

Oracle® Coherence
Tutorial for Oracle Coherence
Release 3.5
E14527-01

June 2009

Oracle Coherence Tutorial for Oracle Coherence, Release 3.5

E14527-01

Copyright © 2009, Oracle and/or its affiliates. All rights reserved.

Primary Author: Thomas Pfaeffle

Contributing Author: Noah Arliss, Mark Falco, Alex Gleyzer, Gene Gleyzer, David Guy, Charlie Helin, Adam Leftik, Tim Middleton, Brian Oliver, Patrick Peralta, Cameron Purdy

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	xi
Audience	xi
Documentation Accessibility	xi
Related Documents	xii
Conventions	xii
1 Installing Coherence and JDeveloper	
Downloading and Installing Coherence	1-1
Downloading and Installing JDeveloper	1-1
Testing a Coherence Installation	1-2
Troubleshooting Cache Server Clustering	1-12
Restricting Coherence to Your Own Host	1-12
Advanced Steps to Restrict Coherence to Your Own Host	1-13
2 Using JDeveloper with Coherence	
Configuring Oracle JDeveloper for Coherence	2-1
JDeveloper Basics	2-8
Creating a New Application and Project	2-8
Creating a New Project in an Existing Application	2-11
Creating a Java Class	2-13
Changing Project Properties, Setting Runtime Configuration, and	2-15
Adding JARS and Libraries to the Project Classpath	2-16
Accessing the Data Grid from Java	2-18
Creating Your First Coherence-Based Java Application	2-23
3 Working with Complex Objects	
Introduction	3-1
Creating and Caching Complex Objects	3-2
Creating the Data Objects	3-2
Creating the Complex Object	3-10
Creating the Driver Class	3-14
Creating Configuration Files and the Cache Server Executable	3-15
Running the Sample	3-17

4	Loading Data Into a Cache	
	Introduction.....	4-1
	Populating a Cache with Domain Objects	4-1
	Querying and Aggregating Data in the Cache	4-12
5	Listening for Changes and Modifying Data	
	Introduction.....	5-1
	Creating a Cache Listener.....	5-2
	Responding to Changes in the Cache.....	5-5
6	Using JPA with Coherence	
	Introduction.....	6-1
	Mapping Relational Data to Java Objects with JPA.....	6-1
7	Interacting with the Cache and the Database	
	Introduction.....	7-1
	Creating a Cache Application	7-2
	Creating a Database Cache	7-7
8	Caching HTTP Sessions with Coherence*Web	
	Install Coherence*Web on WebLogic 10.X	8-1
	Start a Cache Server	8-1
	Configure the WebLogic Server.....	8-2
	Deploy the coherence-web-spi JAR File.....	8-5
	Create the Counter Web Application.....	8-7
	Deploy the Application.....	8-8
	Verify the Example.....	8-9
A	Coherence Client Application Commands	
	bye.....	A-1
	bulkput <# of iterations> <block size> <start key>	A-1
	clear	A-1
	destroy	A-1
	get <key>.....	A-1
	hash	A-2
	help.....	A-2
	inc <key> [<count>]	A-2
	kill	A-2
	list [<map name>].....	A-2
	listen [('start' 'stop') [('cluster' 'local')]]	A-2
	lock <key> [<timeout>].....	A-3
	log (<size> <message>) [<iterations> [<level>]].....	A-3
	map [('Optimistic' 'Replicated' 'Distributed') ':' <map name> [COH33UG:<limit>]]	A-3
	maps	A-3
	memory.....	A-3

put <key> <value>	A-3
release	A-3
remove <key>	A-3
scan <start key> <stop key>	A-4
services	A-4
size	A-4
unlock <key>	A-4
waitkey <start key> <stop key>	A-4
who	A-4
whoami	A-4
worker <command>	A-4
#<repeat count> <command>	A-5
@<command>	A-5
&<functionName> [paramValue]*	A-5

Index

List of Examples

1-1	cache-server.cmd File with an Edited COHERENCE_HOME	1-3
1-2	Output from Starting a Coherence Cache Server	1-4
1-3	coherence.cmd File with an Edited COHERENCE_HOME	1-6
1-4	Output from Starting the Coherence Cache Client	1-7
1-5	Output from Starting a Coherence Cache	1-9
1-6	Exercising Coherence Commands	1-10
1-7	Multicast-Listener Fragment of tangosol-coherence.xml File	1-12
2-1	Creating a NamedCache; Inserting and Verifying Values	2-20
2-2	Getting a Value from the Cache	2-20
2-3	Sample Coherence-Based Java Application	2-24
3-1	Implementation of an Address Class	3-5
3-2	Implementation of a Phone Class	3-8
3-3	Sample Contact Class	3-11
3-4	Sample ContactDriver Class	3-14
3-5	POF Configuration File	3-16
3-6	Cache Configuration File	3-16
3-7	Sample Cache Server Executable	3-17
3-8	Starting the POF Cache Server	3-18
4-1	Simple Contact ID Class	4-2
4-2	POF Configuration File with the ContactId Entry	4-4
4-3	Sample Data Generation Class	4-5
4-4	Sample Cache Loading Program	4-8
4-5	Sample contacts-cache-server.cmd File	4-11
4-6	Sample QueryExample Class	4-15
4-7	Methods to Aggregate Over Keys or by Specifying Filters	4-18
4-8	QueryExample with Aggregation	4-18
5-1	Listener Methods on a NamedCache	5-1
5-2	Code Pattern for Registering an Event	5-1
5-3	Sample Listener Class	5-2
5-4	Sample Program to Update an Object in the Cache	5-6
6-1	persistance.xml File Contents	6-11
6-2	Cache Configuration for JPA	6-12
6-3	Modified jpa-cache-server.cmd File	6-13
6-4	Sample Employee Class File	6-15
7-1	Cache Configuration File	7-4
7-2	Implementation of a Coherence Cache	7-5
7-3	Output of the Coherence Cache Application	7-6
7-4	SQL Script for Creating a Database Table	7-7
7-5	Running the SQL Script for Creating a Database Table	7-7
7-6	Database CacheStore Implementation	7-8
7-7	Database Cache Configuration File	7-11
7-8	Implementation for the Database Cache Class File	7-13
7-9	Output from the select Command	7-17
8-1	Script to Start the Cache Server	8-1

List of Tables

1-1	Network Addresses and Ports Used by Coherence.....	1-2
2-1	Methods in the NamedCache Interface	2-19
2-2	Methods in the CacheFactory Class	2-19
7-1	Descriptions of Cache Types	7-2
7-2	Types of Read-Write Caching Supported by Coherence	7-12

List of Figures

2-1	Select Role Dialog Box.....	2-2
2-2	Creating an Application in JDeveloper.....	2-3
2-3	Default Properties Dialog Box.....	2-4
2-4	Add Library Dialog Box.....	2-5
2-5	Create Library Dialog Box	2-6
2-6	Select Path Entry Dialog Box.....	2-7
2-7	Create Library Dialog Box with the Coherence Jar on the Classpath	2-8
2-8	Creating a New Application in the New Gallery	2-9
2-9	Providing an Application Name.....	2-10
2-10	Providing a Project Name.....	2-11
2-11	Creating a New Project in an Existing Application	2-12
2-12	Providing Details for the New Project	2-13
2-13	Creating a Java Class in the New Gallery	2-14
2-14	Providing Details for the Java Class.....	2-14
2-15	Project Properties Dialog Box.....	2-15
2-16	Setting the Runtime Configuration	2-16
2-17	Adding JARS or Libraries to the Classpath.....	2-17
2-18	Adding a JAR or Library to the Project Classpath	2-18
2-19	Output for Creating a NamedCache and Storing and Retrieving Values	2-20
2-20	Output from the Sample Reader Program	2-21
2-21	Output from the Sample Reader Program with a Running Cache Server	2-22
2-22	Output from JDeveloper if Storage is Disabled.....	2-23
2-23	Output from Coherence-Based Java Application—No Value for the Key	2-24
2-24	Output from Coherence-Based Java Application—A Data Value for the Key	2-25
3-1	Generate Accessors Dialog Box	3-3
3-2	Generate Constructors Dialog Box	3-4
3-3	Adding Labs and Configuration Files to the Classpath	3-19
3-4	Contacts Example Output Run from JDeveloper.....	3-20
4-1	Output from the Sample Cache Loading Program	4-12
4-2	Output of the QueryExample Class	4-17
4-3	Output from the Aggregators	4-21
5-1	Listener Program Waiting for Events	5-5
5-2	Output from the ObserverExample and ProcessorExample Classes	5-9
6-1	Connecting to the Database	6-2
6-2	Unlocking the Database Account	6-2
6-3	Defining the Database Connection.....	6-3
6-4	Creating EJB Entity Beans.....	6-4
6-5	Specifying the EJB Version	6-5
6-6	Defining the Persistence Unit.....	6-6
6-7	Creating Entity Beans from Table Data	6-7
6-8	Choosing the Database Connection	6-8
6-9	Choosing the Table Data for the Entity Bean.....	6-9
6-10	Choosing General Options for the Entity	6-10
6-11	Specifying the Entity Details	6-11
6-12	Generating EJB Entity Beans—the EJB Log Window	6-11
6-13	Adding JARs and Libraries to the Classpath	6-15
6-14	Results from the RunEmployeeExample Application.....	6-17
7-1	Adding the Oracle JDBC Libraries to the Classpath.....	7-3
7-2	Setting the Cache Configuration File for Runtime Options.....	7-4
7-3	Results from Running the DatabaseCache Application.....	7-16
8-1	Creating a New Machine	8-2
8-2	Summary of Machines.....	8-3
8-3	Adding a Server to a Machine.....	8-4
8-4	Summary of Servers Page.....	8-5

