
        }
        if (!(firstname == null ? other.firstname == null :
firstname.equals(other.firstname))) {
            return false;
        }
        if (!(address == null ? other.address == null :
address.equals(other.address))) {
            return false;
        }
        if (age != other.age) {
            return false;
        }
        if (!(gender == null ? other.gender == null :
gender.equals(other.gender))) {
            return false;
        }
        return true;
    }

    @Override
    public int hashCode() {
        final int PRIME = 37;
        int result = 1;
        result = PRIME * result + ((lastname == null) ? 0 : lastname.hashCode());
        result = PRIME * result + ((firstname == null) ? 0 :
firstname.hashCode());
        result = PRIME * result + ((address == null) ? 0 : address.hashCode());
        result = PRIME * result + ((gender == null) ? 0 : gender.hashCode());
        return result;
    }

    public void readExternal(PofReader pofReader) throws IOException{
        this.id = pofReader.readInt(0);
        this.lastname = pofReader.readString(1);
        this.firstname = pofReader.readString(2);
        this.address = pofReader.readString(3);
        this.age = pofReader.readInt(4);
        this.gender = pofReader.readString(5);
    }

    public void writeExternal(PofWriter pofWriter) throws IOException{
        pofWriter.writeInt(0, this.id);
        pofWriter.writeString(1, this.lastname);
        pofWriter.writeString(2, this.firstname);
        pofWriter.writeString(3, this.address);
```

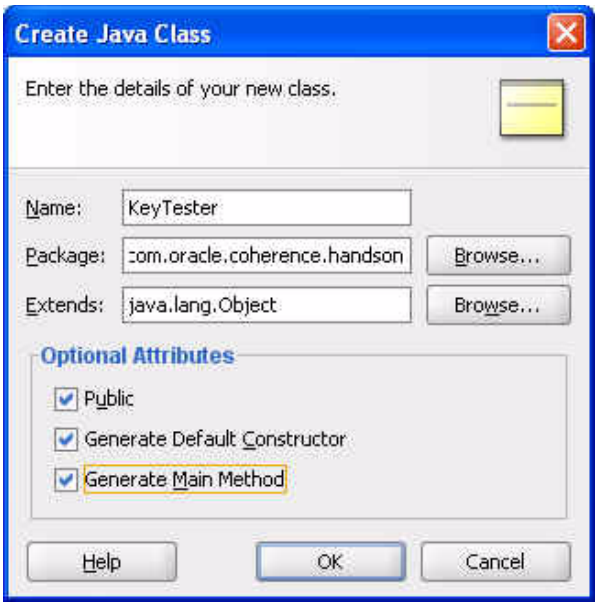
```
        pofWriter.writeInt(4, this.age);
        pofWriter.writeString(5, this.gender);
    }

    public static class Key implements PortableObject, KeyAssociation {

            // lets define a key of id and version
            private int id;
            private int version;
            private String lastname;

            public Key() {
                    //for serializble
            }

            public Object getAssociatedKey() {
               return lastname;
            }


            public Key(int id, int version) {
               this.id = id;
               this.version = version;
            }

            public Key(Person p) {
                this.id = p.getId();
                this.version = 1;  // default
                this.lastname = p.getLastname();
            }

            public void writeExternal(PofWriter dataOutput) throws IOException {
                dataOutput.writeInt(0, this.id);
                dataOutput.writeInt(1, this.version);
                dataOutput.writeString(2, this.lastname);
            }

            public void readExternal(PofReader dataInput) throws IOException {
                this.id = dataInput.readInt(0);
                this.version = dataInput.readInt(1);
                this.lastname = dataInput.readString(2);
            }

            @Override
            public boolean equals(Object object) {
                if (this == object) {
                    return true;
                }
                if (!(object instanceof Person.Key)) {
                    return false;
                }
                final Person.Key other = (Person.Key)object;
                if (id != other.id) {
                    return false;
                }
                if (version != other.version) {
                    return false;
                }
                if (lastname != other.lastname) {
                    return false;
```

```
                }
                return true;
            }

            @Override
            public int hashCode() {
                final int PRIME = 37;
                int result = 1;
                return result;
            }
        }
}
```

3. Create a `KeyTester` class in your project to test your new key.

   In the `KeyTester` class, implement the code to create a new `Person` object, and put and get the object from the cache. The `KeyTester` class must have a `main` method.

Figure 5–19 Creating a KeyTester Class



Example 5–11 illustrates a possible implementation.

Example 5–11 Sample Code to Create an Object and Put and Get it From the Cache

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class KeyTester {

    public KeyTester() {
    }

    public static void main(String[] args) {

        NamedCache person = CacheFactory.getCache("person");
```

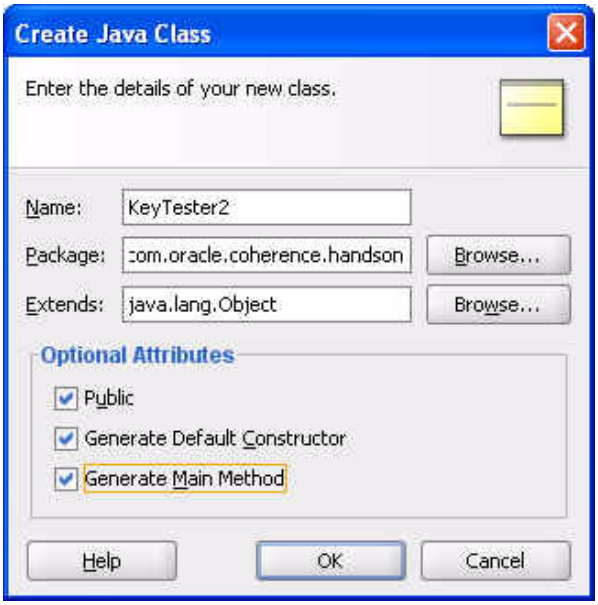
```
        Person p = new Person(1,"Middleton","Tim","Address",29,Person.MALE);

        person.put(p.getKey(), p);

        Person p2 = (Person)person.get(new Person.Key(p));

        System.out.println("Person is " + p2.getFirstname() + " " +
p2.getLastname());

    }
}
```

When you run the `KeyTester` class you should see output similar to the following:

```
Person is Tim Middleton
Process exited with exit code 0.
```

4. Create a new class called `KeyTester2` with a `main` method and add some new `Person` objects.

Figure 5–20 Creating the KeyTester2 Class



Hint: You can use the code in Example 5–12 to add entries:

Example 5–12 Code to Add Entries to KeyTester2

```
...
LinkedList peopleList = new LinkedList();
// build the list
peopleList.add( new Person(1,"Flinstone","Fred","Address",29,Person.MALE));
peopleList.add( new Person(2,"Flinstone","Wilma","Address",29,Person.FEMALE)) ;
peopleList.add( new Person(3,"Rubble","Barney","Address",44,Person.MALE));
peopleList.add( new Person(4,"Rubble","Betty","Address",44,Person.FEMALE));
 for (java.util.Iterator iterator = peopleList.iterator();
iterator.hasNext();) {
Person p = (Person)iterator.next();
person.put(p.getKey(), p); }
...
```

a. Now that you have added the entries, how do you check whether your data affinity has worked? There are several ways of doing this:

—You can use an `EntryProcessor` to do a `System.out.println` on all the `Person` entries.

—You can use the `CacheService.getKeyOwner()` method to find out who owns the entry. The `getKeyOwner()` returns the member that owns the key. Note that the entry does not have to exist. For example, you can find out which member will own a key before you create it. You can use the following:

```
Member m = ((DistributedCacheService)namedCache.getCacheService()).get
KeyOwner(pp.getKey());
```

b. Add the code to get the member that owns each of the keys. After you get the member, you can use various methods to get information about the member, such as first name, last name and who owns it.

Example 5–13 illustrates a possible implementation of `KeyTester2`.

Example 5–13 Sample Code for an Alternate KeyTester Class

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.DistributedCacheService;
import com.tangosol.net.Member;
import com.tangosol.net.NamedCache;

import java.util.LinkedList;

public class KeyTester2 {
    public KeyTester2() {
    }

    public static void main(String[] args) {

        NamedCache person = CacheFactory.getCache("person");
        LinkedList peopleList = new LinkedList();

        // build the list
        peopleList.add( new
Person(1,"Flinstone","Fred","Address",29,Person.MALE));
        peopleList.add( new
Person(2,"Flinstone","Wilma","Address",29,Person.FEMALE));
        peopleList.add( new Person(3,"Rubble","Barney","Address",44,Person.MALE));
        peopleList.add( new
Person(4,"Rubble","Betty","Address",44,Person.FEMALE));
        peopleList.add( new Person(5,"Rubble","Dino","Address",44,Person.FEMALE));

        for (java.util.Iterator iterator = peopleList.iterator();
iterator.hasNext();) {

                Person p = (Person)iterator.next();
                person.put(p.getKey(), p);

                Member m =
((DistributedCacheService)person.getCacheService()).getKeyOwner(p.getKey());
                System.out.println("Person " + p.getFirstname() + " " +
p.getLastname() + " is owned by member " + m.getId());
```

```
        }


    }
}
```

c. Run `KeyTester2` with one cache server running. You should get an output similar to the following:

```
Person Fred Flinstone is owned by member 1
Person Wilma Flinstone is owned by member 1
Person Barney Rubble is owned by member 1
Person Betty Rubble is owned by member 1
Person Dino Rubble is owned by member 1
```

d. Start up a second cache server and rerun `KeyTester2`. What happens?

You see that the same members own all the people with the same last name. If you still see all the people in the same member, add another cache server (and optionally more person objects). This should distribute the data, and you should see something like the following:

```
Person Fred Flinstone is owned by member 3
Person Wilma Flinstone is owned by member 3
Person Barney Rubble is owned by member 1
Person Betty Rubble is owned by member 1
Person Dino Rubble is owned by member 1
Process exited with exit code 0.
```

6

In-Place Processing of Data

In this chapter, you learn how `EntryProcessors` can be used to modify and perform processing on entries in the Coherence cache. This chapter contains the following sections:

Introduction

Until now, to perform actions on the entries in a cache, you used the put and get operations. If you want to control concurrency to the data, you have the option of locking and unlocking keys. However, there is a better way to perform operations on data that ensure consistent behavior when concurrent data access is required. `EntryProcessors` (`com.tangosol.util.InvocableMap.EntryProcessor`) are agents that perform processing against entries, and will process entries directly where the data is being held. The sort of processing you perform may change the data, for example create, update, remove data, or may only perform calculations on the data. `EntryProcessors` that work against the same key are logically queued. This means that you can achieve lock-free (high performance) processing. In `Lab5`, you used an `EntryAggregator` that is a type of `EntryProcessor` to aggregate data across the grid. The `com.tangosol.util.InvocableMap` interface (which the `NamedCache` implements) has the following methods for operating on data:

• `Object invoke(Object oKey, InvocableMap.EntryProcessor processor)`—Invokes the passed `EntryProcessor` against an individual object and returns the result of the invocation

• `Map invokeAll(Collection keys, InvocableMap.EntryProcessor processor)`—Invokes the `EntryProcessor` against the collection of keys and returns the result for each invocation

• `Map invokeAll(Filter filter, InvocableMap.EntryProcessor processor)`—Invokes the `EntryProcessor` against the entries that match the filter and returns the result for each invocation

To create an entry process, you can extend `com.tangosol.util.processes.AbstractProcessor` and implement the `process()` method. For example, the following code creates an `EntryProcessor` to increase the salary of all employees by 10%:

```
class RaiseSalary extents AbstractProcessor {
...
public Object process (Entry entry) {
Employee emp = (Employee)entry.getValue();
```

```
emp.setSalary(emp.getSalary() * 1.10);
entry.setValue(emp);
return null;
}
```

To invoke the `RaiseSalary` class, you perform the following:

```
empCache.invokeAll(AlwaysFilter.INSTANCE, new RaiseSalary());
```

Modifying and Processing Data Entries

This section describes how to use `EntryProcessors` to modify and perform processing on entries in the Coherence cache.

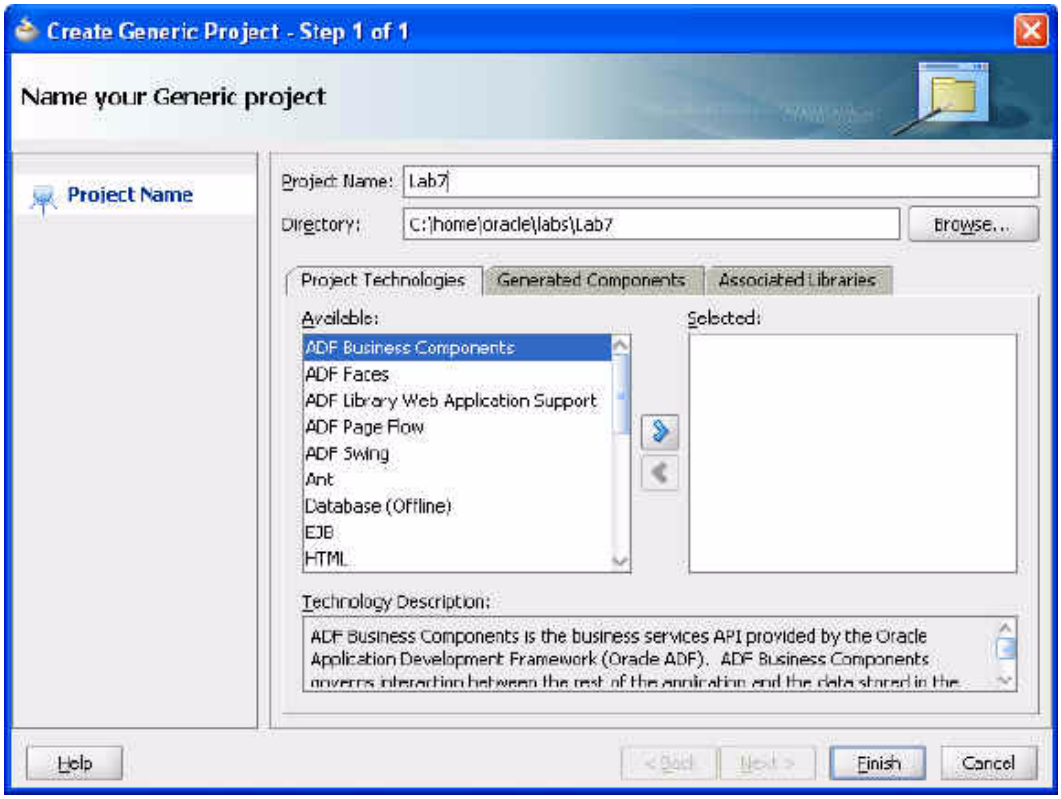
In this exercise you will use JDeveloper, to create the following:

- An `Employee` class to hold your employees
- A `RaiseSalary` class to be invoked on all entries
- An `InvokeTest` class to insert employees and run `RaiseSalary` on
- A class that shows where data is held in a cluster

To create Java classes to modify and perform processing on data entries:

1. Create a new class for the `Employee` object.
   a. Create a new project called `Lab7`.

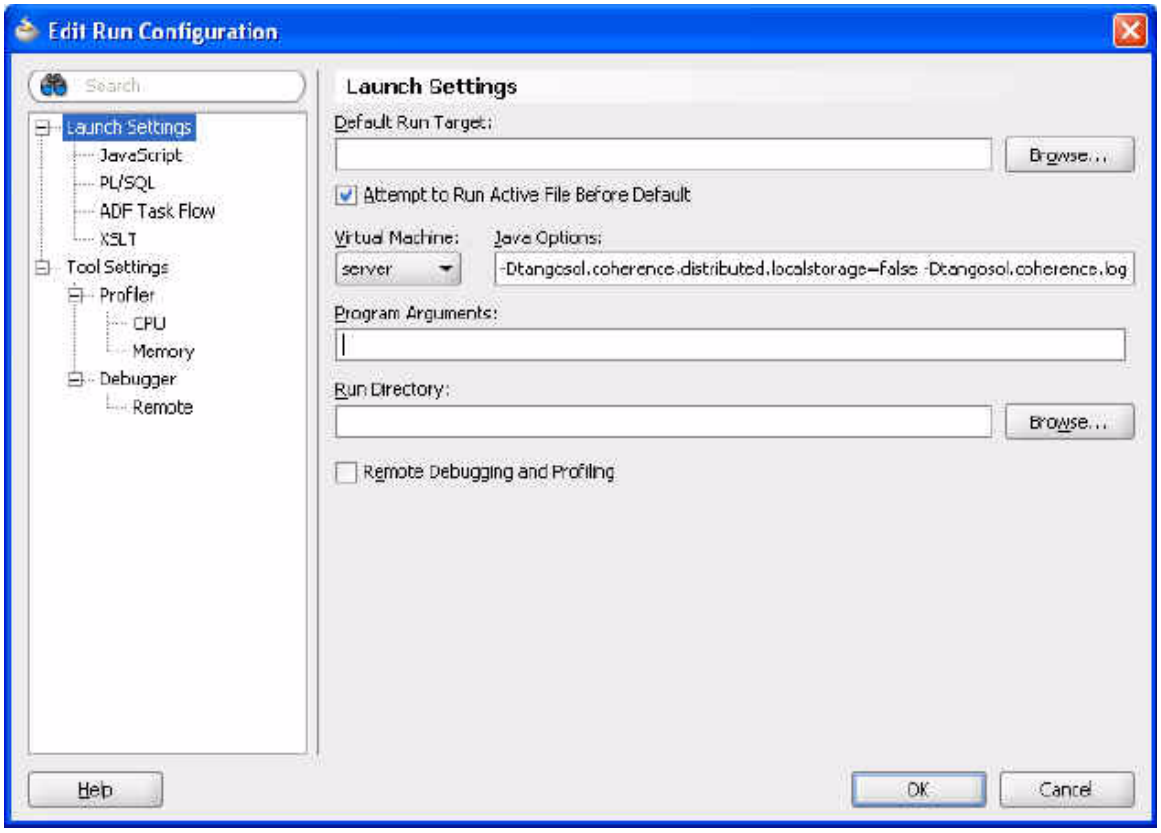
Figure 6–1 Creating a New Project



   b. Make sure that you change the default local storage properties to the following values.