8-5	Selecting the coherence-web-spi.jar File for Deployment.....	8-6
8-6	Choosing the Deployment Servers.....	8-7
8-7	Deployments Window Showing the Deployed Application.....	8-9
8-8	Counter Page with Counter Set to 1.....	8-9
8-9	Counter Page with Counter Set to 4.....	8-10

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.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at

<http://www.fcc.gov/cgb/consumerfacts/trs.html>, and a list of phone numbers is available at <http://www.fcc.gov/cgb/dro/trsphonebk.html>.

Related Documents

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

- *Getting Started with Oracle Coherence*
- *Developer's Guide for Oracle Coherence*
- *Client Guide for Oracle Coherence*
- *User's Guide for Oracle Coherence*Web*
- *Integration 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.
<i>italic</i>	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.

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 Release 3.5. 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.

This chapter also assumes that you are running one of the following Microsoft Windows operating systems: XP, Vista, 2000, 2003, or 2008.

Downloading and Installing Coherence

To download and install Oracle Coherence:

1. Download Oracle Coherence to your desktop.

Coherence (Java edition) ships as a single zip file, typically called `coherence-<version>.zip`. You can obtain Coherence 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 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. This ensures 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 must reconfigure your network and firewall settings.

This exercise assumes that you have installed Oracle Coherence (Java Edition) Release 3.5 (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 operates 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 information 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 the Java JDK (for example, \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
```

- 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 produces output similar to the text 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 release number: `Oracle Coherence Version 3.5/453`
- the multicast address. This address changes with each Coherence version. Note the 3.5.0 in the address for Coherence Release 3.5: `Group{Address=224.3.5.0, Port=35450, 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)

2009-04-20 17:49:40.179/0.703 Oracle Coherence 3.5/453 (Pre-release) <Info> (thread=main,
member=n/a): Loaded operational configuration from resource
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2009-04-20 17:49:40.195/0.719 Oracle Coherence 3.5/453 (Pre-release) <Info> (thread=main,
member=n/a): Loaded operational overrides from resource "jar
:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2009-04-20 17:49:40.195/0.719 Oracle Coherence 3.5/453 (Pre-release) <D5> (thread=main,
member=n/a): Optional configuration override "/tangosol-coherence-override.xml" is not specified
2009-04-20 17:49:40.210/0.734 Oracle Coherence 3.5/453 (Pre-release) <D5> (thread=main,
member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.5/453 (Pre-release)
Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.

2009-04-20 17:49:40.898/1.422 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=main,
member=n/a): Loaded cache configuration from resource
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2009-04-20 17:49:42.054/2.578 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Service Cluster joined the cluster with senior service member n/a
```



```

2009-04-20 17:49:45.304/5.828 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=Cluster,
member=n/a): Created a new cluster "cluster:0x29DB" with Member(Id=1, Timestamp=2009-04-20
17:49:41.648, Address=130.35.99.50:8088, MachineId=49714,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:2208, Role=CoherenceServer, Edition=Grid
Edition, Mode=Development, CpuCount=1, SocketCount=1) UID=0x8223633200000120C6265310C2321F98
2009-04-20 17:49:45.335/5.859 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=Invocation:Management, member=1): Service Management joined the cluster with senior service
member 1
2009-04-20 17:49:45.898/6.422 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=DistributedCache, member=1): Service DistributedCache joined the cluster with senior
service member 1
2009-04-20 17:49:46.007/6.531 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=ReplicatedCache, member=1): Service ReplicatedCache joined the cluster with senior service
member 1
2009-04-20 17:49:46.023/6.547 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=OptimisticCache, member=1): Service OptimisticCache joined the cluster with senior service
member 1
2009-04-20 17:49:46.023/6.547 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=Invocation:InvocationService, member=1): Service InvocationService joined the cluster with
senior service member 1
2009-04-20 17:49:46.023/6.547 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=main,
member=1): Started DefaultCacheServer...

```

SafeCluster: Name=cluster:0x29DB

Group{Address=224.3.5.0, Port=35453, TTL=4}

MasterMemberSet

```

(
  ThisMember=Member(Id=1, Timestamp=2009-04-20 17:49:41.648, Address=130.35.99.50:8088,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:2208,
Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2009-04-20 17:49:41.648, Address=130.35.99.50:8088,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:2208,
Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2009-04-20 17:49:41.648, Address=130.35.99.50:8088, MachineId=49714,
Location=site:us.oracle.com,machine:tpfaeffl-pc,proces
s:2208, Role=CoherenceServer)
  )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
  )
)

```

Services

```

(
  TcpRing{TcpSocketAcceptor{State=STATE_OPEN, ServerSocket=130.35.99.50:8088}, Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.4,
OldestMemberId=1}
  InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
  DistributedCache{Name=DistributedCache, State=(SERVICE_STARTED), LocalStorage=enabled,
PartitionCount=257, BackupCount=1, AssignedPartitions=257, BackupPartitions=0}
  ReplicatedCache{Name=ReplicatedCache, State=(SERVICE_STARTED), Id=3, Version=3.0,
OldestMemberId=1}
  Optimistic{Name=OptimisticCache, State=(SERVICE_STARTED), Id=4, Version=3.0, OldestMemberId=1}
  InvocationService{Name=InvocationService, State=(SERVICE_STARTED), Id=5, Version=3.1,
OldestMemberId=1}
)

```

Note: By default, Coherence is configured to use multicast to attempt to join a cluster and to distribute cluster events. Multicast can also be used to distribute a message efficiently to more than one other node of the cluster. Coherence can be configured so that it does not use multicast.