```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

Figure 6–2 Edit the Runtime Configuration



c. Create an `Employee` class that implements `PortableObject`. Add the following attributes to the class:

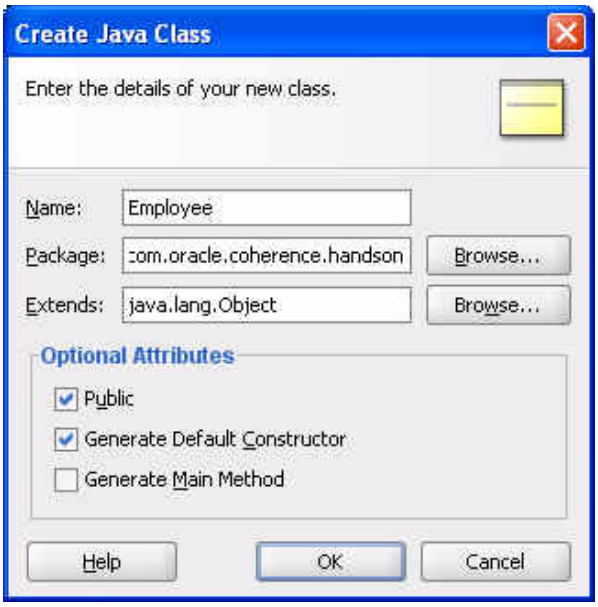
—`private int empId`

—`private String lastname`

—`private String firstname`

—`private double salary`

Figure 6–3 Creating the Employee Class



d. Ensure that you implement all of the required methods. (Use the **Source** menu for this).

Example 6–1 illustrates a possible implementation of the `Employee` class.

Example 6–1 Sample Employee Class

```
package com.oracle.coherence.handson;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import java.math.BigDecimal;

public class Employee implements PortableObject {
    private int empId;
    private String lastname;
    private String firstname;
    private double salary;


    public Employee() {
    }

    public Employee(int empId1, String lastname1, String firstname1,
                    double salary1) {
        super();
        this.empId = empId1;
        this.lastname = lastname1;
        this.firstname = firstname1;
        this.salary = salary1;
    }
```

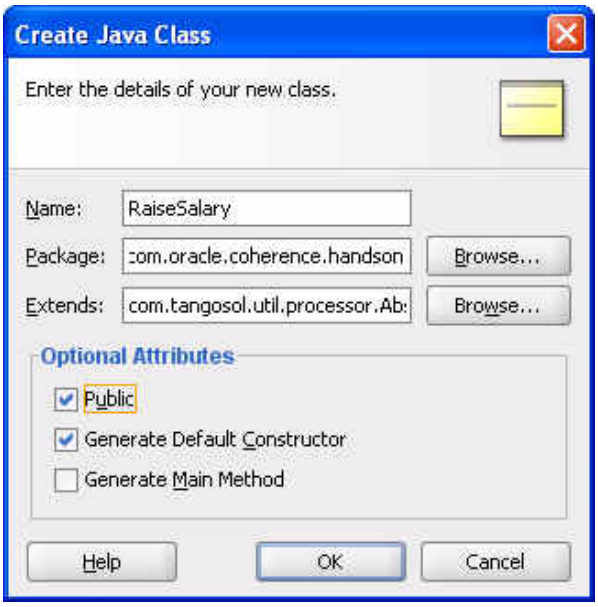
```
    public void setEmpId(int param) {
        this.empId = param;
    }

    public int getEmpId() {
        return empId;
    }

    public void setLastname(String param) {
        this.lastname = param;
    }

    public String getLastname() {
        return lastname;
    }

    public void setFirstname(String param) {
        this.firstname = param;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setSalary(double param) {
        this.salary = param;
    }

    public double getSalary() {
        return salary;
    }

    @Override
    public boolean equals(Object object) {
        if (this == object) {
            return true;
        }
        if (!(object instanceof Employee)) {
            return false;
        }
        final Employee other = (Employee)object;
        if (empId != other.empId) {
            return false;
        }
        if (!(lastname == null ? other.lastname == null :
lastname.equals(other.lastname))) {
            return false;
        }
        if (!(firstname == null ? other.firstname == null :
firstname.equals(other.firstname))) {
            return false;
        }
        if (Double.compare(salary, other.salary) != 0) {
            return false;
        }
        return true;
    }

    @Override
```

```
    public int hashCode() {
        final int PRIME = 37;
        int result = 1;
        result = PRIME * result + ((lastname == null) ? 0 : lastname.hashCode());
        result = PRIME * result + ((firstname == null) ? 0 :
firstname.hashCode());
        long temp = Double.doubleToLongBits(salary);
        result = PRIME * result + (int) (temp ^ (temp >>> 32));
        return result;
    }

    public void readExternal(PofReader dataInput) throws IOException {
        this.empId = dataInput.readInt(0);
        this.lastname = dataInput.readString(1);
        this.firstname = dataInput.readString(2);
        this.salary  = dataInput.readBigDecimal(3).doubleValue();

    }

    public void writeExternal(PofWriter dataOutput) throws IOException {
        dataOutput.writeInt(0, this.empId);
        dataOutput.writeString(1, this.lastname);
        dataOutput.writeString(2, this.firstname);
        dataOutput.writeBigDecimal(3, new BigDecimal(this.salary));

    }
}
```

2. Create a class to increase the salary of employees by 10%.

   a. Create a class called `RaiseSalary` that extends `AbstractProcessor`.

Figure 6–4 Create Raise Salary Class



   b. Implement the `process()` method to raise the salary of an employee by 10%. Example 6–2 illustrates a possible solution.

Example 6–2 Sample Raise Salary Class

```
package com.oracle.coherence.handson;

import com.tangosol.util.processor.AbstractProcessor;
import com.tangosol.util.InvocableMap.Entry;

import java.util.Map;

public class RaiseSalary extends AbstractProcessor {
    public RaiseSalary() {
    }

    public Object process(Entry entry ) {
        Employee emp = (Employee)entry.getValue();
        emp.setSalary(emp.getSalary() * 1.10);
        entry.setValue(emp);
        return null;
    }
}
```

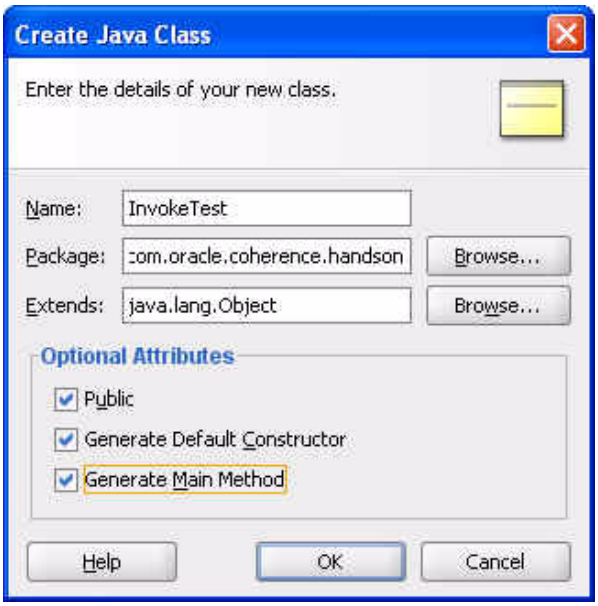
3. Create a class to test whether the `RaiseSalary` class works.
   a. Create a class called `InvokeTest` that contains a `main` method and performs the following:

      —Creates several `Employee` objects and puts them in the `employees` cache.

      —Invokes the `RaiseSalary` method on them.

      —Prints out the new salaries to confirm the changes that have been made.

Figure 6–5 Creating the Invoke Test Class



Example 6–3 illustrates a possible solution.

Example 6–3 Sample Program to Test RaiseSalary Class

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
```

```
import com.tangosol.net.NamedCache;
import com.tangosol.util.filter.AlwaysFilter;

public class InvokeTest {
    public InvokeTest() {
    }

    public static void main(String[] args) {

        NamedCache empCache = CacheFactory.getCache("employees");

        Employee e1 = new Employee(1,"Middleton","Tim",5000);
        empCache.put(e1.getEmpId(), e1);

        Employee e2 = new Employee(2,"Jones","Chris",10000);
        empCache.put(e2.getEmpId(), e2);

        empCache.invokeAll(AlwaysFilter.INSTANCE, new RaiseSalary());

        e1 = (Employee)empCache.get(e1.getEmpId());
        e2 = (Employee)empCache.get(e2.getEmpId());

        System.out.println("Salary for emp 1 is now: " + e1.getSalary());
        System.out.println("Salary for emp 2 is now: " + e2.getSalary());

    }
}
```

b. When you first try to invoke `InvokeTest`, you get a "`class not found`" error. Why does this happen and how do you fix it?

Figure 6–6 Class Not Found Error

When you invoke an `EntryProcessor`, the request is processed on the storage-enabled member that contains the entry.

The Coherence cache server does not yet know about the `RaiseSalary` or `Employee` objects that you have created. You must add these to the `CLASSPATH` of the Coherence cache server.

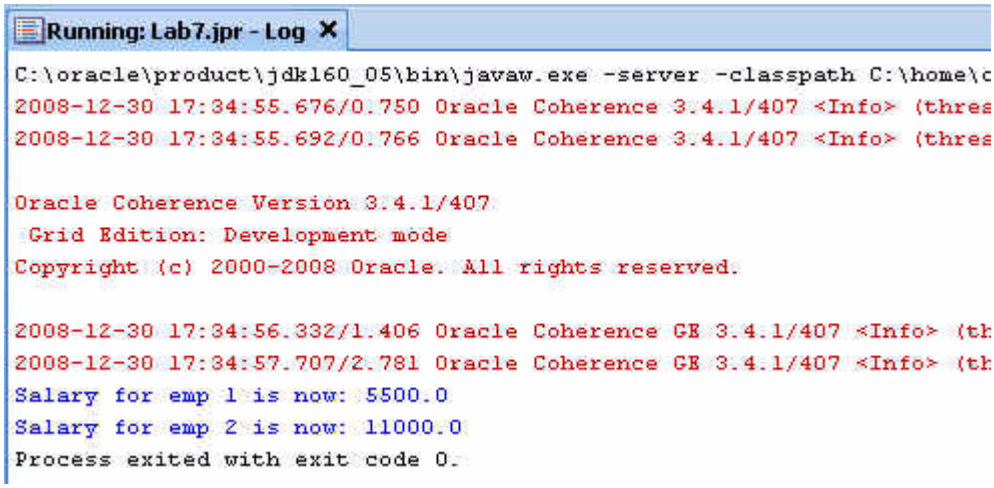
Edit the `cache-server.cmd` file. Modify the `-cp` entry on the line beginning with `"%java_exec%" -server...` to set the `CLASSPATH` environment variable to include `/home/oracle/labs/Lab7/classes`. The `"%java_exec%"` line should look similar to the following:

```
"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar;C:\home\oracle\labs\Lab4\classes;C:\home\oracle\lab
s\Lab7\classes;C:\home\oracle\labs" com.tangosol.net.DefaultCacheServer %1
```

c. Restart the cache server.

d. Run `InvokeTest` again to ensure that `RaiseSalary` class works.

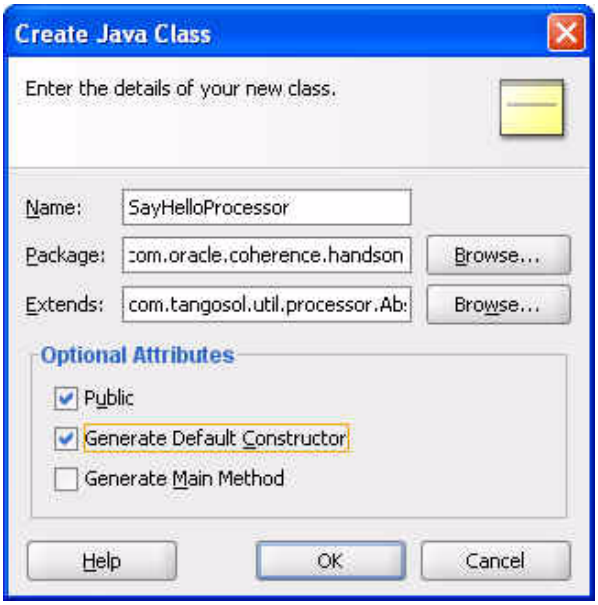
Figure 6–7 A Successful Run of the Invoke Test Class



4. Create a class to show where data is held within a cluster.

a. Create a class called `SayHelloProcessor`, which extends `AbstractProcessor`.

Figure 6–8 Creating the Say Hello Processor Class



b. Implement the `process()` method to say `"hello"`.

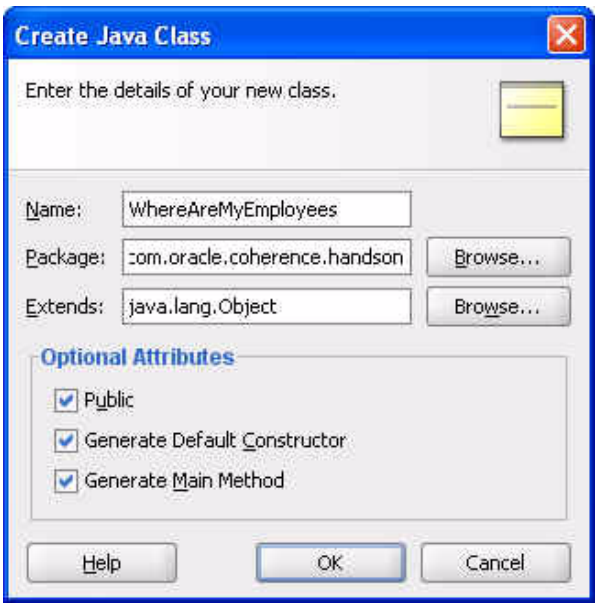
Example 6–4 illustrates a possible solution.

Example 6–4 Sample Program to Show Where Data is Held

```
package com.oracle.coherence.handson;

import com.tangosol.util.processor.AbstractProcessor;
import com.tangosol.util.InvocableMap.Entry;

public class SayHelloProcessor extends AbstractProcessor {
    public SayHelloProcessor() {

    }

    public Object process(Entry entry ) {
        Employee emp = (Employee)entry.getValue();
        System.out.println("\nHello from " + emp.getFirstname() + " " +
emp.getLastname() + "\n");
        return null;
    }
}
```

c. Create a new class called `WhereAreMyEmployees`. Ensure that this class has a `main` method.

Figure 6–9 Creating the Where Are My Employees Class



Example 6–5 illustrates a possible solution.

Example 6–5 Sample Program to Trace Member Location of Employees

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.filter.AlwaysFilter;
```

```
public class WhereAreMyEmployees {
    public WhereAreMyEmployees() {
    }

    public static void main(String[] args) {

        NamedCache empCache = CacheFactory.getCache("employees");

        empCache.invokeAll(AlwaysFilter.INSTANCE, new SayHelloProcessor());
    }
}
```

d. Invoke the `WhereAreMyEmployees` on all the entries in the cache.

Example 6–6 Output of WhereAreMyEmployees Class with One Cache Server Console

```
...
2008-12-30 18:01:04.411/25.985 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluste
r, member=1): Member(Id=3, Timestamp=2008-12-30 18:01:04.231, Address=130.35.99.
248:8089, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proce
ss:5988, Role=OracleWhereAreMyEmployees) joined Cluster with senior member 1

Hello from Tim Middleton

Hello from Chris Jones

2008-12-30 18:01:04.848/26.422 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluste
r, member=1): Member 3 joined Service DistributedCache with senior member 1
2008-12-30 18:01:04.848/26.422 Oracle Coherence GE 3.4.1/407 <D5> (thread=Distri
butedCache, member=1): Service DistributedCache: sending ServiceConfigSync conta
ining 259 entries to Member 3
2008-12-30 18:01:04.942/26.516 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluste
r, member=1): TcpRing: disconnected from member 2 due to the peer departure
```

e. Add some more `Employees` and a second cache server. Run the `WhereAreMyEmployees` class again. Observe the messages on the cache server console.

Example 6–7 illustrates the response from one cache server console.

Example 6–7 Response on Cache Server Console 1

```
2008-12-30 18:04:34.582/236.156 Oracle Coherence GE 3.4.1/407 <D5> (thread=Clust
er, member=1): Member(Id=3, Timestamp=2008-12-30 18:04:34.433, Address=130.35.99
.248:8090, MachineId=49912, Location=site:us.oracle.com,machine:tpfaeffl-pc,proc
ess:3560, Role=OracleWhereAreMyEmployees) joined Cluster with senior member 1

Hello from Chris Jones

2008-12-30 18:04:35.239/236.813 Oracle Coherence GE 3.4.1/407 <D5> (thread=Clust
er, member=1): Member 3 joined Service DistributedCache with senior member 1
2008-12-30 18:04:35.254/236.828 Oracle Coherence GE 3.4.1/407 <D5> (thread=Distr
ibutedCache, member=1): Service DistributedCache: sending ServiceConfigSync cont
aining 259 entries to Member 3
```

Example 6–8 illustrates the response from the second cache server console.

Example 6–8 Response on Cache Server Console 2

```
2008-12-30 18:04:35.239/27.047 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluste
r, member=2): Member 3 joined Service DistributedCache with senior member 1
```

```
Hello from Tim Middleton

2008-12-30 18:05:07.707/59.515 Oracle Coherence GE 3.4.1/407 <D5> (thread=Cluste
r, member=2): MemberLeft request for Member 3 received from Member(Id=1, Timesta
mp=2008-12-30 18:00:40.286, Address=130.35.99.248:8088, MachineId=49912, Locatio
n=site:us.oracle.com,machine:tpfaeffl-pc,process:4460, Role=CoherenceServer)
```

7

Using JPA with Coherence

In this exercise, you learn how to use Java Persistence API (JPA) to perform object-relational mapping. This chapter contains the following sections:

- Introduction
- Mapping Relational Data to Java Objects with JPA

This exercise assumes that you have a working version of the Oracle Database 10g Express Edition (also known as the OracleXE database) installed on your system. If you do not already have database, you can download one here:

`http://www.oracle.com/technology/software/products/database/xe/index.html`

Introduction

A major enhancement in EJB technology is the addition of JPA, which simplifies the entity persistence model and adds capabilities that were not in the EJB 2.1 technology.

JPA deals with the way relational data is mapped to Java objects ("persistent entities"), the way that these objects are stored in a relational database so that they can be accessed at a later time, and the continued existence of an entity's state even after the application that uses it ends. In addition to simplifying the entity persistence model, the JPA standardizes object-relational mapping.

To determine how data is stored within a Coherence cluster, a backing map is used. By default, Coherence uses a memory-based backing map. To persist data, there are several backing map implementations.

You use the JPA implementation within this lesson. This implementation provides Object Relational Mapping (ORM) from the Java world to the database world, allowing you to use the standard Coherence get or put, and have the Coherence calls translated into database calls using JPA and EclipseLink.

Mapping Relational Data to Java Objects with JPA

In this exercise, you will use JDeveloper to perform the following:

- Create a connection to the `HR` schema in the OracleXE database.
- Automatically generate JPA objects for the `EMPLOYEES` table.
- Modify `cache-server.cmd` to point to the sample JPA `cache-config.xml`.

To use the Java Persistence API (JPA) to demonstrate data mapping with Coherence:

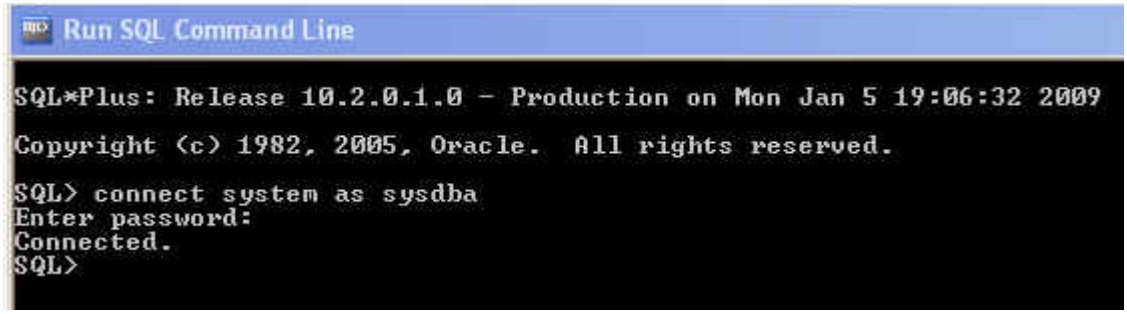
1. Unlock the `HR` account in your pre-installed OracleXE database.

It is assumed that you have the OracleXE database installed on your machine and can access the HR schema. To unlock the HR account, perform the following steps:

a. Navigate to **Start > All Programs> Oracle Database 10g Express Edition > Run SQL Command Line**.

b. Enter `connect system as sysdba`, and then enter `welcome1` when prompted for the password. (Note this exercise assumes that your user name is `system` and password is `welcome1`).

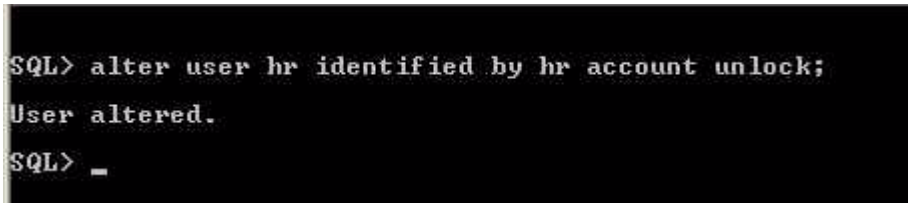
Figure 7–1 Connecting to the Database



c. Enter the command to unlock the account:

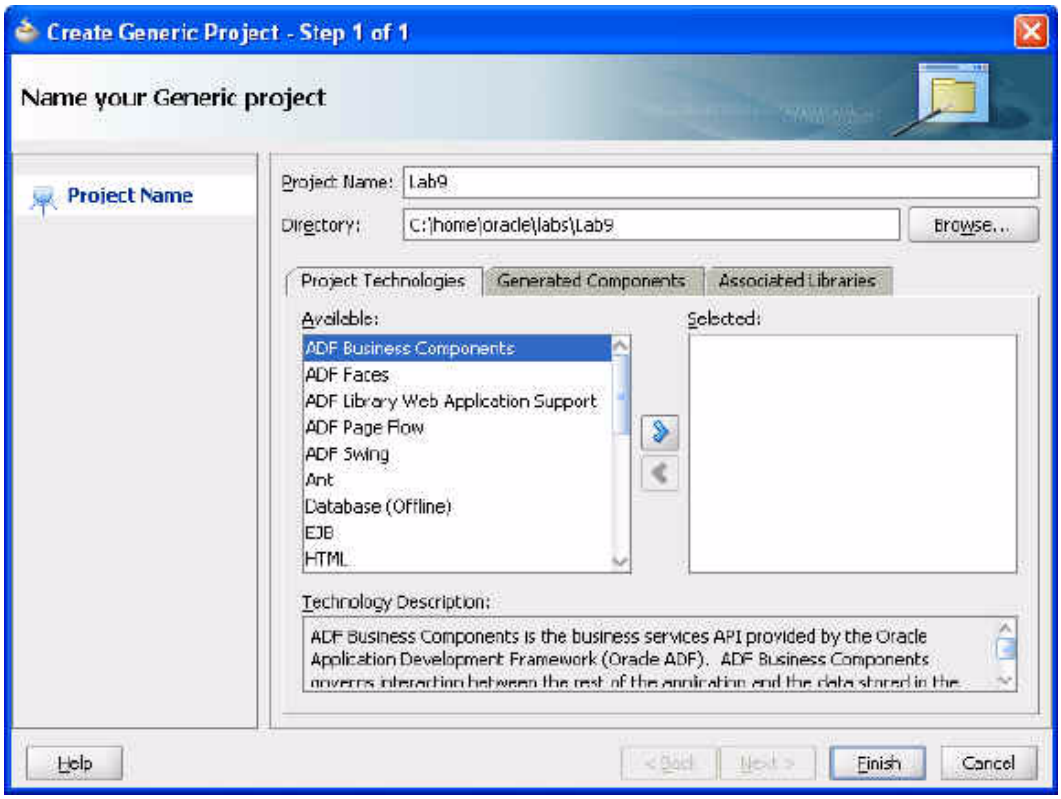
```
alter user hr identified by hr account unlock;
```