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 start 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

```

C:\oracle\product\coherence\bin>coherence.cmd
** 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)

2009-04-20 18:01:27.632/0.750 Oracle Coherence 3.5/453 (Pre-release) <Info> (thread=main,
member=n/a): Loaded operational configuration from resource
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2009-04-20 18:01:27.648/0.766 Oracle Coherence 3.5/453 (Pre-release) <Info> (thread=main,
member=n/a): Loaded operational overrides from resource
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2009-04-20 18:01:27.648/0.766 Oracle Coherence 3.5/453 (Pre-release) <D5> (thread=main,
member=n/a): Optional configuration override "/tangosol-coherence-override.xml" is not specified
2009-04-20 18:01:27.663/0.781 Oracle Coherence 3.5/453 (Pre-release) <D5> (thread=main,
member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.5/453 (Pre-release)

```

Testing a Coherence Installation

Grid Edition: Development mode

Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.

```
2009-04-20 18:01:29.429/2.547 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Service Cluster joined the cluster with senior service member n/a
2009-04-20 18:01:29.538/2.656 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=Cluster,
member=n/a): Failed to satisfy the variance: allowed=16, actual=47
2009-04-20 18:01:29.538/2.656 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=Cluster,
member=n/a): Increasing allowable variance to 19
2009-04-20 18:01:29.882/3.000 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=Cluster,
member=n/a): This Member(Id=2, Timestamp=2009-04-20 18:01:29.64, Address=130.35.99.50:8089,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:1760,
Role=CoherenceConsole, Edition=Grid Edition, Mode=Development, CpuCount=1, SocketCount=1) joined
cluster "cluster:0x29DB" with senior Member(Id=1, Timestamp=2009-04-20 17:49:41.648,
Address=130.35.99.50:8088, MachineId=49714,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:2208, Role=CoherenceServer, Edition=Grid
Edition, Mode=Development, CpuCount=1, SocketCount=1)
2009-04-20 18:01:29.913/3.031 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Member 1 joined Service Management with senior member 1
2009-04-20 18:01:29.913/3.031 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Member 1 joined Service DistributedCache with senior member 1
2009-04-20 18:01:29.913/3.031 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Member 1 joined Service ReplicatedCache with senior member 1
2009-04-20 18:01:29.913/3.031 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Member 1 joined Service OptimisticCache with senior member 1
2009-04-20 18:01:29.913/3.031 Oracle Coherence GE 3.5/453 (Pre-release) <D5> (thread=Cluster,
member=n/a): Member 1 joined Service InvocationService with senior member 1
2009-04-20 18:01:30.101/3.219 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=Invocation:Management, member=2): Service Management joined the cluster with senior service
member 1 SafeCluster: Name=cluster:0x29DB
```

```
Group{Address=224.3.5.0, Port=35453, TTL=4}
```

```
MasterMemberSet
```

```
(
  ThisMember=Member(Id=2, Timestamp=2009-04-20 18:01:29.64, Address=130.35.99.50:8089,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:1760,
Role=CoherenceConsole)
  OldestMember=Member(Id=1, Timestamp=2009-04-20 17:49:41.648, Address=130.35.99.50:8088,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc,process:2208,
Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2009-04-20 17:49:41.648, Address=130.35.99.50:8088, MachineId=49714,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:2208, Role=CoherenceServer)
    Member(Id=2, Timestamp=2009-04-20 18:01:29.64, Address=130.35.99.50:8089, MachineId=49714,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:1760, Role=CoherenceConsole)
  )
  RecycleMillis=120000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
  )
)
```

```
Services
(
  TcpRing{TcpSocketAcceptor{State=STATE_OPEN, ServerSocket=130.35.99.50:8089}, Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.4,
OldestMemberId=1}
  InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
)
```

```
Map (?):
```

```
2009-04-20 18:01:30.929/4.047 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=TcpRingListener, member=2): TcpRing: connecting to member 1 using
  TcpSocket{State=STATE_OPEN, Socket=Socket[addr=/130.35.99.50,port=3609,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
2009-04-20 18:08:16.773/409.891 Oracle Coherence GE 3.5/453 (Pre-release) <Info> (thread=main,
member=2): Loaded cache configuration from resource
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2009-04-20 18:08:17.132/410.250 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=DistributedCache, member=2): Service DistributedCache joined the cluster with senior
service member 1
2009-04-20 18:08:17.163/410.281 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=DistributedCache, member=2): Service DistributedCache: received ServiceConfigSync
containing 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 (myCache)` prompt in the Coherence shell. For definitions of these commands, see [Appendix A, "Coherence Client Application Commands."](#)

- `get message`
- `put message "hello"`

- list
- size
- get message
- put message "second message"
- get message
- remove message
- 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
Map (mycache): bye2009-04-20 18:16:35.898/909.016 Oracle Coherence GE 3.5/453
(Pre-release) <D5> (thread=Invocation:Management, member=2): Service Management
left the cluster
2009-04-20 18:16:35.913/909.031 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=DistributedCache, member=2): Service DistributedCache left the cluster
2009-04-20 18:16:36.007/909.125 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
```

```
(thread=Cluster, 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 the terminal window displays a message similar to the following that describes where the first client is running:

```
Map (mycache):
2009-04-20 18:18:15.468/66.407 Oracle Coherence GE 3.5/453 (Pre-release)
(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. Terminate one of the `coherence.cmd` shells (`bye`). Note that the other shell displays a message indicating that the member has left the cluster. For example:

```
Map (mycache):
2009-04-20 18:18:50.835/878.204 Oracle Coherence GE 3.5/453 (Pre-release) <D5>
(thread=Cluster, member=3): Member 4 left service DistributedCache with senior
member 1
```

```
Map (mycache):
2009-04-20 18:18:50.835/878.204 Oracle Coherence GE 3.5/453 (Pre-release)<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)
2009-04-20 18:19:50.835/878.204 Oracle Coherence GE 3.5/453 (Pre-release) <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 terminate 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 start another Coherence cache server, and then terminate the initial one that was started (using `Ctrl+C` or by closing the command window), note that the data is still available.
14. Terminate both the `coherence.cmd` shells and `cache-server.cmd`. Start `coherence.cmd`. Create a `NamedCache` called `test` using `cachetest`. Try to put in a value as before and note the results.
15. Start a cache server by running the `cache-server.cmd` file from the `coherence\bin` directory. Try to put in a value again and note the results.
16. Terminate 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 the default behavior for Coherence—to cluster with other processes running Coherence locally or on a network—it is strongly advised, while you perform this tutorial, 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 might 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 run 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.

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](#)
- [JDeveloper Basics](#)
- [Accessing the Data Grid from Java](#)
- [Creating Your First Coherence-Based Java Application](#)

Configuring Oracle JDeveloper for Coherence

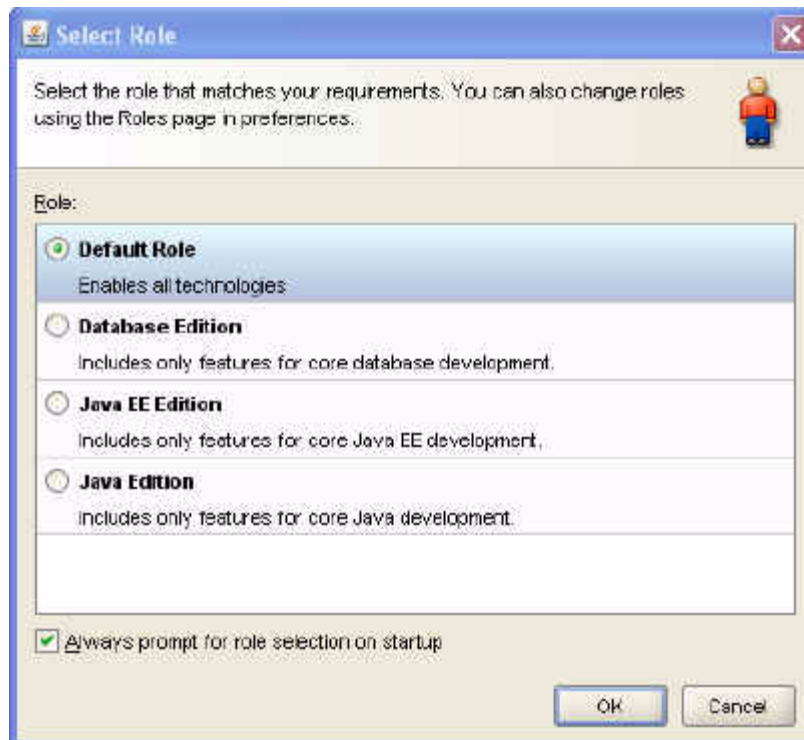
To start and configure JDeveloper for use with Coherence:

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.

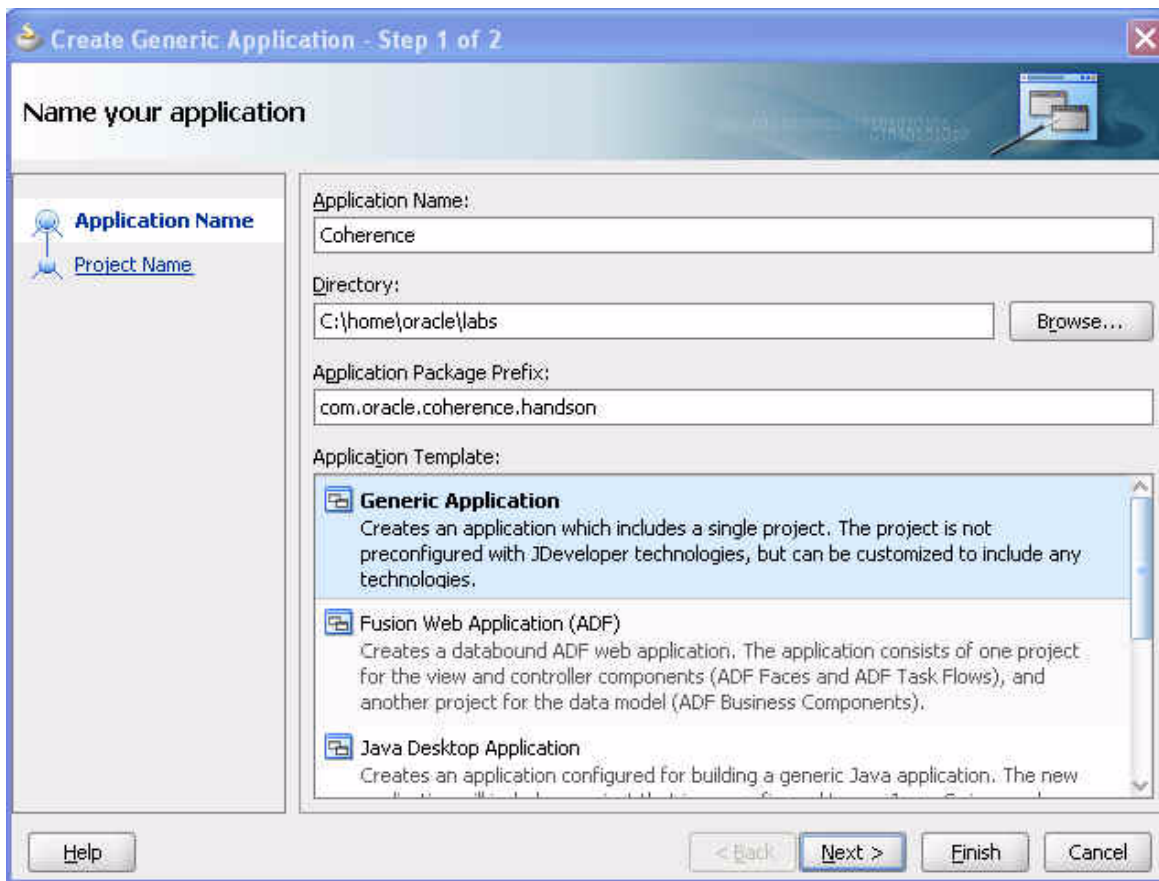
```
set PATH=%JAVA_HOME%\bin;%PATH%
```
 - c. Start JDeveloper. If you get a message asking you to select a role, select **Default Role**. If you get a message asking whether you want to migrate from previous versions of JDeveloper, select **No**. Close any window that displays the **Tip of the Day**.