Figure 7–2 Unlocking the Database Account



2. Create a new project called `Lab9`.

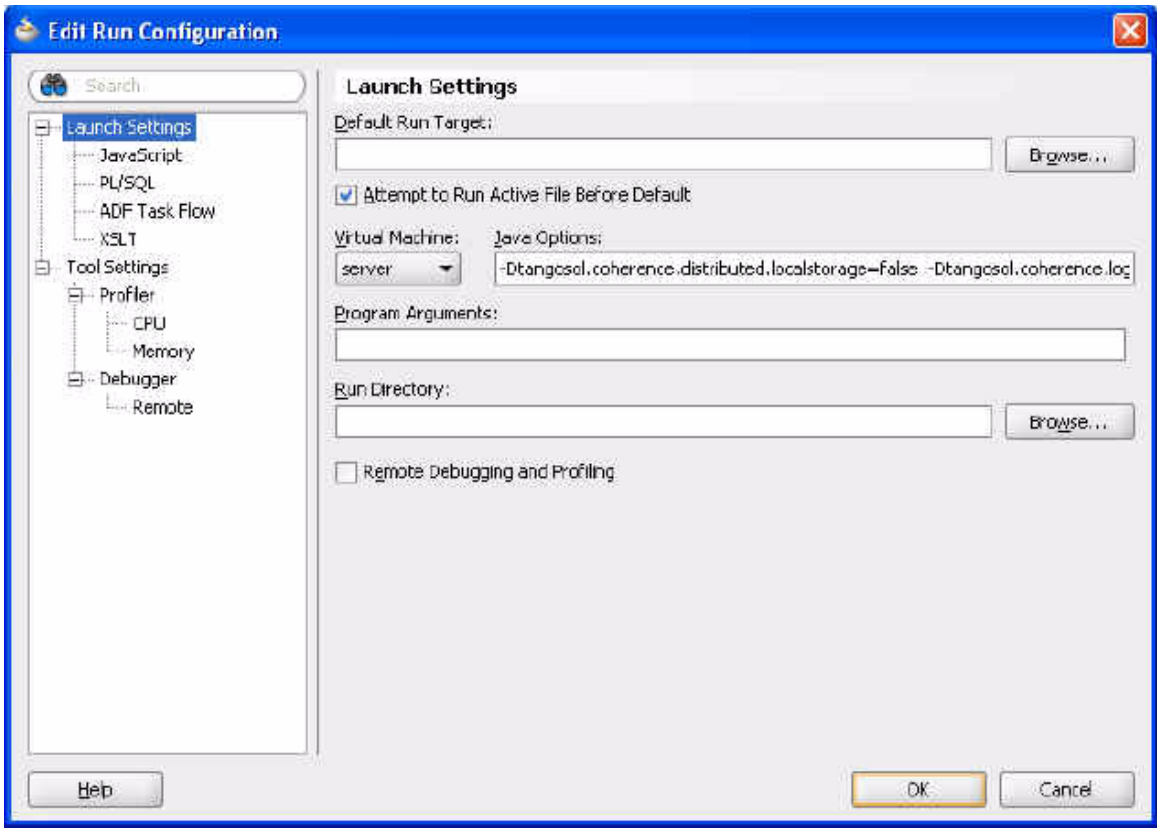
Figure 7–3 Creating a New Project



a. Set the default Java Options to the appropriate log level and to disable local storage.

```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

Figure 7–4 Setting Java Options



3. Create a new database connection to the HR schema.
   a. In the **Application Resources** section of the navigator, right-click **Connections**, select **New Connection**, and then **Database Connection**.
   b. Enter the details to connect to your HR schema and click OK.

      —**Connection Name**: XE_HR

      —**Connection Type**: Oracle (JDBC)

      —**Username**: hr

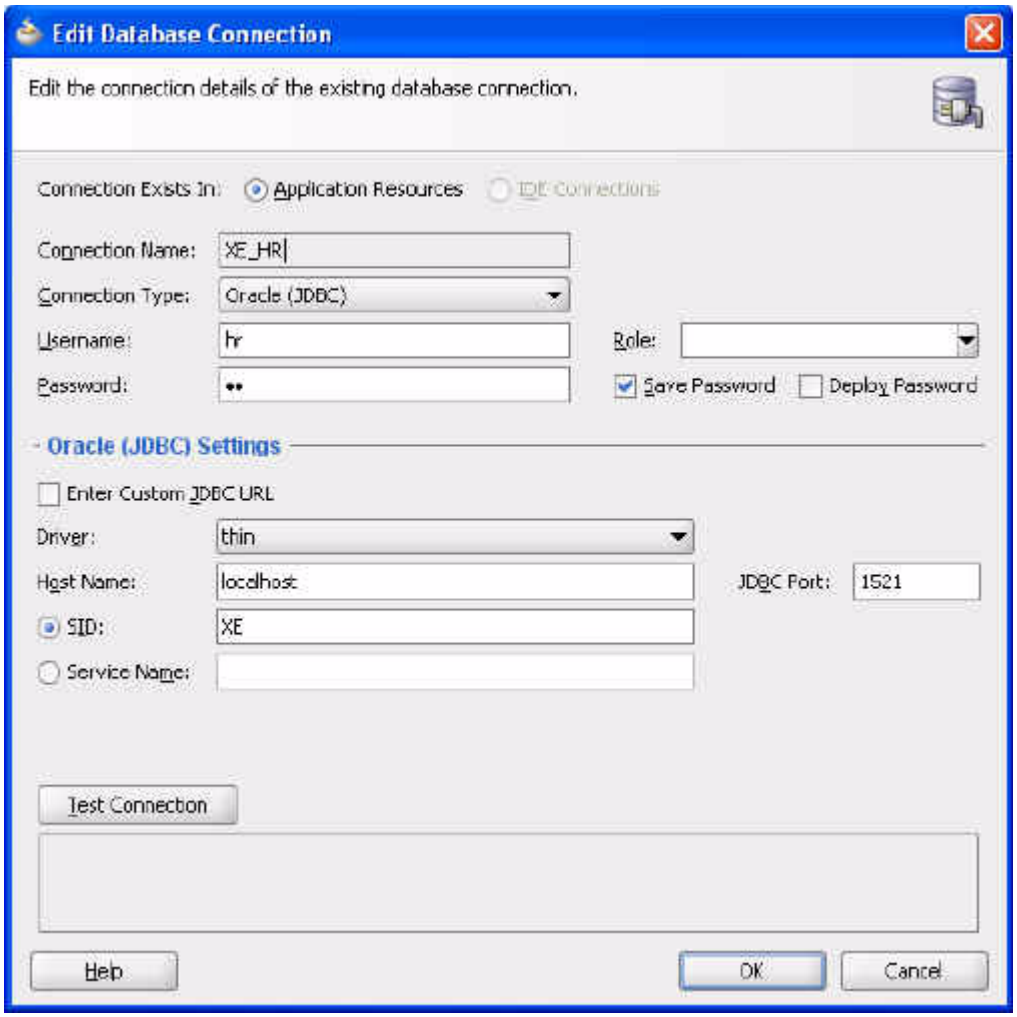
      —**Password**: hr

      —Click **Test Connection**.

      This should display "`Success`!"

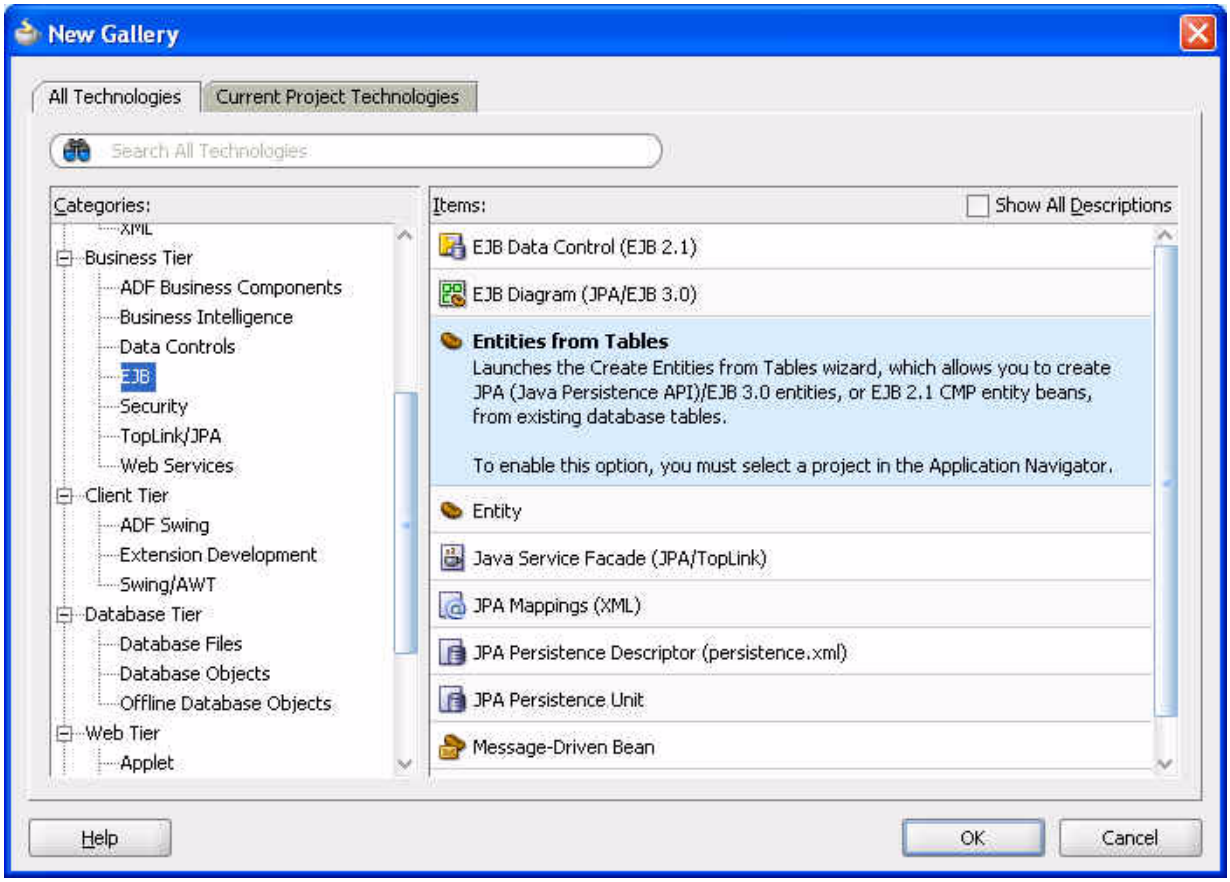
      Click **OK**.

Figure 7–5 Defining the Database Connection



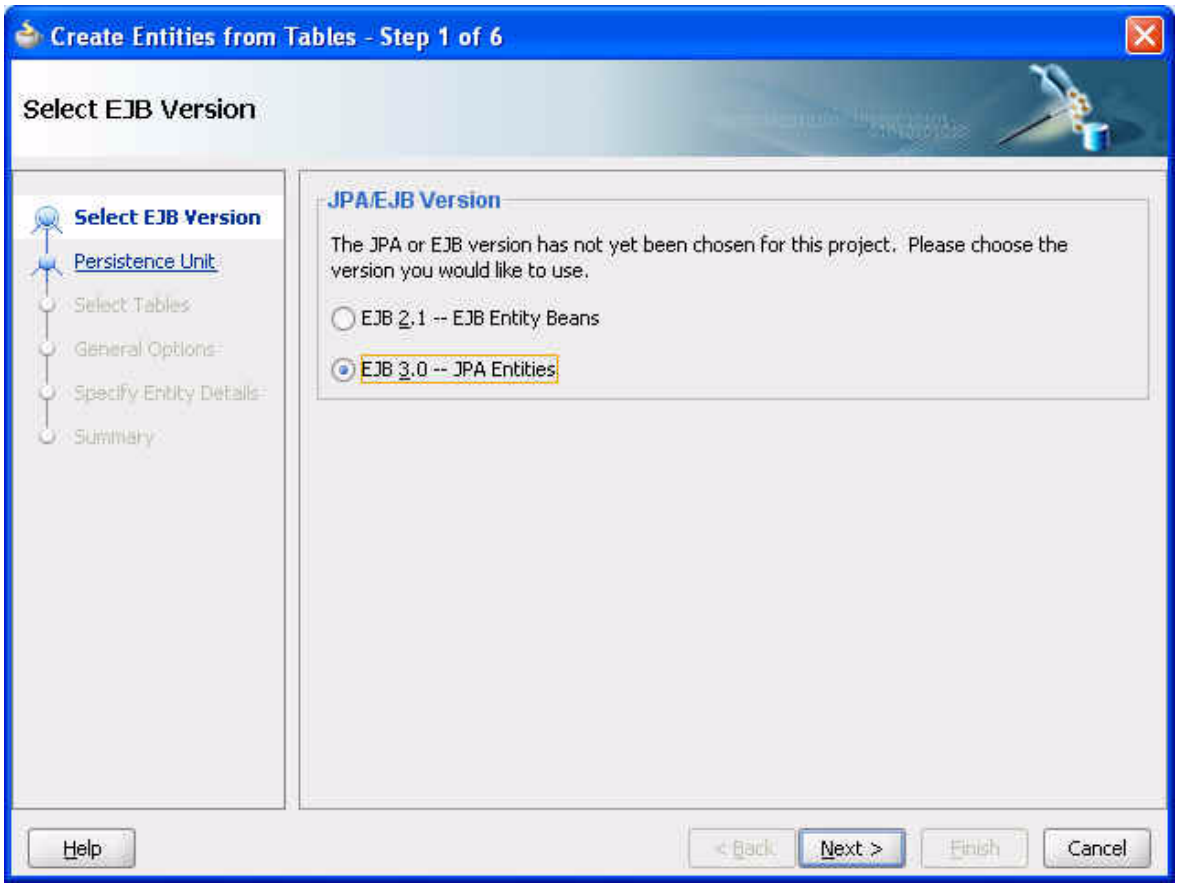
c. Right-click the `Lab9` project and select **New**. Under **Business Tier**, select **EJB**, and then select **Entities from Tables**. Click **OK**.

Figure 7–6 Creating EJB Entity Beans



d. In the **Create Entities from Tables** window, select **EJB 3.0 --JPA Entities**, and then click Next.

Figure 7–7 Specifying the EJB Version



e. Click **New** to create a new persistence unit. (Each persistence unit defines a set of classes and their mapping characteristics when persisting them.) Enter the following details and click **OK**.

Name: JPA

JTA Datasource Name: JPADS

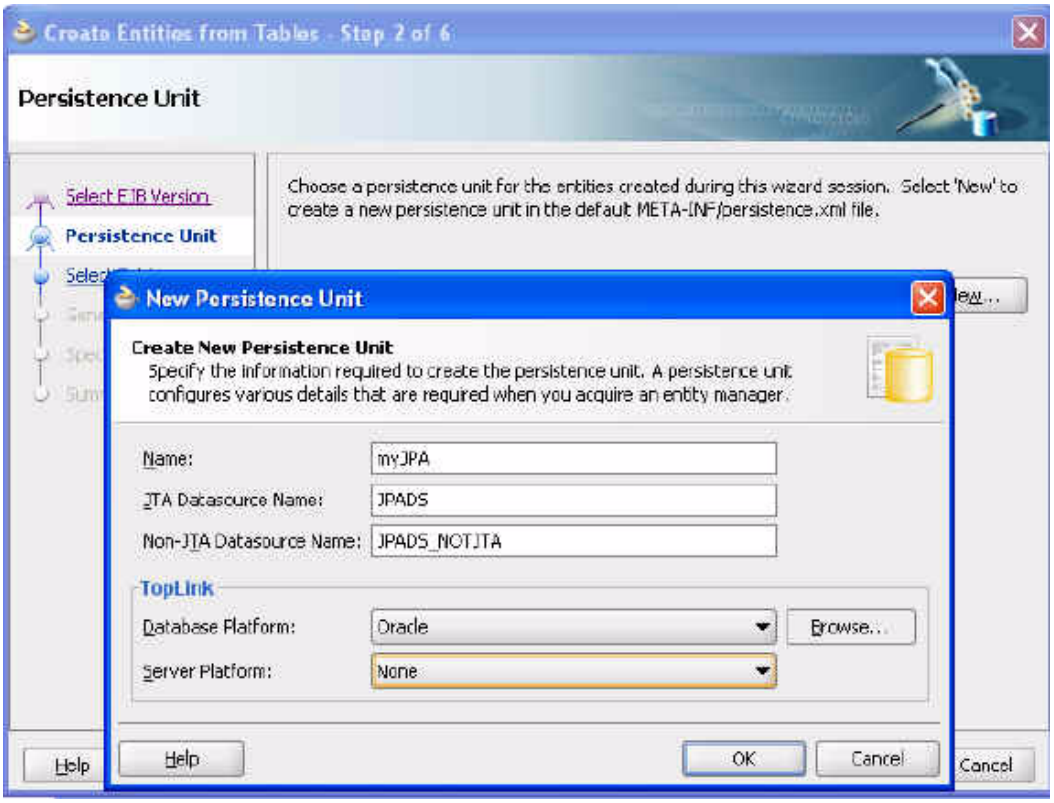
Non-JTA Datasource Name: JPADS_NOTJTA

Database Platform: Oracle Server

Platform: None

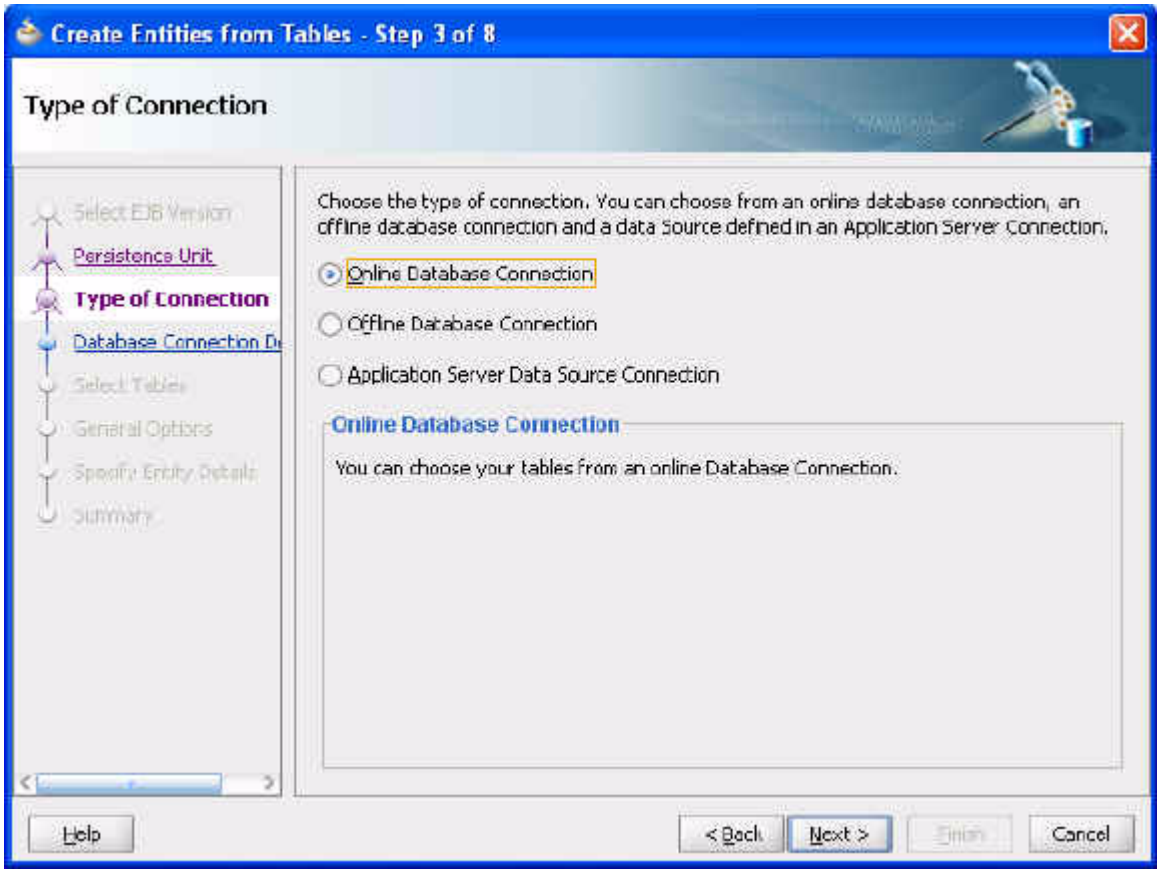
After the **Create Entities from Tables** screen returns, click **Next**.

Figure 7–8 Defining the Persistence Unit



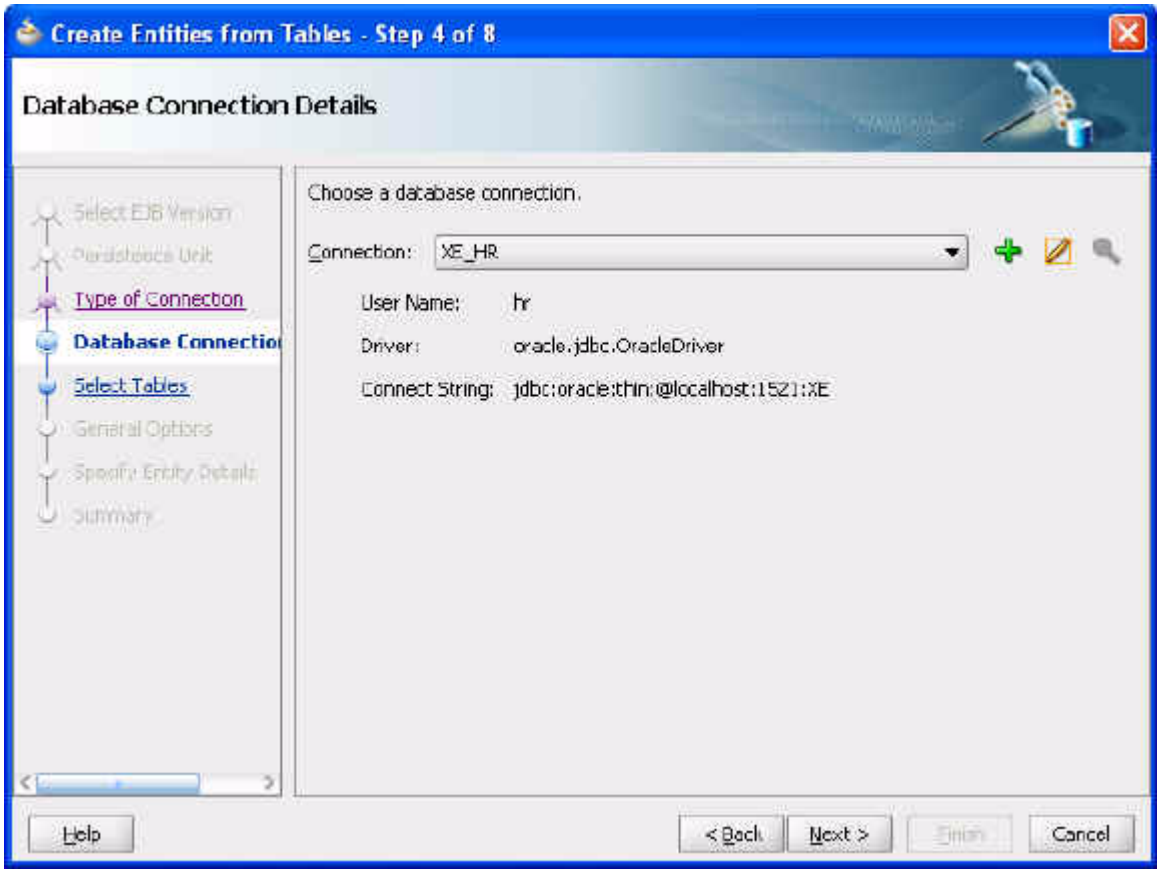
f. Select the **Online Database Connection** option and click **Next**.

Figure 7–9 Creating Entity Beans from Table Data



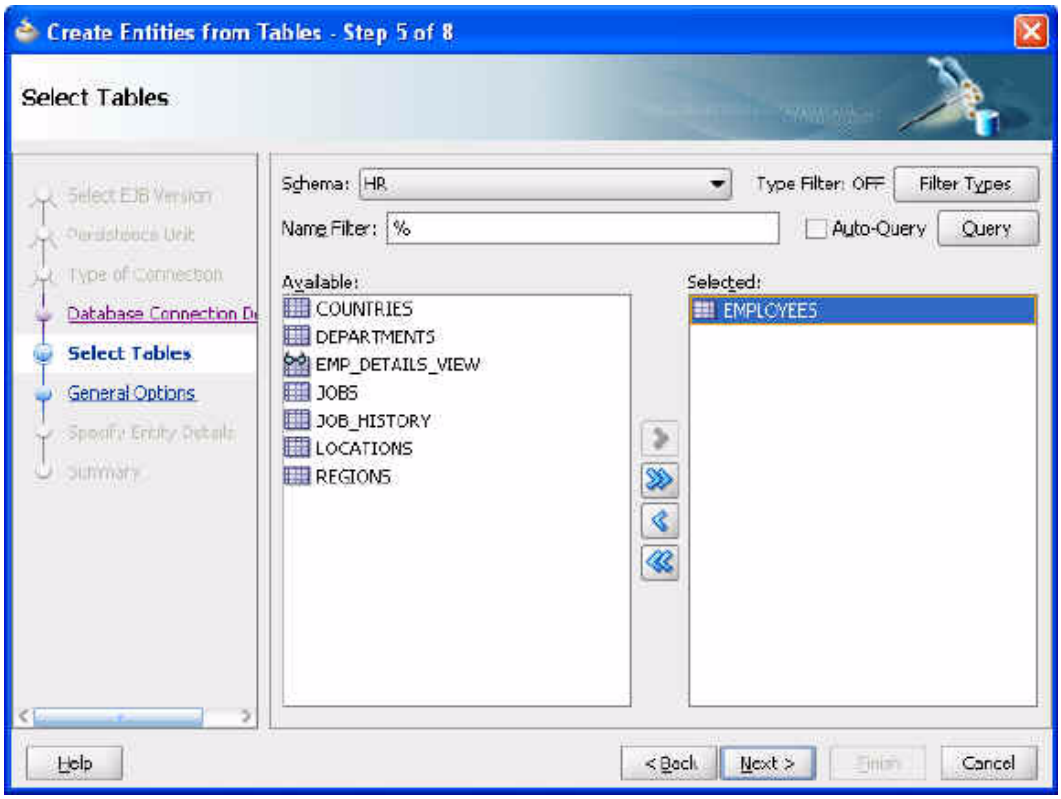
g. In the **Database Connection Details** window, click **Next**.

Figure 7–10 Choosing the Database Connection



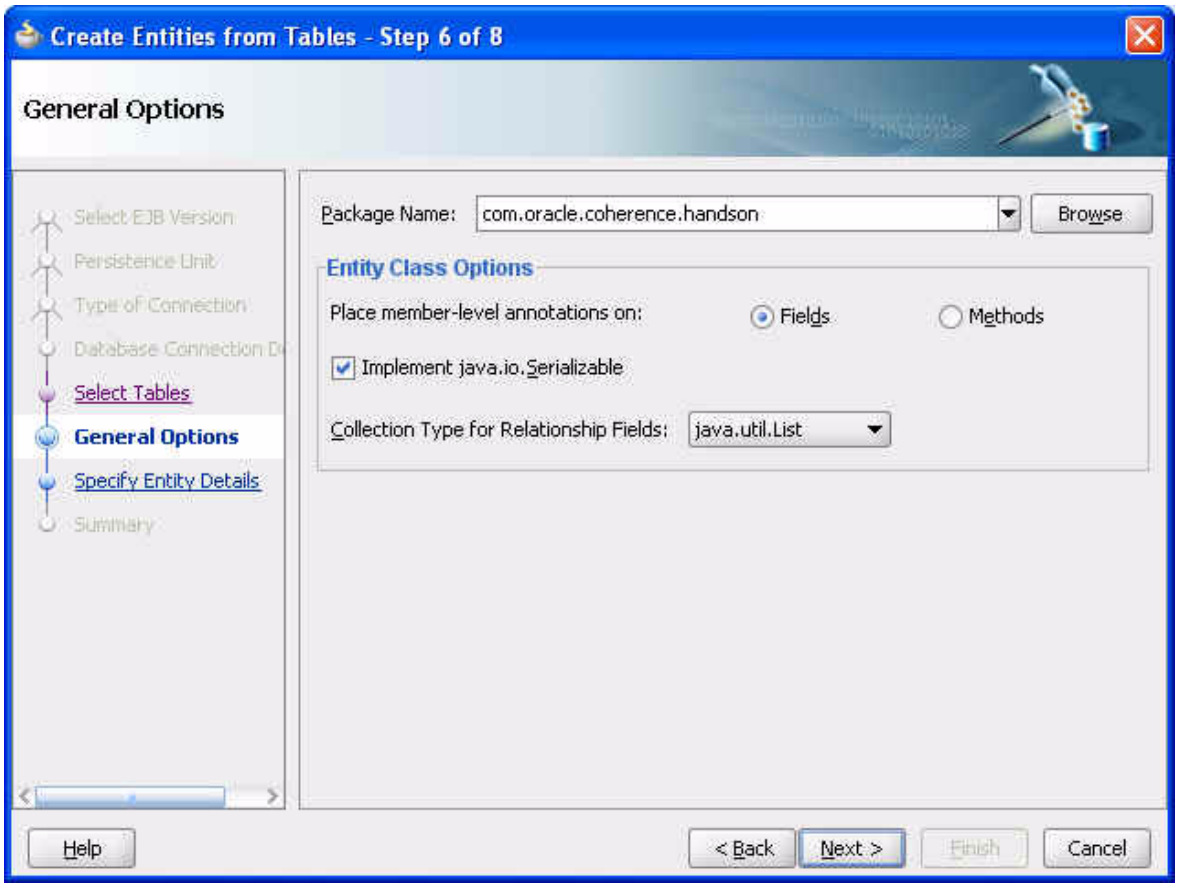
h. Query for the `EMPLOYEES` table and select it as shown in Figure 7–11. Click **Next**.

Figure 7–11 Choosing the Table Data for the Entity Bean



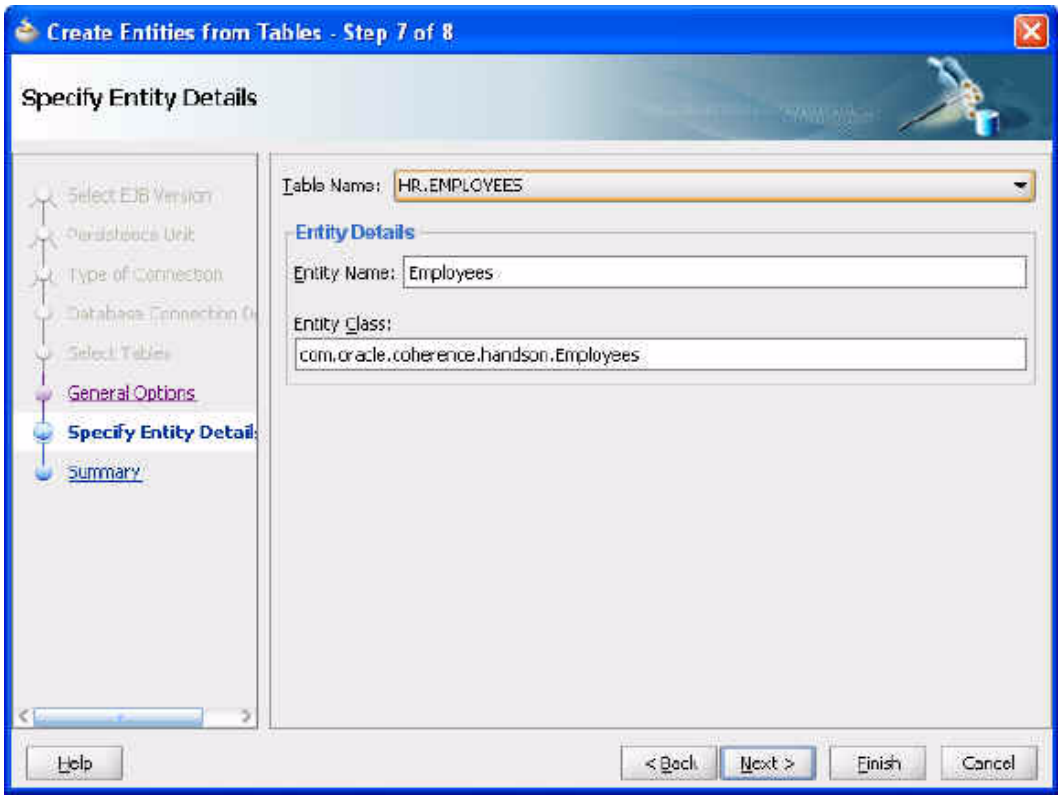
i. Retain the **General Options** as they are and click **Next**.

Figure 7–12 Choosing General Options for the Entity



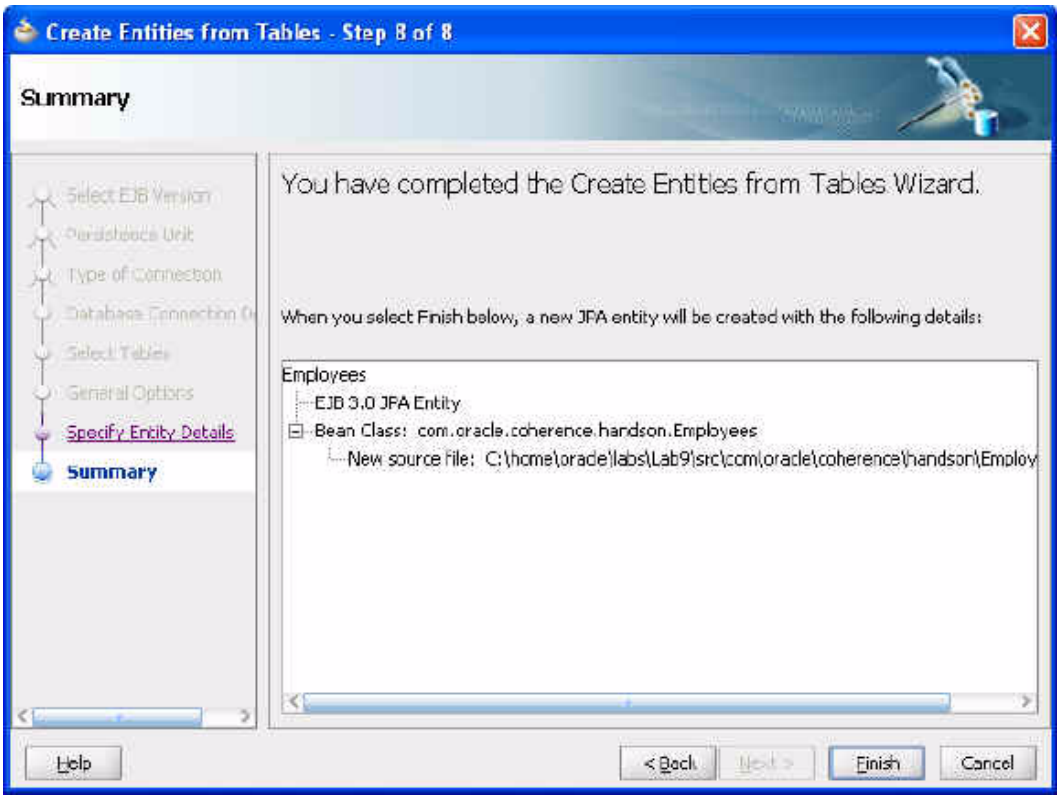
j. Retain the **Entity Class Details** as they are and click **Next**.

Figure 7–13 Specifying the Entity Details



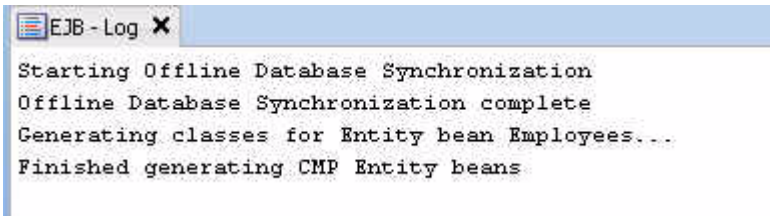
k. You should see the following in the **Summary** page.

Figure 7–14 Entity Bean Summary Page



l. Click **Finish** to complete the creation. You should now see the following in the navigator.

Figure 7–15 Generating EJB Entity Beans—the EJB Log Window



m. Replace the contents of `persistence.xml` with the code in Example 7–1 and save the file. Ensure that the connection details match your database connection details.

Example 7–1 persistance.xml File Contents

```
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance" version="1.0"
xmlns="http://java.sun.com/xml/ns/persistence">
<persistence-unit name="JPA" transaction-type="RESOURCE_LOCAL">
    <provider>
        org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>com.oracle.coherence.handson.Employees</class>
    <properties>
        <property name="eclipselink.logging.level" value="INFO"/>
        <property name="eclipselink.jdbc.driver"
value="oracle.jdbc.OracleDriver"/>
```

```
        <property name="eclipselink.jdbc.url"
value="jdbc:oracle:thin:@localhost:1521:XE"/>
        <property name="eclipselink.jdbc.password" value="hr"/>
        <property name="eclipselink.jdbc.user" value="hr"/>
    </properties>
</persistence-unit>
</persistence>
```

4. Create a cache configuration file for JPA.

   Open a text editor and create a file named `jpa-cache-config.xml`. Use the code illustrated in Example 7–2. Save the file in the `home\oracle\labs\` directory.

Example 7–2 Cache Configuration for JPA

```
<?xml version="1.0" encoding="windows-1252" ?>
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <!-- Set the name of the cache to be the entity name  -->
      <cache-name>Employees</cache-name>
      <!-- Configure this cache to use the scheme defined below  -->
      <scheme-name>jpa-distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <distributed-scheme>
      <scheme-name>jpa-distributed</scheme-name>
      <service-name>JpaDistributedCache</service-name>
      <backing-map-scheme>
        <read-write-backing-map-scheme>
          <!--
      Define the cache scheme
      -->
          <internal-cache-scheme>
            <local-scheme/>
          </internal-cache-scheme>
          <cachestore-scheme>
            <class-scheme>
              <class-name>com.tangosol.coherence.jpa.JpaCacheStore</class-name>
              <init-params>
                <!--
              This param is the entity name
              This param is the fully qualified entity class
              This param should match the value of the
              persistence unit name in persistence.xml
              -->
                  <init-param>
                    <param-type>java.lang.String</param-type>
                    <param-value>{cache-name}</param-value>
                  </init-param>
                  <init-param>
                    <param-type>java.lang.String</param-type>

<param-value>com.oracle.coherence.handson.{cache-name}</param-value>
                  </init-param>
                  <init-param>
                    <param-type>java.lang.String</param-type>
                    <param-value>JPA</param-value>
                  </init-param>
```

```
            </init-params>
          </class-scheme>
        </cachestore-scheme>
      </read-write-backing-map-scheme>
    </backing-map-scheme>
    <autostart>true</autostart>
  </distributed-scheme>
 </caching-schemes>
</cache-config>
```

5. Copy the `cache-server.cmd` file and modify the server properties.

   a. Open a terminal window. Navigate to the `/oracle/product/coherence/bin` directory and copy the `cache-server.cmd` file to `jpa-cache-server.cmd`.

   ```
   cp cache-server.cmd jpa-cache-server.cmd
   ```

   b. Edit `jpa-cache-server.cmd`. Add the following switch to `Java_OPTS`:

   ```
   -Dtangosol.coherence.cacheconfig=\home\oracle\labs\jpa-cache-config.xml
   ```

   c. Add the following `CLASSPATH` to the `-cp` argument: `C:\home\oracle\labs\Lab9\classes`

   ```
   C:\home\oracle\labs\Lab9\classes
   ```

   d. You must also add the following jar files to the `CLASSPATH`:

   — Coherence JPA libraries: `C:\oracle\product\coherence\lib\coherence-jpa.jar`

   — JDBC libraries: `C:\oracle\product\wlserver_10.3\server\lib\ojdbc5.jar`

   —`javax.persistence.*` libraries: `C:\oracle\product\modules\javax.persistence_1.0.0.0_1-0.jar`

   — EclipseLink Libraries: `C:\oracle\product\jdeveloper\modules\oracle.toplink_11.1.1\eclipselink.jar`

   ```
   ...
   C:\oracle\product\coherence\lib\coherence-jpa.jar;
   C:\oracle\product\wlserver_10.3\server\lib\ojdbc5.jar;
   C:\oracle\product\jdeveloper\modules\oracle.toplink_11.1.1\toplink.jar;
   ...
   ```

   Example 7–3 illustrates a modified `jpa-cache-server.cmd` file:

Example 7–3 Modified jpa-cache-server.cmd File

```
@echo off
@
@rem This will start a cache server
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:/oracle/product/coherence
```

```
@rem specify the JVM heap size
set memory=512m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch

set java_opts="-Xms%memory% -Xmx%memory%
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\jpa-cache-config.xml"

"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar;C:\home\oracle\labs\Lab9\classes;C:\oracle\product\coheren
ce\lib\coherence-jpa.jar;C:\oracle\product\wlserver_
10.3\server\lib\ojdbc5.jar;C:\oracle\product\jdeveloper\modules\oracle.toplink_
11.1.1\eclipselink.jar;C:\oracle\product\modules\javax.persistence_1.0.0.0_
1-0.jar" com.tangosol.net.DefaultCacheServer %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\cache-server.cmd
goto exit

:exit
endlocal
@echo on
```

e. After modifying `jpa-cache-server.cmd`, make sure that all other cache servers are stopped. Run `jpa-cache-server.cmd`.

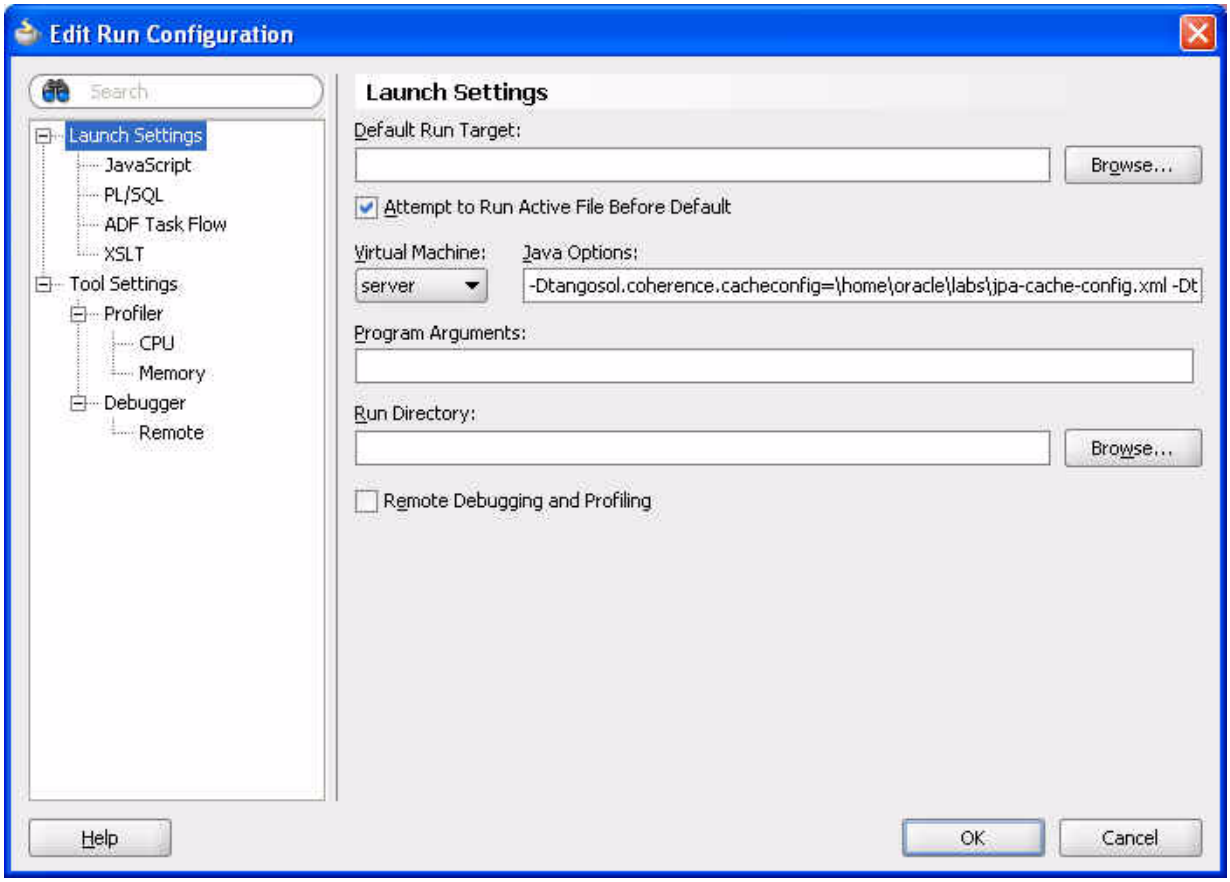
```
C:\oracle\product\coherence\bin>jpa-cache-server.cmd
```

6. Modify the `Lab9` Project Properties.

a. Edit the `Lab9` Project Properties and modify the **Run/Debug/Profile** configuration. Append the following line to the existing **Java Options**.