Figure 2–1 Select Role Dialog Box



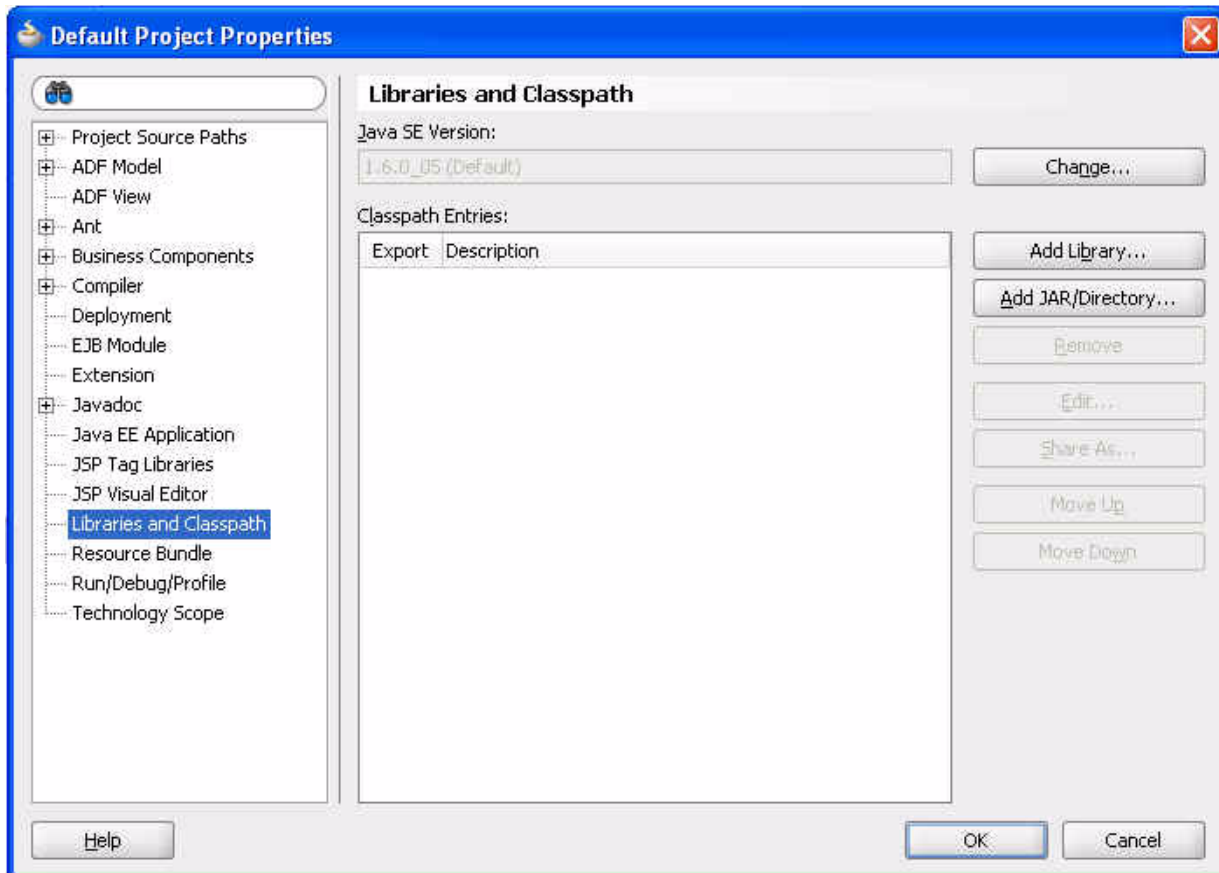
2. After JDeveloper starts, click **New Application** to create an application.
An application is a way of grouping projects or source code. This new application will hold the Coherence projects.
3. In the dialog box, change the application name to Coherence, the directory to \home\oracle\labs, and the 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

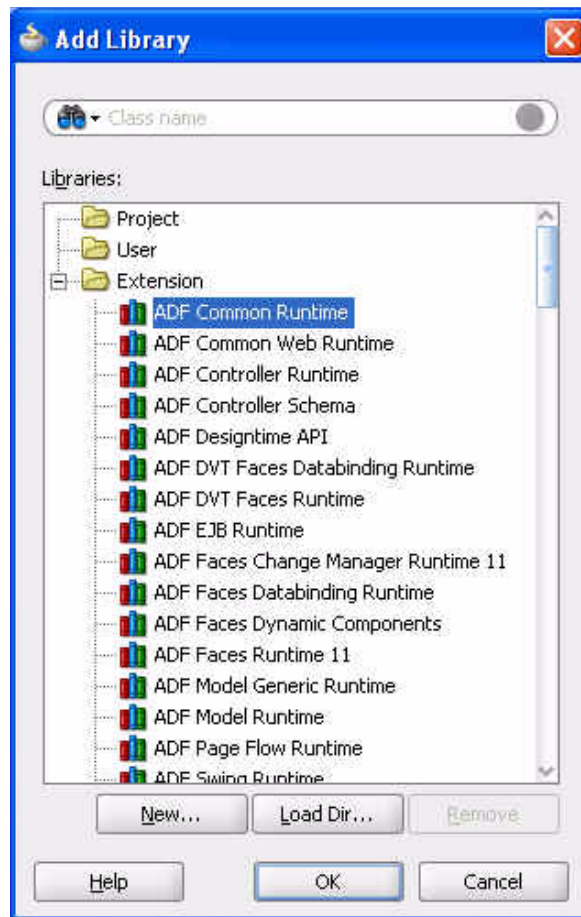


4. Add the Coherence JAR files to the default project properties. From the **Tools** menu, select **Default Project Properties**. The **Default Properties** dialog box opens.

Figure 2-3 Default Properties Dialog Box

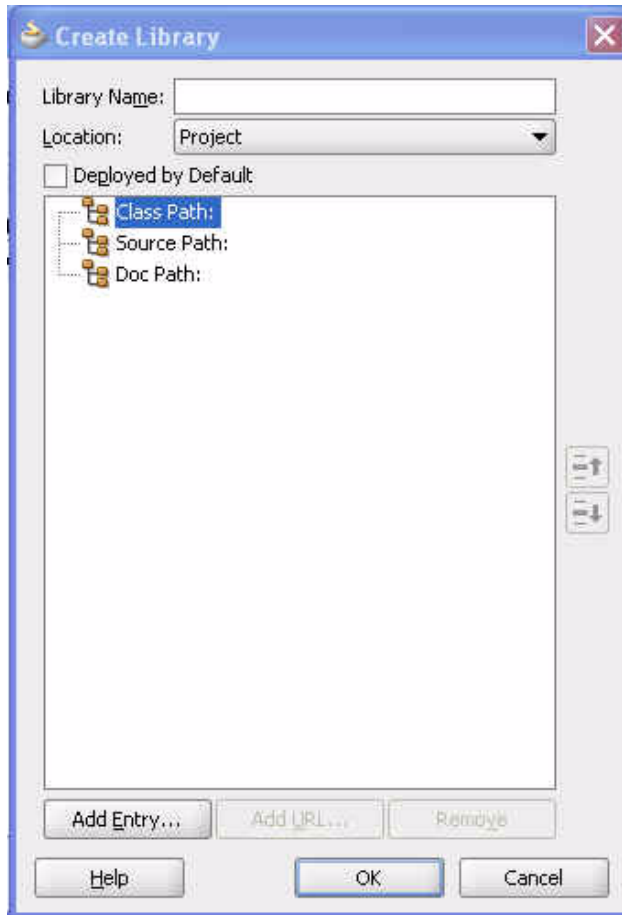


5. Click **Libraries and Classpath**, and click the **Add Library** button. The **Add Library** dialog box opens.

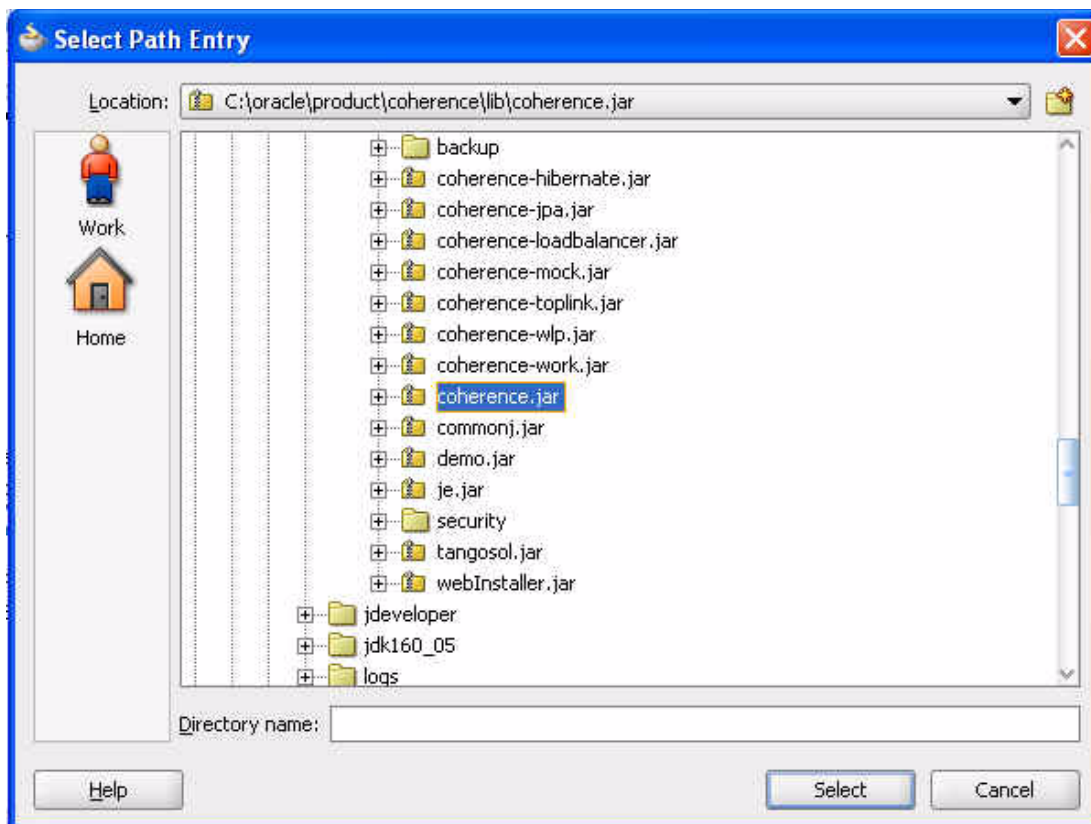
Figure 2–4 Add Library Dialog Box

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

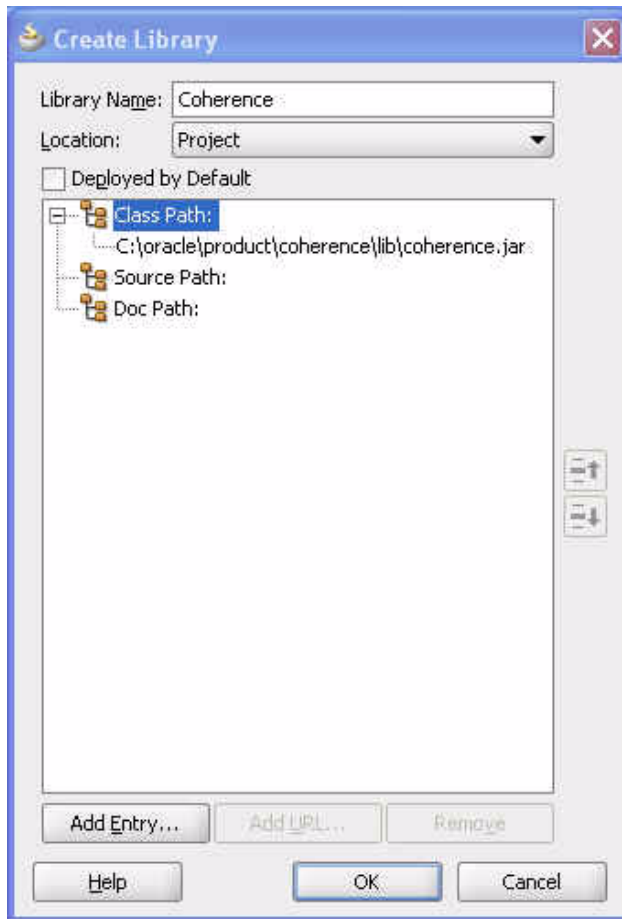
Figure 2-5 Create Library Dialog Box



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.

JDeveloper Basics

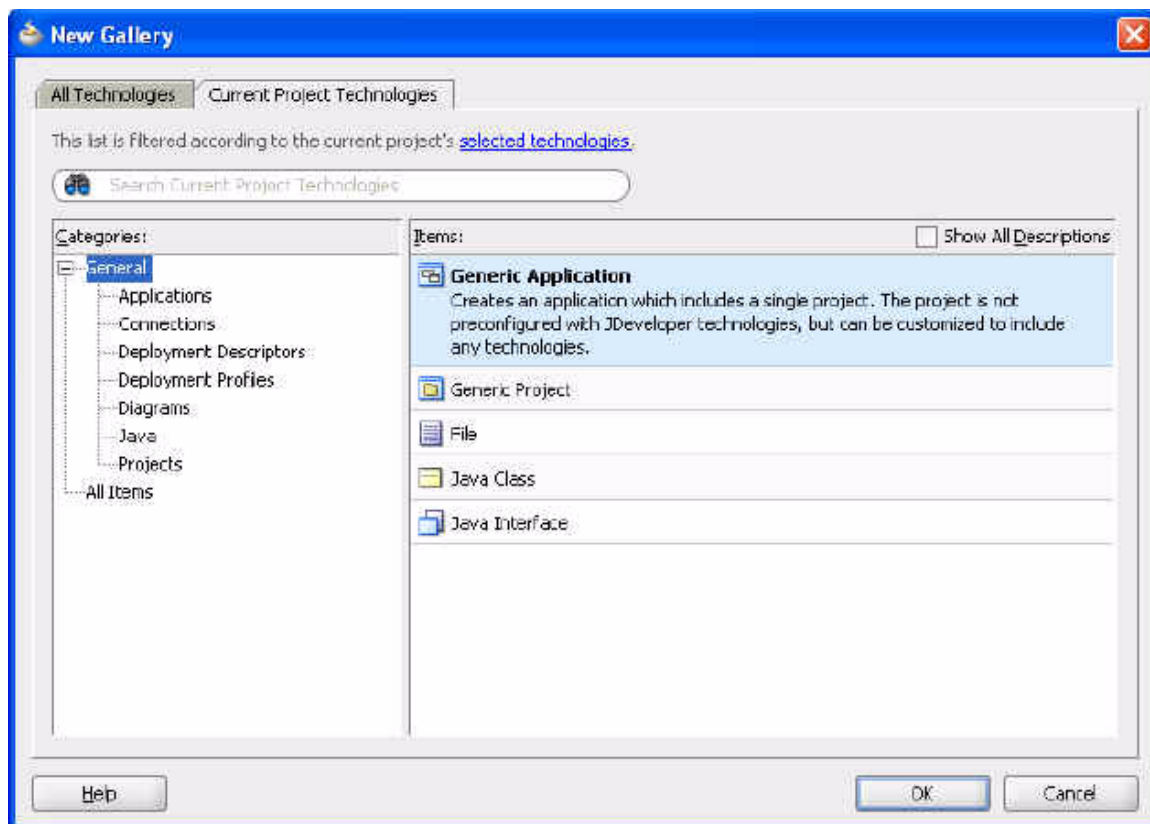
This section describes some of the common tasks that are a part of building projects in JDeveloper.