```
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\jpa-cache-config.xml
```

Figure 7–16 Adding a JPA Cache Configuration to the Runtime Configuration



b. Add additional `CLASSPATH` entries to the existing project properties.

Navigate to **Tools > Project Properties > Libraries and Classpath**. Use the **Add JAR/Directory** and **Add Library** buttons to add the following JAR files and libraries into `CLASSPATH` (Note: the `coherence.jar` file should already be present):

— `eclipselink.jar` for the EclipseLink API:

```
C:\oracle\product\jdeveloper\modules\oracle.toplink_
11.1\eclipselink.jar
```

— `Coherence-eclipselink.jar` for the `CacheStore` that can interact with JPA:

```
C:\oracle\product\jdeveloper\modules\oracle.toplink_
11.1\Coherence-eclipselink.jar
```

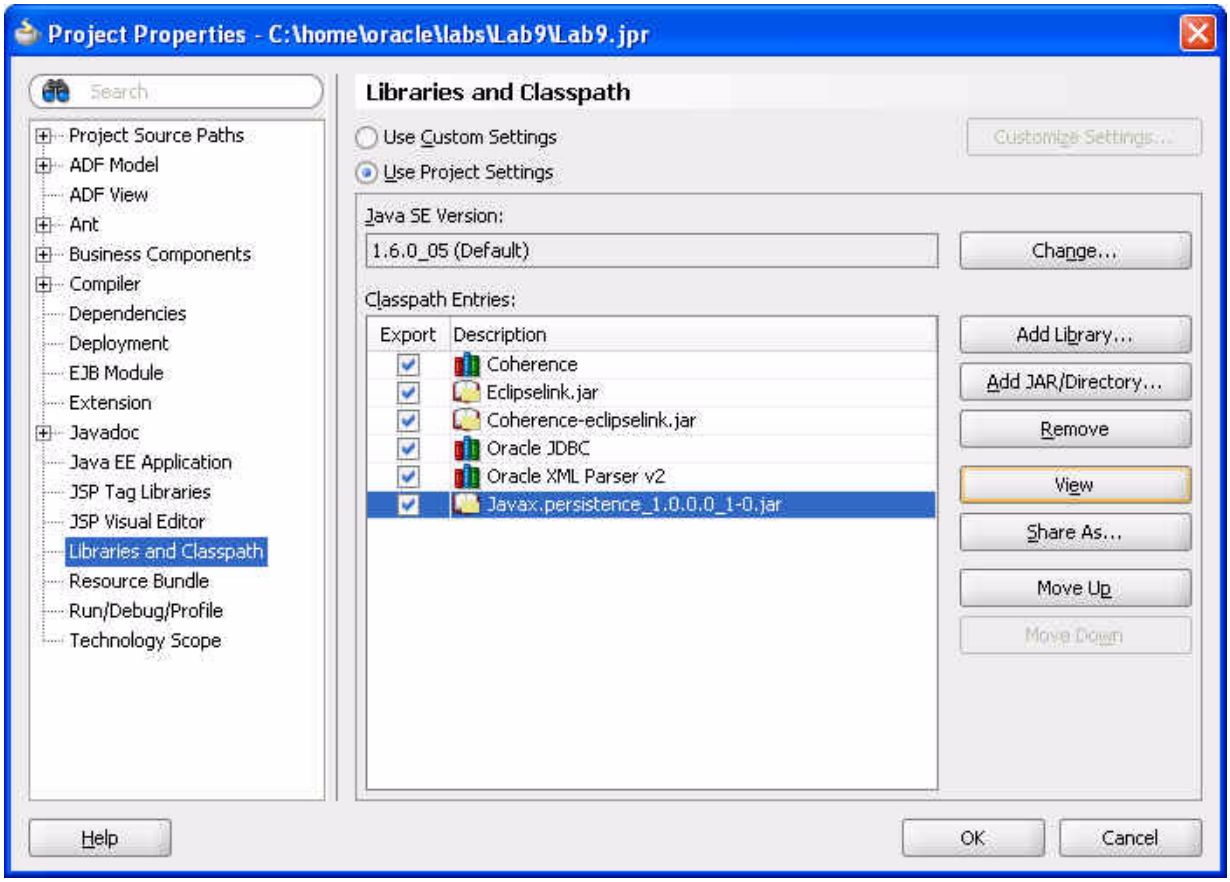
— Java Persistence JAR file for the persistence API:
`C:\oracle\product\modules\javax.persistence_1.0.0.0_1-0.jar`

— `Oracle JDBC` library for database connectivity

— `Oracle XML Parser v2` library for interpreting XML

The **Libraries and Classpath** screen should look similar to Figure 7–17:

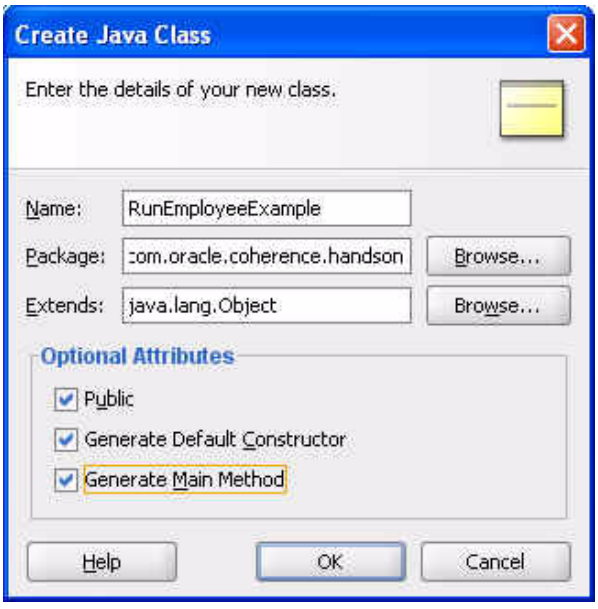
Figure 7–17 Adding JARs and Libraries to the Classpath



7. In `Lab9`, create a new class to interact with the `Employee` object.

   a. Create a new class called `RunEmployeeExample`. Ensure that this has a `main` method.

Figure 7–18 Creating a Java Class



b. Create the code to perform the following:

— Get an employee using `EMPLOYEE_ID`. `EMPLOYEE_ID` should be of the `long` data type.

— Display the salary.

— Give them a 10% pay raise.

— Get the value again to confirm the pay raise.

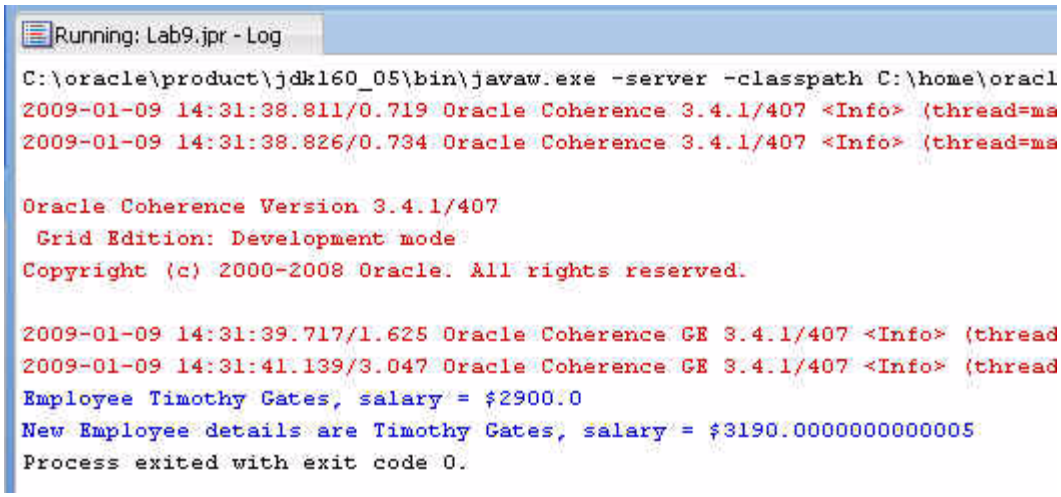
Example 7–4 illustrates a possible solution.

Example 7–4 Sample Employee Class File

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class RunEmployeeExample {
    public RunEmployeeExample() {
    }

    public static void main(String[] args) {
        long empId = 190L;  // emp 190 - Timothy Gates

        NamedCache employees = CacheFactory.getCache("Employees");

        Employees emp = (Employees)employees.get(empId);

        System.out.println("Employee " + emp.getFirstName() + " " +
                           emp.getLastName() + ", salary = $" + emp.getSalary() );

        // give them a 10% pay rise
        emp.setSalary( emp.getSalary() * 1.1);

        employees.put(empId, emp);

        Employees emp2 = (Employees)employees.get(empId);

        System.out.println("New Employee details are " + emp2.getFirstName() + " "
+ emp2.getLastName() + ", salary = $" + emp2.getSalary() );
    }
}
```

c. Run `RunEmployeeExample`. The output should look similar to the output illustrated Figure 7–19.

Figure 7–19 Results from the RunEmployeeExample Application



Now that the `Employees` class has been annotated to persist to the database using JPA, and you have included the `persistence.xml` file to tell JPA where your database is, Coherence uses a `CacheStore` implementation that uses JPA to load and store objects to the database. When you use the `get(Object key)` method, the following happens:

—Coherence looks for the entry with the key.

—If the entry has not already been cached, or it is expired from the cache, Coherence asks the backing map, which uses JPA and EclipseLink to retrieve the data.

—If the entry is in the cache, Coherence returns the entry directly to the application without going through EclipseLink. When you use `put(Object Key,Object Value)`, Coherence uses JPA through EclipseLink to persist any changes to the database.

8

Interacting with the Cache and the Database

In this chapter, you create and configure an Oracle Coherence cache in Oracle JDeveloper using all the concepts presented in this tutorial.

Introduction

A Coherence cache is a collection of data objects that serves as an intermediary between the database and the client applications. Database data may be loaded into a cache and made available to different applications. Thus, Coherence caches reduce load on the database and provide faster access to database data.

Coherence caches provide higher availability through database isolation and data replication. Modifications made to a cache may be synchronized with the database whenever the database is available. Even if the database or an application server node is not available, database updates are still reliable due to the lazy load and lazy write mechanism used by a Coherence cache and due to the failover and fail back provided by Oracle Coherence.

Coherence caches provide distributed processing not only across a cluster of application server nodes but also across the data objects in the cache, because data modification operations may be performed on the data objects.

Oracle Coherence also provides event-based processing. The state of data objects in a cache may be monitored and actions invoked on other processes such as the start of a business process execution language (BPEL) process.

Oracle Coherence supports different types of caches.

- Replicated caches—In a replicated cache, data is replicated to each of the application server nodes in the cluster. This is suitable if faster read access is required but not suitable for writes, because data has to be written to each of the nodes.
- Distributed (or "partitioned") caches—In a distributed cache, data is distributed (load-balanced) across different nodes. Failover is implemented in a distributed cache using backups, which are also distributed across the cluster nodes.

Oracle Coherence is implemented by using services such as the cluster service, the distributed cache service, and the replicated cache service. Whichever type of cache is used, an application uses the same API to access and store data.

The cache configuration deployment descriptor is used to configure a cache. The root element of the cache configuration file is `cache-config`. Cache names and name patterns are mapped to cache types in the `caching-scheme-mapping` element using the subelement `cache-mapping`. Cache types are defined in the `caching-schemes` element. Some of the commonly used cache types are described in Table 8–1.

Table 8–1 Descriptions of Cache Types

Cache Type	Description
distributed scheme	Defines a distributed cache in which data is stored across a cluster of nodes
replicated scheme	Defines a cache in which cache entries are replicated across all the cluster nodes
read-write-backing-map scheme	Defines a map, which provides a cache of a persistent store such as a relational database
external scheme	Defines an external cache such as a disk
class scheme	Defines a custom cache implementation, which is required to implement the java.util. Map interface

Creating and Configuring a Coherence Cache with Oracle JDeveloper

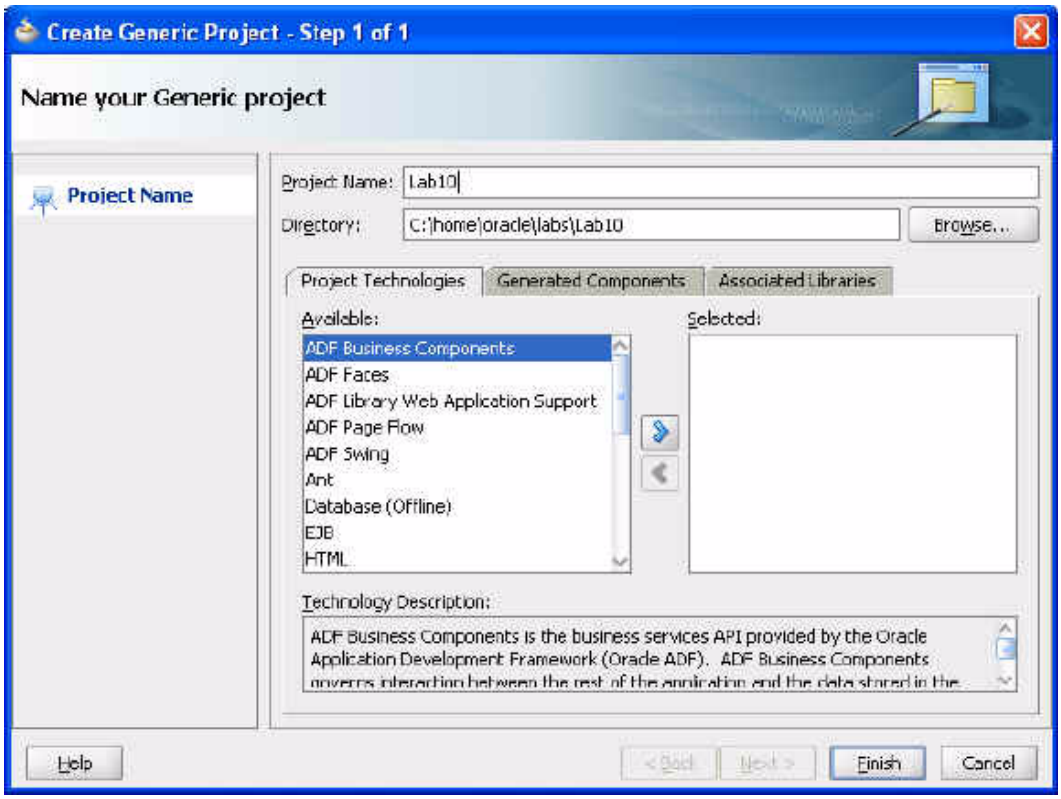
This exercise pulls together all of the concepts presented in this tutorial. In this exercise you will

- Create a Java class that creates a `NamedCache` and can put and get cache entries.
- Create a cache configuration file to define the mapping for cache names, cache types, and naming patterns
- Create a Java class that creates a connection to the Oracle database and can retrieve and store table data.
- Create a database cache. This class will add cache entries, query the database cache, and retrieve entries.

To create and configure a Coherence cache:

1. Create a project and an application in Oracle JDeveloper.
   a. Create a project called `Lab10` in Oracle JDeveloper.

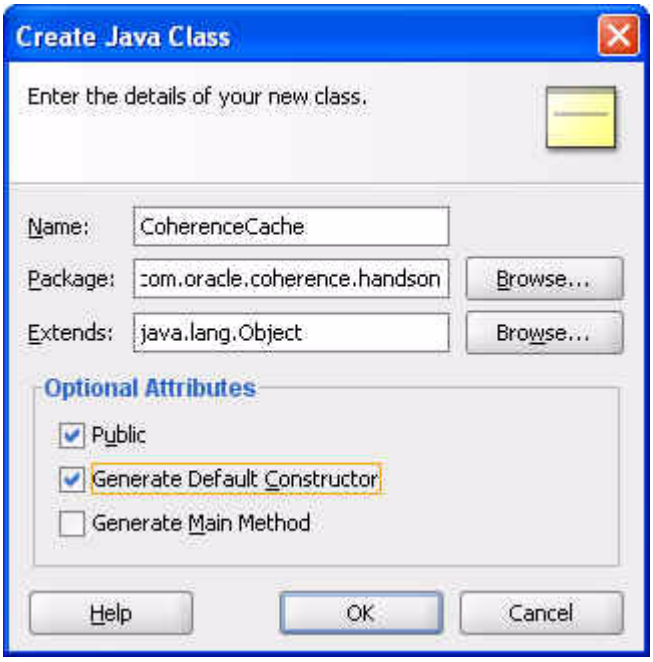
Figure 8–1 Creating a Project



b. Create a Java class, `CoherenceCache`, in the project.

The Java class will be used to create a Coherence cache.

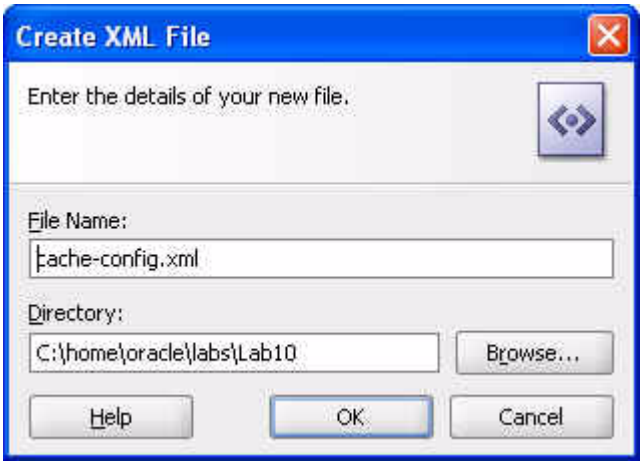
Figure 8–2 Creating a Java Class



c. Create an XML document, `cache-config.xml`, as the cache configuration deployment descriptor.

   To add the XML document, right-click **Lab10 > New**. In the **New Gallery** window, select **XML** under **General** categories. Select **XML Document** and click **OK**. Change the **File Name** to `cache-config.xml`. Note: You will update the contents of `cache-config.xml` in Step 4 of this exercise.

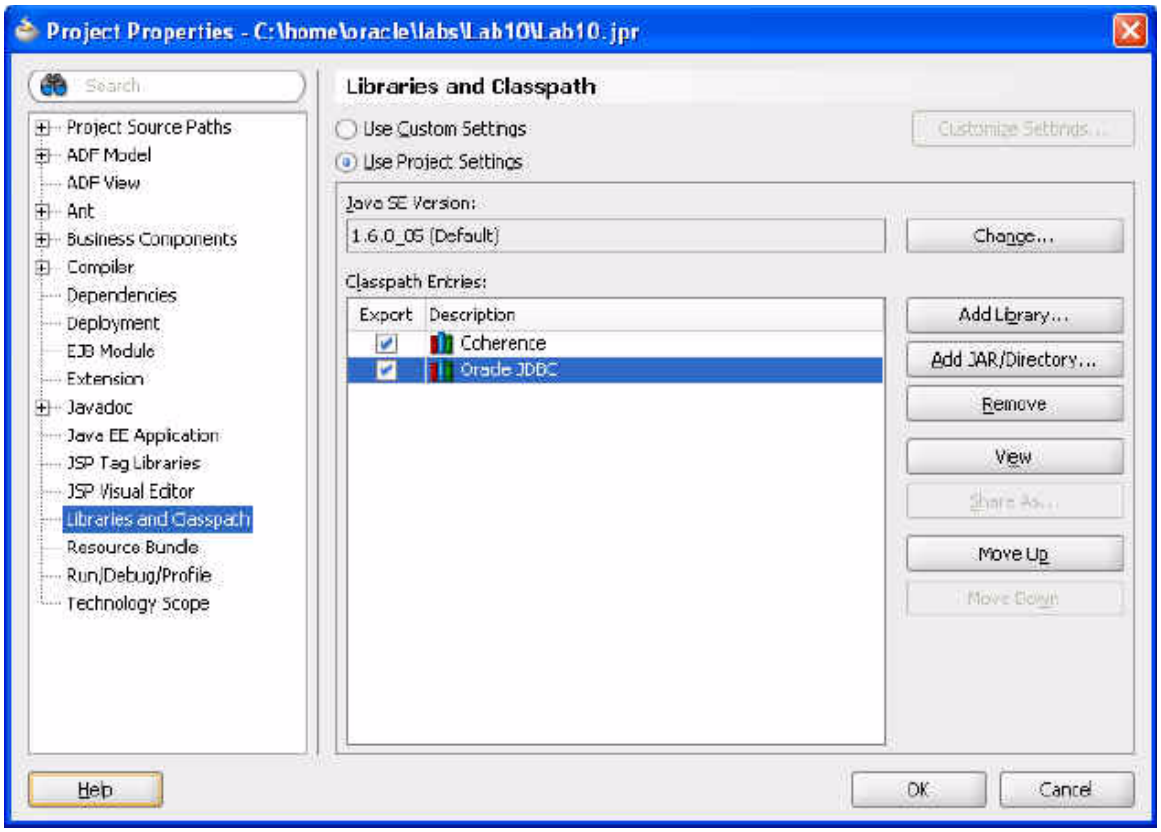
Figure 8–3 Creating an XML File



2. Add the Coherence JAR file `coherence.jar` to the project libraries. Also add the Oracle JDBC library, which is required for database access to the project libraries.

   The Coherence JAR file is in the `oracle\product\coherence\lib` directory (the Oracle Coherence installation). It should already be present in the Libraries and Classpath dialog box. To add the Oracle JDBC library, click **Tools > Project Properties > Add Library > Oracle JDBC**.

Figure 8–4 Adding the Oracle JDBC Libraries to the Classpath



3. Modify the Run configuration for the application to add the cache configuration file as a run-time Java option.

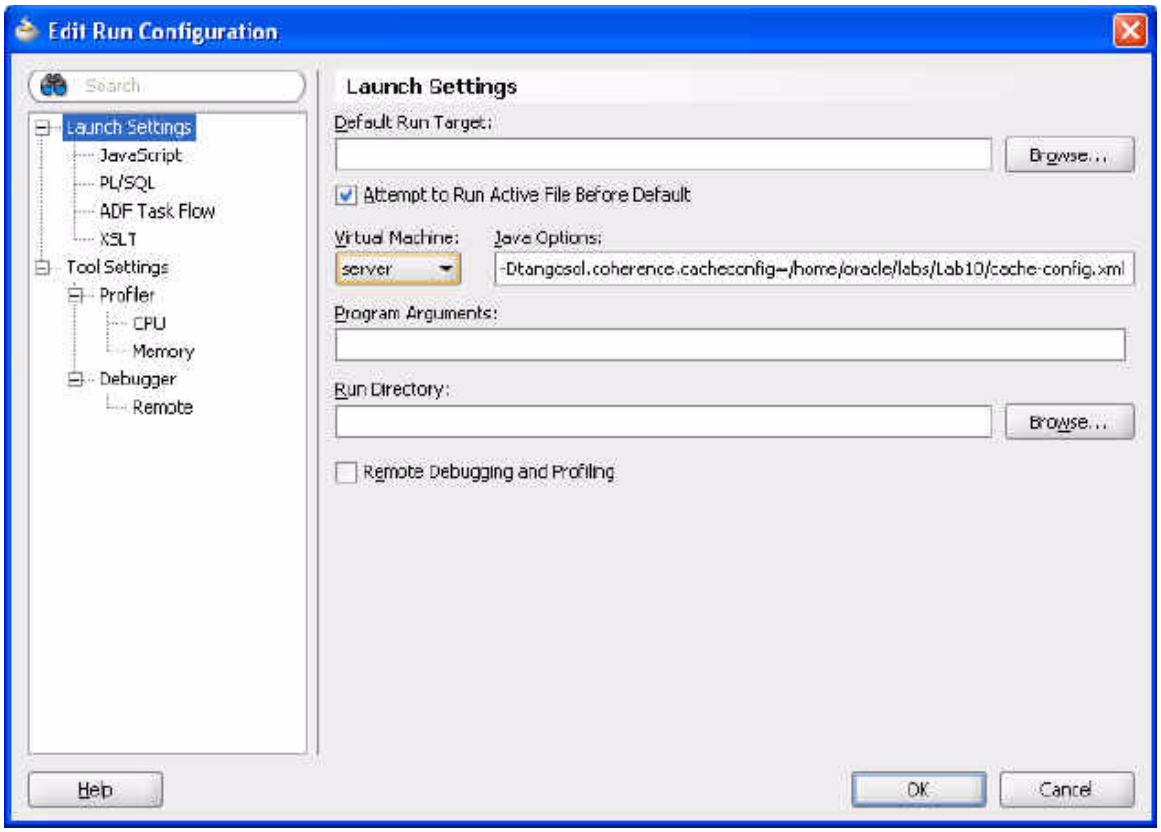
   a. Select the project node and select **Tools > Project Properties**. In the **Project Properties** window, select **Run/Debug/Profile**. The **Default** configuration is selected by default. Click **Edit** for the **Default** configuration.

   b. In the **Edit Run Configuration** window, select **Launch Settings**. In the **Java Options** field, specify the cache configuration file (`cache-config.xml`) with `-Dtangosol.coherence.cacheconfig=[`*`path-to-cache-config-file`*`]/cache-config.xml`.

      For example, for the Oracle Coherence application that you created, specify the following (your path to `cache-config.xml` may vary) in the **Java Options** field and click **OK**.

      ```
      -Dtangosol.coherence.cacheconfig=/home/oracle/labs/Lab10/cache-config.xml
      ```

Figure 8–5 Setting the Cache Configuration File for Runtime Options



Click **OK** in the **Run/Debug/Profile** window. The cache configuration file will be added as a run-time Java option to the Coherence Java application.

4. In the cache configuration file:

   - Define mapping for cache names and naming patterns with the `cache-mapping` elements in the `caching-scheme-mapping` element.
   - Specify the default mapping to cache type `default-replicated` and map cache name `VirtualCache` to cache type `default-distributed`.
   - Define the distributed caching scheme with the `distributed-scheme` element using the `DistributedCache` service.

   The cache configuration file is listed in Example 8–1. Copy the contents of this example to the `cache-config.xml` file in Oracle JDeveloper.

Example 8–1 Cache Configuration File

```
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
    <caching-scheme-mapping>

        <cache-mapping>
            <cache-name>VirtualCache</cache-name>
            <scheme-name>default-distributed</scheme-name>
        </cache-mapping>
    </caching-scheme-mapping>
    <caching-schemes>
        <!--
```

```
        Default Distributed caching scheme.
        -->
        <distributed-scheme>
            <scheme-name>default-distributed</scheme-name>
            <service-name>DistributedCache</service-name>
            <backing-map-scheme>
                <class-scheme>
                    <scheme-ref>default-backing-map</scheme-ref>
                </class-scheme>
            </backing-map-scheme>
        </distributed-scheme>
  <class-scheme>
            <scheme-name>default-backing-map</scheme-name>
            <class-name>com.tangosol.util.SafeHashMap</class-name>
            </class-scheme>
 </caching-schemes>
</cache-config>
```

5. Create the cache application.

   a. Create a cache in the `CoherenceCache` Java class. Import the `CacheFactory` class and the `NamedCache` interface.

      ```
      import com.tangosol.net.CacheFactory;
      import com.tangosol.net.NamedCache;
      ```

   b. An instance of a cache is created from the `CacheFactory` class. Create a `NamedCache` using the `getCache()` method of the `CacheFactory` class. Use the cache name `VirtualCache`, which is mapped to a distributed caching scheme.

      ```
      NamedCache cache = CacheFactory.getCache ( "VirtualCache");
      ```

   c. A `NamedCache` is a `java.util.Map` that holds resources that are shared across nodes in a cluster. Add a cache entry using the `put()` method.

      ```
      cache.put (key, "Hello Cache");
      ```

   d. A cache entry can be retrieved using the `get()` method.

      ```
      System.out.println((String)cache.get("hello"));
      ```

      Example 8–2 illustrates a possible solution. You can copy the code to the `CoherenceCache` application in Oracle JDeveloper.

Example 8–2 Implementation of a Coherence Cache

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class CoherenceCache {
    NamedCache cache;
    public CoherenceCache() {
    }
    public void putCache(){
            cache = CacheFactory.getCache ( "VirtualCache");
           String key = "hello";
           cache.put (key, "Hello Cache");

       }
```

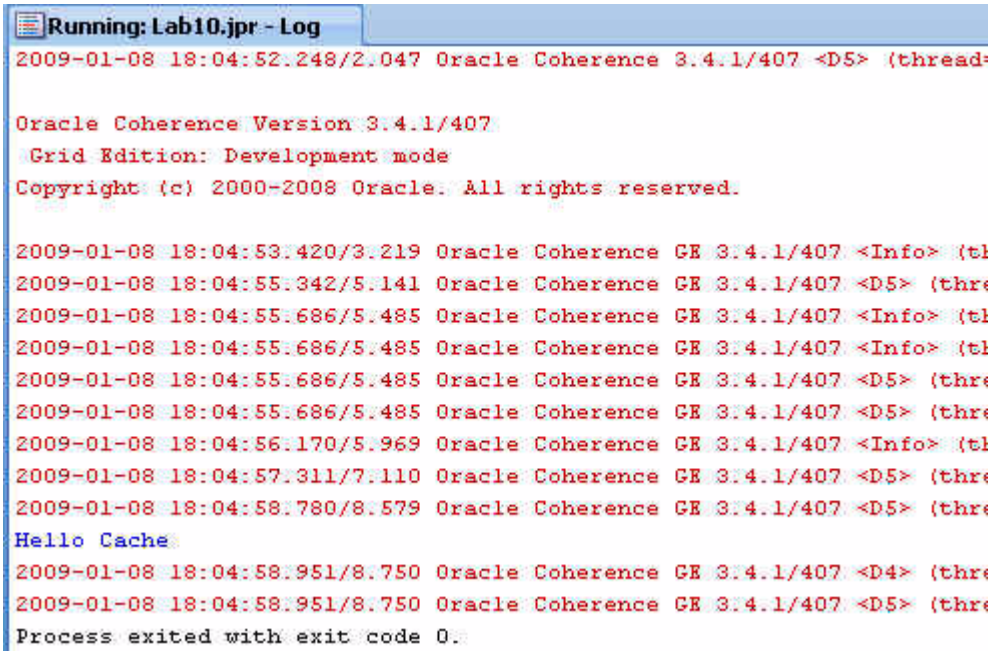
```
    public void retrieveCache(){

        System.out.println((String)cache.get("hello"));

    }

     public static void main (String [] args) {
     CoherenceCache cache = new CoherenceCache();
     cache.putCache();
     cache.retrieveCache();
     }
}
```

e. Stop any running cache servers. Start the JPA cache server (`jpa-cache-server.cmd`) that you created in Chapter 7, "Using JPA with Coherence".

f. Right-click the Oracle Coherence application `CoherenceCache.java` and click **Run**.

The Oracle Coherence application runs and the output is displayed in the Log window. The output shows that the operational configuration is loaded from `tangosol-coherence.xml`, the cache configuration is loaded from `cache-config.xml`, a new cluster is created, and the `DistributedCache` service joins the cluster. The operational deployment descriptor `tangosol-coherence.xml` specifies the operational and run-time settings used by Coherence for its clustering, communication, and data management services.

Figure 8–6 Output from the CoherenceCache Program



If you see any exceptions, such as the following, can you determine the reason?

```
"java.lang.IllegalArgumentException: No scheme for cache: "cachename"
```

6. Create an Oracle database cache.

   In this section, you create an Oracle database cache, which is a cache backed by the Oracle database.

   a. Invoke SQL*Plus.

      Navigate to **Start > All Programs > Oracle Database 10g Express Edition > Run SQL Command Line**.

   b. Connect as `hr` user with `hr` as the password.

      ```
      connect hr/hr;
      ```

   c. Create an Oracle Database table.

      Open a text editor and copy the following SQL code. Save the file as `dbscript.sql` in the `/home/oracle/labs/` folder.

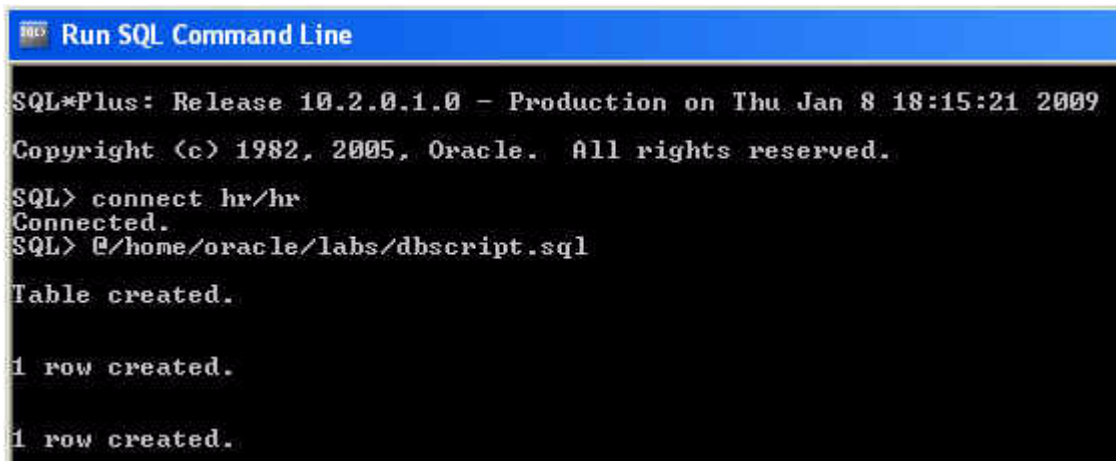
Example 8–3 SQL Script for Creating a Database Table

```
CREATE TABLE HR.CATALOG(id VARCHAR(25) PRIMARY KEY, value VARCHAR(96));
INSERT INTO HR.CATALOG VALUES('catalog1', 'Tuning Undo Tablespace');
INSERT INTO HR.CATALOG VALUES('catalog2', 'Tuning Your View Objects');
```

   d. Run the SQL script.

      Figure 8–7 illustrates Steps 6b, 6c, and 6d.

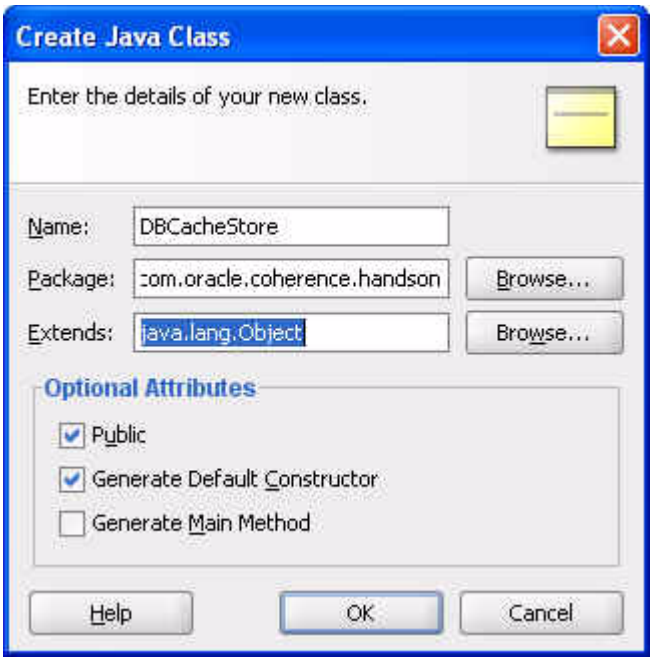
Figure 8–7 Creating a Table in the Oracle Database



7. Create a custom `CacheStore`.

   a. Create a Java class `DBCacheStore` in Oracle JDeveloper.

Figure 8–8 Creating a Java Class



b. Create the code to connect to the database and get table data.

Example 8–4 illustrates a possible solution. Copy the code to the `DBCacheStore` application in Oracle JDeveloper. The `DBCacheStore` application uses Java Database Connectivity (JDBC) to access Oracle Database, but another mechanism, such as Hibernate or Java Data Objects (JDO), may also be used.

Example 8–4 Database CacheStore Implementation

```
package com.oracle.coherence.handson;

import com.tangosol.net.cache.CacheStore;
import com.tangosol.util.Base;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

public class DBCacheStore

        extends Base
        implements CacheStore {

        protected Connection m_con;
        protected String m_sTableName;
        private static final String DB_DRIVER   = "oracle.jdbc.OracleDriver";
```