- [Creating a New Application and Project](#)
- [Creating a Java Class](#)
- [Changing Project Properties, Setting Runtime Configuration, and](#)

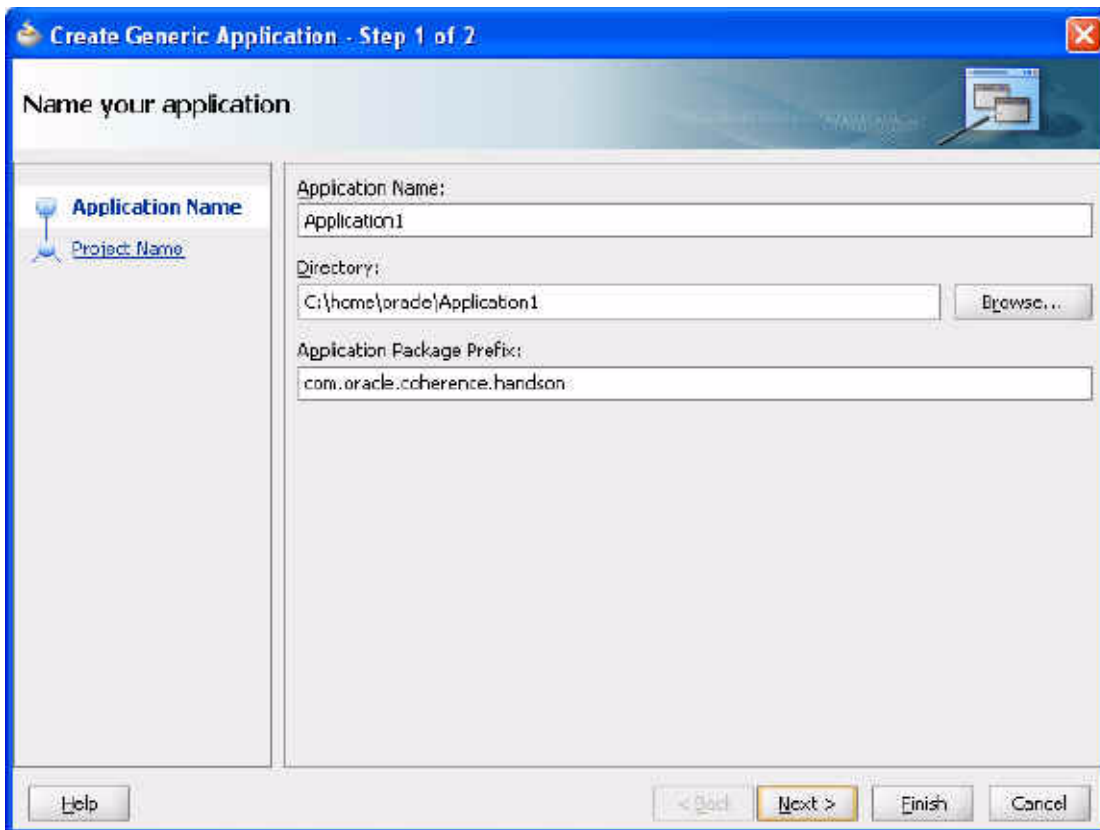
Creating a New Application and Project

To create a new application in JDeveloper:

1. Choose **File > New...** in the JDeveloper IDE. This opens the **New Gallery**.
2. Choose **General > Projects** in the **Categories** section of the **New Gallery** dialog box and **Generic Project** in the **Items** section. Click **OK**.

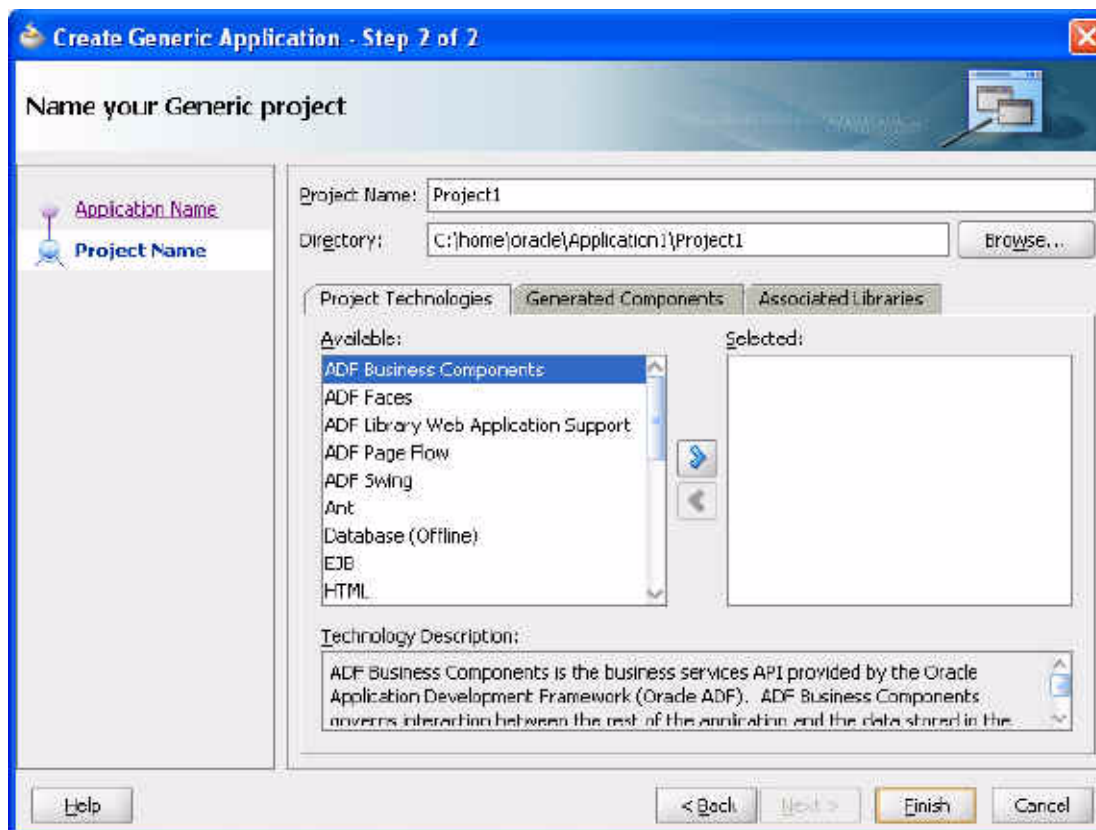
Figure 2–8 *Creating a New Application in the New Gallery*

3. In the **Create Generic Application** dialog box, enter the name the project.
 - Replace **Application1** with the name of your application.
 - Ensure that directory path to the project is correct.
 - Click **Next**.

Figure 2–9 Providing an Application Name

4. JDeveloper asks you to create a new project within the application. Enter a name for the project. If necessary, select a project technology from the **Available** list. Click **Finish** to create the new project.

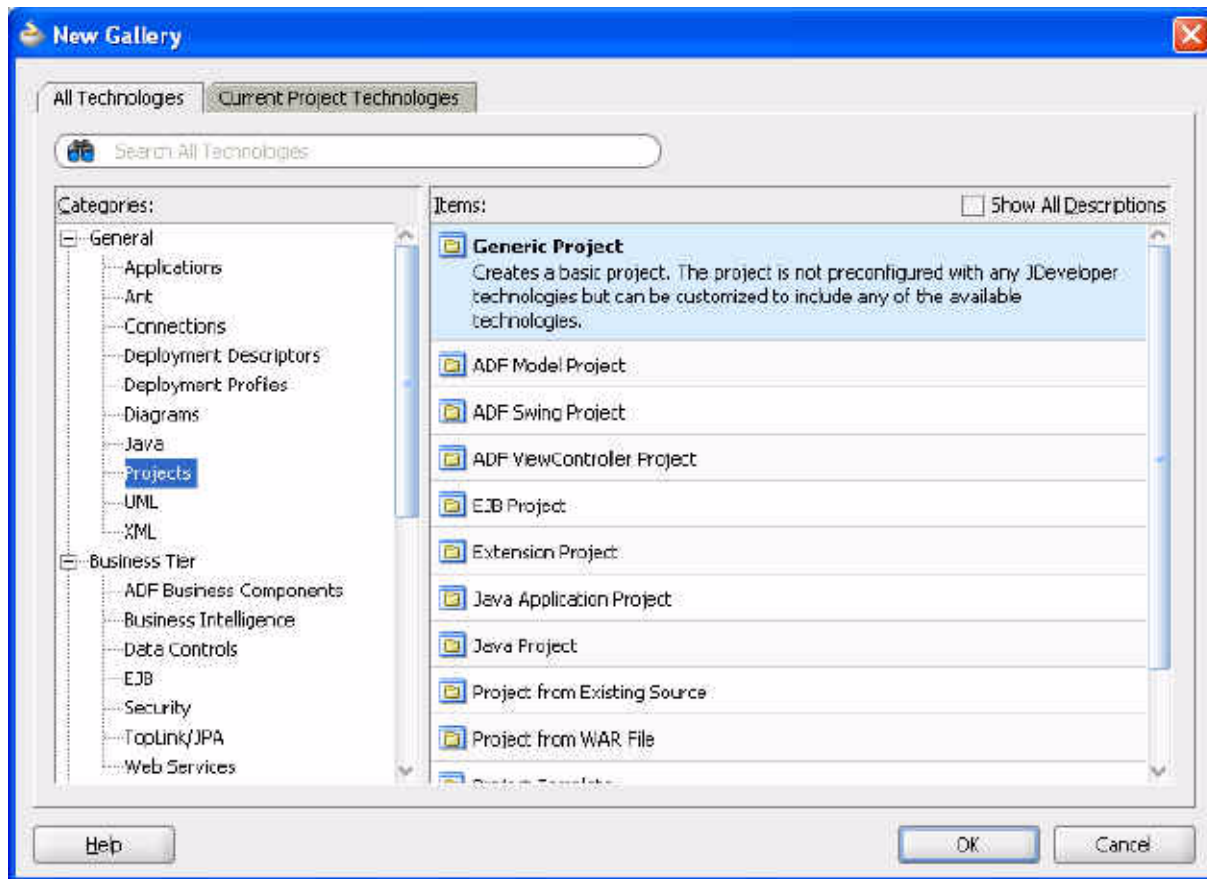
Figure 2–10 Providing a Project Name



Creating a New Project in an Existing Application

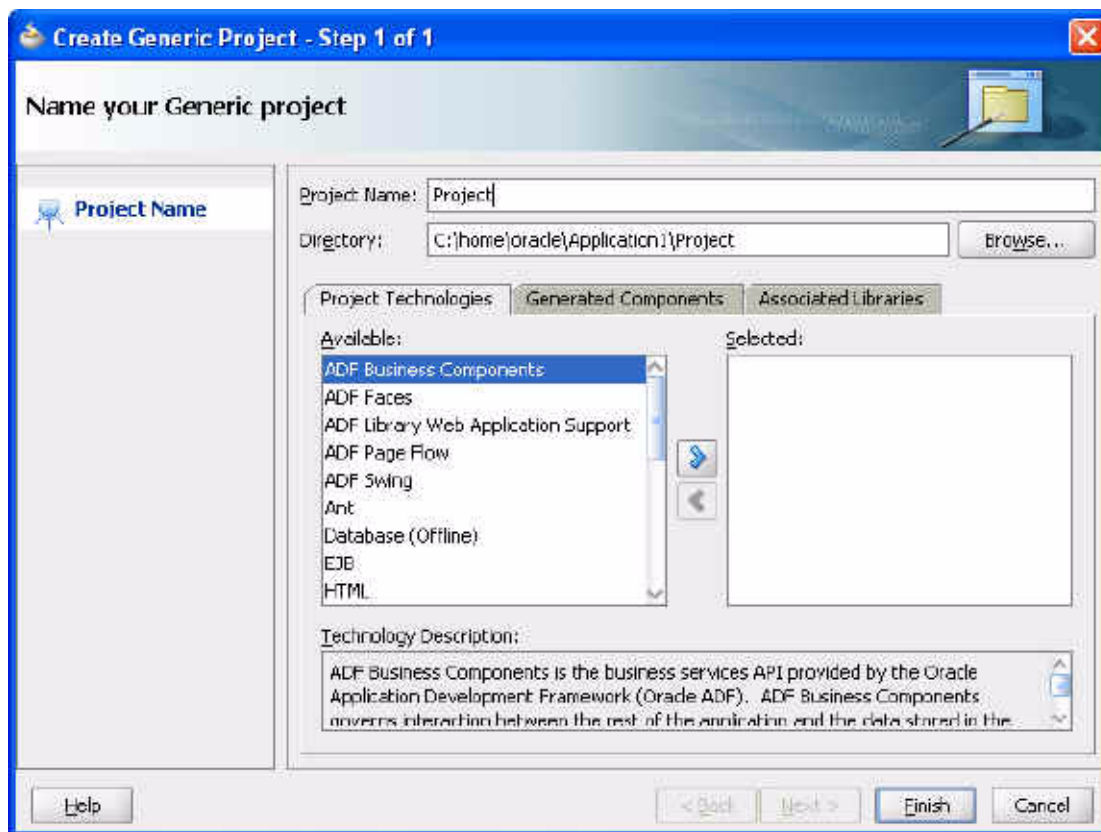