```
            private static final String DB_URL      =
"jdbc:oracle:thin:@localhost:1521:XE";
            private static final String DB_USERNAME = "hr";
            private static final String DB_PASSWORD = "hr";

    public DBCacheStore(String sTableName)
        {
        m_sTableName = sTableName;

        configureConnection();
        }
    protected void configureConnection()
        {
        try
            {
            Class.forName("oracle.jdbc.OracleDriver");
            m_con = DriverManager.getConnection(DB_URL, DB_USERNAME, DB_PASSWORD);
            m_con.setAutoCommit(true);
            }
        catch (Exception e)
            {
            throw ensureRuntimeException(e, "Connection failed");
            }
        }

    public String getTableName()
        {
        return m_sTableName;
        }

    public Connection getConnection()
        {
        return m_con;
        }

    public Object load(Object oKey)
        {
        Object     oValue = null;
        Connection con    = getConnection();
        String     sSQL   = "SELECT id, value FROM " + getTableName()
                    + " WHERE id = ?";
        try
            {
            PreparedStatement stmt = con.prepareStatement(sSQL);

            stmt.setString(1, String.valueOf(oKey));
            ResultSet rslt = stmt.executeQuery();
            if (rslt.next())
                {
                oValue = rslt.getString(2);
                if (rslt.next())
                    {
                    throw new SQLException("Not a unique key: " + oKey);
                    }
                }
            stmt.close();
            }
        catch (SQLException e)
            {
```

```
            throw ensureRuntimeException(e, "Load failed: key=" + oKey);
            }
        return oValue;
        }

      public void store(Object oKey, Object oValue)
        {
        Connection con     = getConnection();
        String     sTable  = getTableName();
        String     sSQL;

        if (load(oKey) != null)
            {

            sSQL = "UPDATE " + sTable + " SET value = ? where id = ?";
            }
        else
            {

            sSQL = "INSERT INTO " + sTable + " (value, id) VALUES (?,?)";
            }
        try
            {
            PreparedStatement stmt = con.prepareStatement(sSQL);
            int i = 0;
            stmt.setString(++i, String.valueOf(oValue));
            stmt.setString(++i, String.valueOf(oKey));
            stmt.executeUpdate();
            stmt.close();
            }
        catch (SQLException e)
            {
            throw ensureRuntimeException(e, "Store failed: key=" + oKey);
            }
        }
    public void erase(Object oKey)
        {
        Connection con  = getConnection();
        String     sSQL = "DELETE FROM " + getTableName() + " WHERE id=?";
        try
            {
            PreparedStatement stmt = con.prepareStatement(sSQL);

            stmt.setString(1, String.valueOf(oKey));
            stmt.executeUpdate();
            stmt.close();
            }
        catch (SQLException e)
            {
            throw ensureRuntimeException(e, "Erase failed: key=" + oKey);
            }
        }
    public void eraseAll(Collection colKeys)
        {
        throw new UnsupportedOperationException();
        }



    public Map loadAll(Collection colKeys)
```

```
        {
        throw new UnsupportedOperationException();
        }

    public void storeAll(Map mapEntries)
        {
        throw new UnsupportedOperationException();
        }

    public Iterator keys()
        {
        Connection con  = getConnection();
        String     sSQL = "SELECT id FROM " + getTableName();
        List       list = new LinkedList();

        try
            {
            PreparedStatement stmt = con.prepareStatement(sSQL);
            ResultSet         rslt = stmt.executeQuery();
            while (rslt.next())
                {
                Object oKey = rslt.getString(1);
                list.add(oKey);
                }
            stmt.close();
            }
        catch (SQLException e)
            {
            throw ensureRuntimeException(e, "Iterator failed");
            }

        return list.iterator();
        }

    }
```

c. Modify the cache configuration file that you created earlier (`cache-config.xml`) for the database cache.

To connect a cache to a back-end database, a cache configuration file (`cache-config.xml`) element `cachestore-scheme` is required. The `cachestore-scheme` element must be configured with a custom class that implements either the `com.tangosol.net.cache.CacheLoader` or `com.tangosol.net.cache.CacheStore` interface.

Copy the cache configuration file for the database cache in Example 8–5 and Replace the existing code in the `cache-config.xml` file in Oracle JDeveloper.

Example 8–5 Database Cache Configuration File

```
<?xml version="1.0" encoding="UTF-8" ?>
<cache-config>
    <caching-scheme-mapping>


  <!--
    Caches with names that start with 'DBBacked' will be created
    as distributed-db-backed.
    -->
```

```
  <cache-mapping>
   <cache-name>DBBacked*</cache-name>
   <scheme-name>distributed-db-backed</scheme-name>
  </cache-mapping>
 </caching-scheme-mapping>
 <caching-schemes>
  <!--
    DB Backed Distributed caching scheme.
    -->
  <distributed-scheme>
   <scheme-name>distributed-db-backed</scheme-name>
   <service-name>DistributedCache</service-name>
   <backing-map-scheme>
    <read-write-backing-map-scheme>
     <internal-cache-scheme>
      <class-scheme>
       <class-name>com.tangosol.util.ObservableHashMap</class-name>
      </class-scheme>
     </internal-cache-scheme>
     <cachestore-scheme>
      <class-scheme>
       <class-name>com.oracle.coherence.handson.DBCacheStore</class-name>
       <init-params>
        <init-param>
         <param-type>java.lang.String</param-type>
         <param-value>CATALOG</param-value>
        </init-param>
       </init-params>
      </class-scheme>
     </cachestore-scheme>
     <read-only>false</read-only>
     <!--
        To make this a write-through cache just change the value below to 0 (zero)
        -->
     <write-delay-seconds>0</write-delay-seconds>
    </read-write-backing-map-scheme>
   </backing-map-scheme>
   <listener/>
   <autostart>true</autostart>
  </distributed-scheme>
 </caching-schemes>
</cache-config>
```

In the cache configuration file, you have taken care of the following:

—Define a cache name pattern `DBBacked*`, which is mapped to a distributed caching scheme `distributed-db-backed`.

—Specify the `CacheStore` scheme in the distributed scheme using the class `coherence.DBCacheStore`, which implements the `CacheStore` interface.

—An `init` parameter for the database table that is at the back end of the cache is specified for the `DBCacheStore` class. The table name is specified in the `init-param` element. The `DBCacheStore` class performs database operations such as reading and writing cache entries.

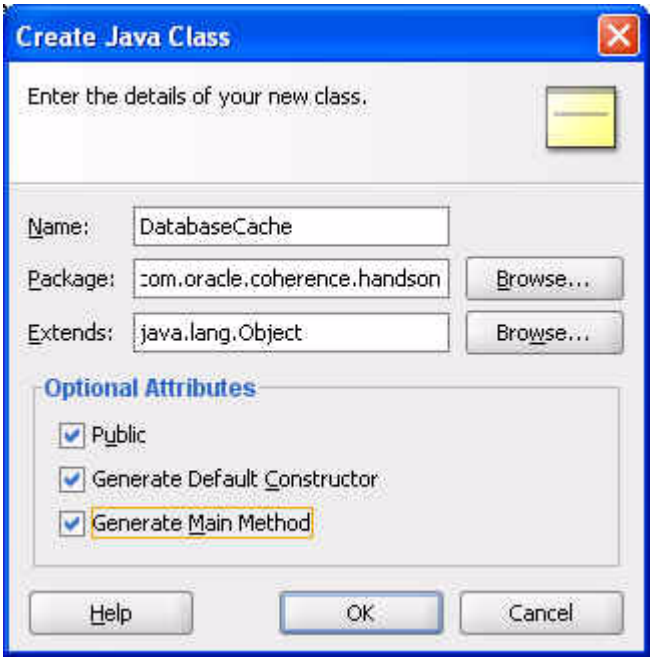
—Coherence supports read/write caching of a data source for which the `read-write-backing-map` scheme is used. The `read-write-backing-map` scheme defines a backing map, which provides a size-limited cache of a persistent store. Here, you use the Write-Through mechanism. Oracle Coherence supports the types of read/write caching described in Table 8–2:

Table 8–2 Types of Read-Write Caching Supported by Coherence

Types of Read-Write Caching	Action
Read-Through	A cache entry is read into a cache from the database when required and made available to an application.
Write-Through	Updates to cache entries are synchronized with the database without delay.
Refresh-Ahead	Cache entries are refreshed periodically.
Write-Behind	Updates to cache entries are asynchronously written to a database after a delay specified in the write-delay-seconds element in the cache configuration file.

8. Create a Java class `DatabaseCache` for the database cache in Oracle JDeveloper. The class must contains a `main` method.

Figure 8–9 Creating a Java Class



In the class file, add code to add a cache entry, query a database cache, and retrieve a cache entry. Add the following methods: `createCache()`, `addEntry()`, `retrieveEntry()`, `eraseEntry()`, and `queryCache()`. You can copy the code that is listed in Example 8–6.

Example 8–6 Implementation for the Database Cache Class File

```
package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.net.cache.ContinuousQueryCache;
import com.tangosol.util.Filter;
import com.tangosol.util.extractor.IdentityExtractor;
import com.tangosol.util.filter.LikeFilter;

import java.util.HashSet;
```

```
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

public class DatabaseCache {
    NamedCache cache;
    public DatabaseCache() {
    }

    public void createCache() {
         cache = CacheFactory.getCache("DBBackedCache");
        //cache.put(new String("catalog3"), new String("Evolving Grid
Management"));
          // System.out.println((String) cache.get( "catalog3"));

    }

    public void addEntry() {

        cache.put(new String("catalog3"), new String("Tuning Grid Management"));
        cache.put(new String("catalog4"), new String("Tuning Coherence"));
        cache.put(new String("catalog5"), new String("Tuning Database"));
        //System.out.println((String) cache.get( "catalog3"));

    }

    public void retrieveEntry() {
    System.out.println((String) cache.get( "catalog3"));
    }

    public void eraseEntry() {
    cache.remove(new String("catalog3"));
    }

    public void queryCache() {
        Filter filter = new LikeFilter(IdentityExtractor.INSTANCE, "Tuning%",
'\\', true);
        HashSet hashSet=new HashSet();
        hashSet.add(new String("catalog3"));
        hashSet.add(new String("catalog4"));
        hashSet.add(new String("catalog5"));

        Map map=cache.getAll(hashSet);
        //ContinuousQueryCache queryCache = new ContinuousQueryCache(cache,
filter);
        //Set results = queryCache.entrySet(filter);
        Set results = cache.entrySet(filter);
       /* Set results = cache.entrySet(filter);*/



       // if(results.isEmpty())
       // System.out.println("Result Set Empty");
            for (Iterator i = results.iterator(); i.hasNext();)
                {
                Map.Entry e = (Map.Entry) i.next();
                System.out.println("Catalog ID: "+e.getKey() + ", Title: "+e.
getValue());
                }
```

```
    }


    public static void main(String[] args) {
        DatabaseCache databaseCache = new DatabaseCache();
        databaseCache.createCache();
       databaseCache.addEntry();
       //databaseCache.retrieveEntry();
        //databaseCache.eraseEntry();
       databaseCache.queryCache();
    }
}
```

Note the following features of the code:

- A `NamedCache` object is created using the `getCache()` method of the `CacheFactory` class in the `addEntry()` method.

  ```
  NamedCache cache = CacheFactory.getCache("DBBackedCache");
  ```

- The `DBBackedCache` matches the cache pattern `DBBacked*` and is, therefore, mapped to a distributed caching scheme `distributed-db-backed` in the `cache-config.xml` file. Add a cache entry using the `put()` method of the `NamedCache` object.

  ```
  cache.put(new String("catalog3"), new String("Tuning Grid Management"));
  ```

- Because the Write-Through mechanism is used, the new cache entry also gets synchronized with the database; a new row is added to the `CATALOG` table. Comment out all the methods except the `createCache()` and `addEntry()` methods.
- When the `put()` method is invoked, the `store()` method, which maps the new cache entry to the database table `CATALOG` using JDBC, gets invoked in the `DBCacheStore` class. The output from the Oracle Coherence application is displayed in the Log window and a new cache entry is added. The output shows that the operational configuration deployment descriptor is loaded, the cache configuration is loaded, a new cluster is created, and the `DistributedCache` service has joined the cluster.
- The new cache entry may be removed with the `remove()` method of the `NamedCache` object.

  ```
  cache.remove(new String("catalog3"));
  ```

- Bulk uploading of cache entries is performed using the `putAll()` method.
- A cache entry is retrieved using the `get()` method of the `NamedCache` object. For example, retrieving the cache entry for ID `catalog1`:

  ```
  System.out.println((String) cache.get("catalog1"));
  ```

- When the `get()` method is invoked, the `load()` method, which retrieves database table data using JDBC, gets invoked in the `DBCacheStore` class.
- Bulk retrieval is performed using the `getAll()` method of the `NamedCache` object.
- Oracle Coherence supports searching for cache entries based on a search criteria using filters. Coherence filters are available in the `com.tangosol.util.filter` package. In Oracle Coherence Enterprise Edition and Grid Edition, indexes may be added to the Coherence cache to improve

performance. You query the database cache using a `LikeFilter` filter, which matches cache entries with a specified pattern. To query a database cache, the cache entries are required to be created before querying; the cache entries must be retrieved into the cache using the `get()` or `getAll()` method before a query using a filter may be performed. Therefore, you can retrieve database data and create a collection of cache entries using the `getAll()` method.

```
HashSet hashSet=new HashSet();
hashSet.add(new String("catalog1"));
hashSet.add(new String("catalog2")); hashSet.add(new String("catalog3"));
Map map=cache.getAll(hashSet);
```

- A `LikeFilter` filter is created to search for cache entries starting with `Tuning`.

```
Filter filter = new LikeFilter(IdentityExtractor.INSTANCE, "Tuning%", '\\',
true);
```

- The database cache is queried using the `entrySet()` method with the `LikeFilter` filter.

```
Set results = cache.entrySet(filter);
```

- Iterate over the results of the query to output the cache entries retrieved.

```
for (Iterator i = results.iterator(); i.hasNext();) { Map.Entry e = (Map.
Entry) i.next();
System.out.println("Catalog ID: "+e.getKey() + ", Title: "+e.getValue()); }
```

- Oracle Coherence supports continuous query using the `com.tangosol.net.cache.ContinuousQueryCache` class. A continuous query is a query that is kept up-to-date using a continuous query cache. In a `ContinuousQueryCache`, the results of a query are updated using event listeners on events that could change the results of the query. Create a `ContinuousQueryCache` object using the `NamedCache` object and the `LikeFilter` object.

```
ContinuousQueryCache queryCache = new ContinuousQueryCache(cache, filter );
```

- A result set is created using the `entrySet()` method.

```
Set results = queryCache.entrySet(filter);
```

9. Stop any running cache servers. Start the JPA cache server (`jpa-cache-server.cmd`) that you created in Chapter 7, "Using JPA with Coherence".
10. Right-click the `DatabaseCache` application in Oracle JDeveloper and select **Run**. Figure 8–10 illustrates the expected results.

Figure 8–10 Results from Running the DatabaseCache Application

If you receive any exceptions, such as the following:

```
java.lang.IllegalArgumentException: No scheme for cache: "cachename,
```

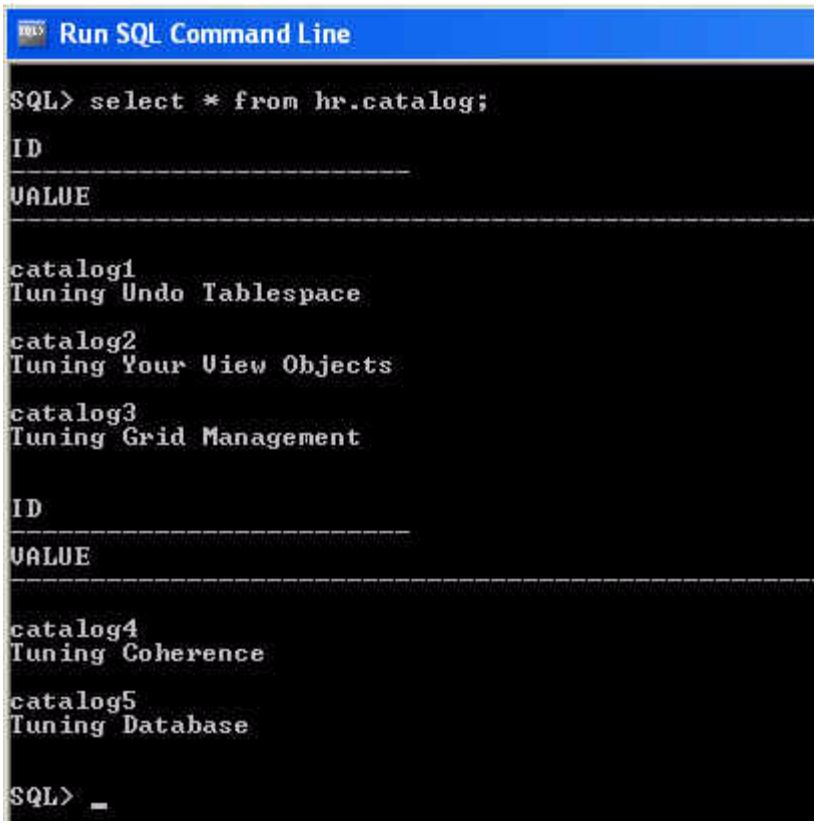
You may be able to remove them by editing the `cache-config.xml` file and replacing `DBBacked*` in the `<cache-name>` element with `*`. Save the file. Re-run the `DatabaseCache` application in Oracle JDeveloper. You should not see any exceptions now.

11. Note that because you are using a Write-Through cache, the database table also gets updated. From the SQL prompt, enter the following code:

```
select * from hr.catalog;
```

Figure 8–11 illustrates the results.

Figure 8–11 Updates to Database Tables



Index

N

O

P

Q

R

T

U

W

X