To create a project in an existing application:

1. Right-click the Coherence application and choose **New...** In the **New Gallery** choose **Projects** under **Categories** and **Generic Project** under **Items**. Click **OK**.

Figure 2–11 Creating a New Project in an Existing Application

2. Enter a **Project Name**, ensure that the **Default Package** is set correctly, and select a **Project Technology** if necessary. Click **Finish**.

Figure 2–12 Providing Details for the New Project

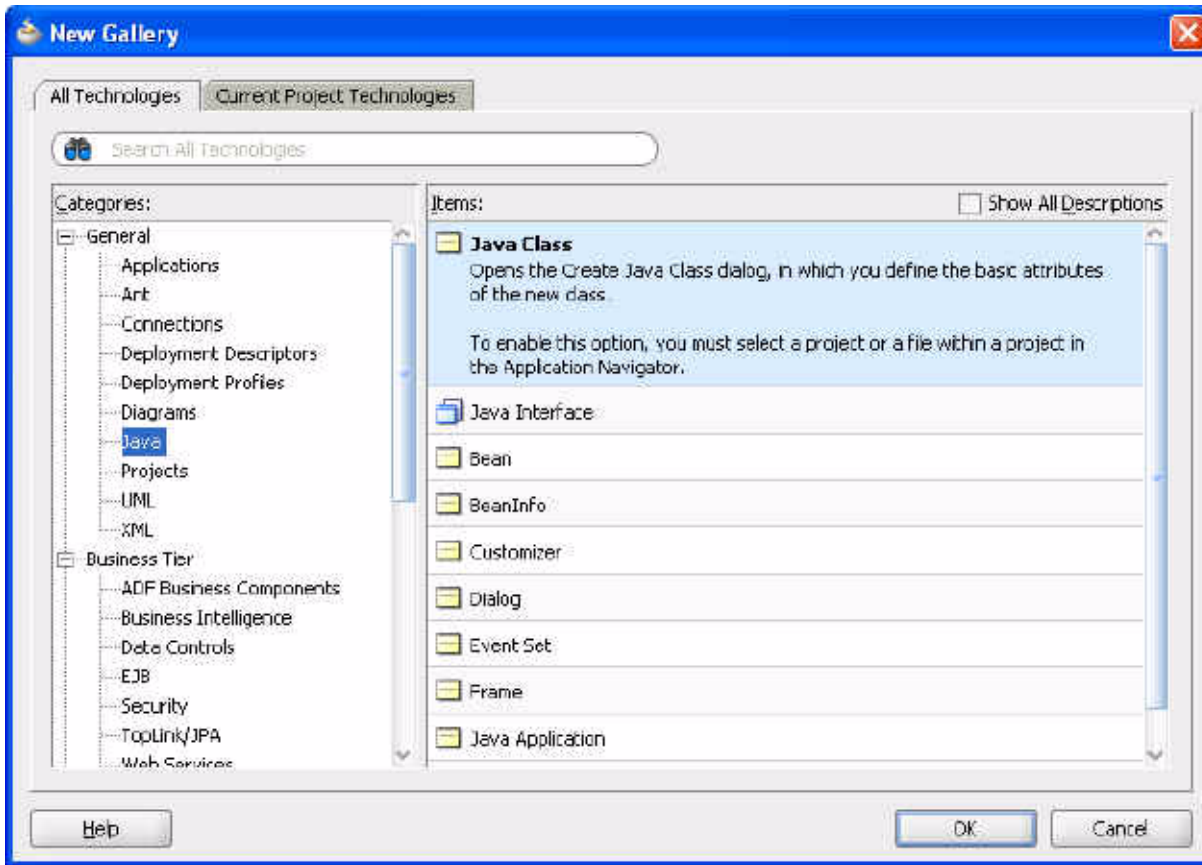


Creating a Java Class

To create a new Java class in JDeveloper:

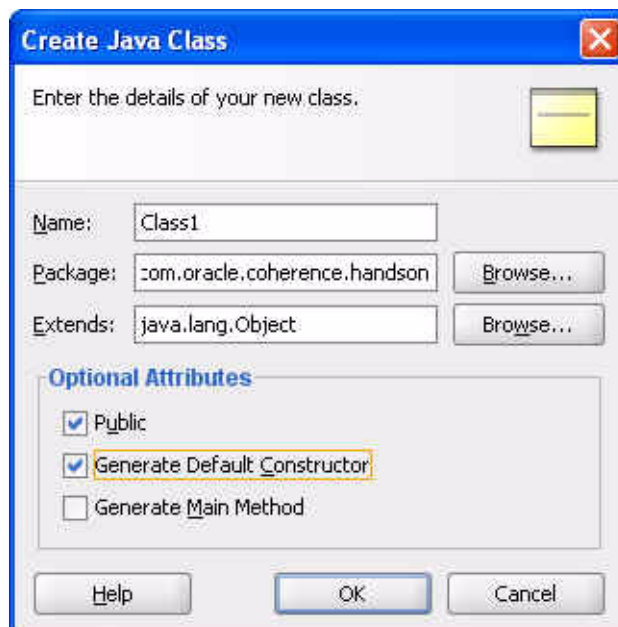
1. Right-click the project entry in the Navigator pane and select New.
2. Select **Java** under the **General** category and select **Java Class**. Click **OK**.

Figure 2–13 *Creating a Java Class in the New Gallery*



3. Replace **Class1** with the name of your class. Select the **Generate Main Method** check box if the file must be runnable. Click **OK** to create the Java class.

Figure 2–14 *Providing Details for the Java Class*

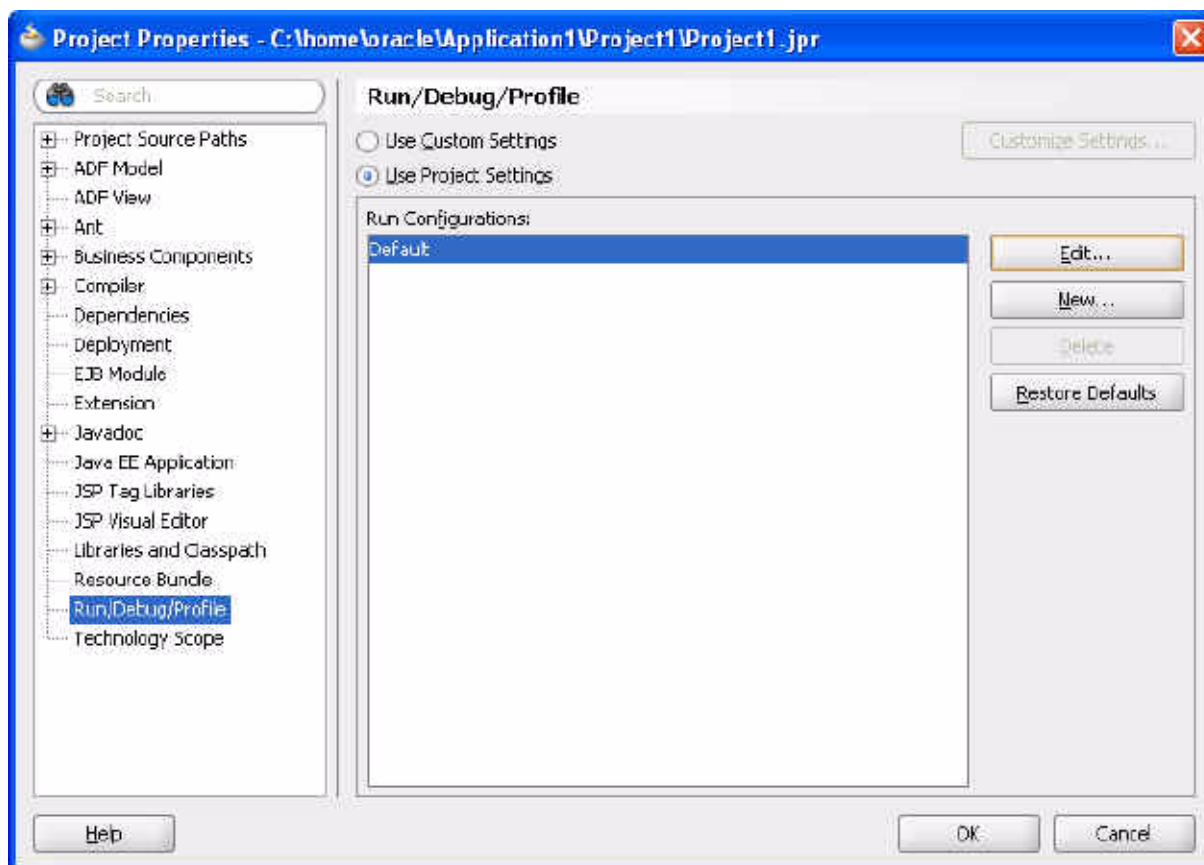


Changing Project Properties, Setting Runtime Configuration, and

To change the project's runtime properties in JDeveloper:

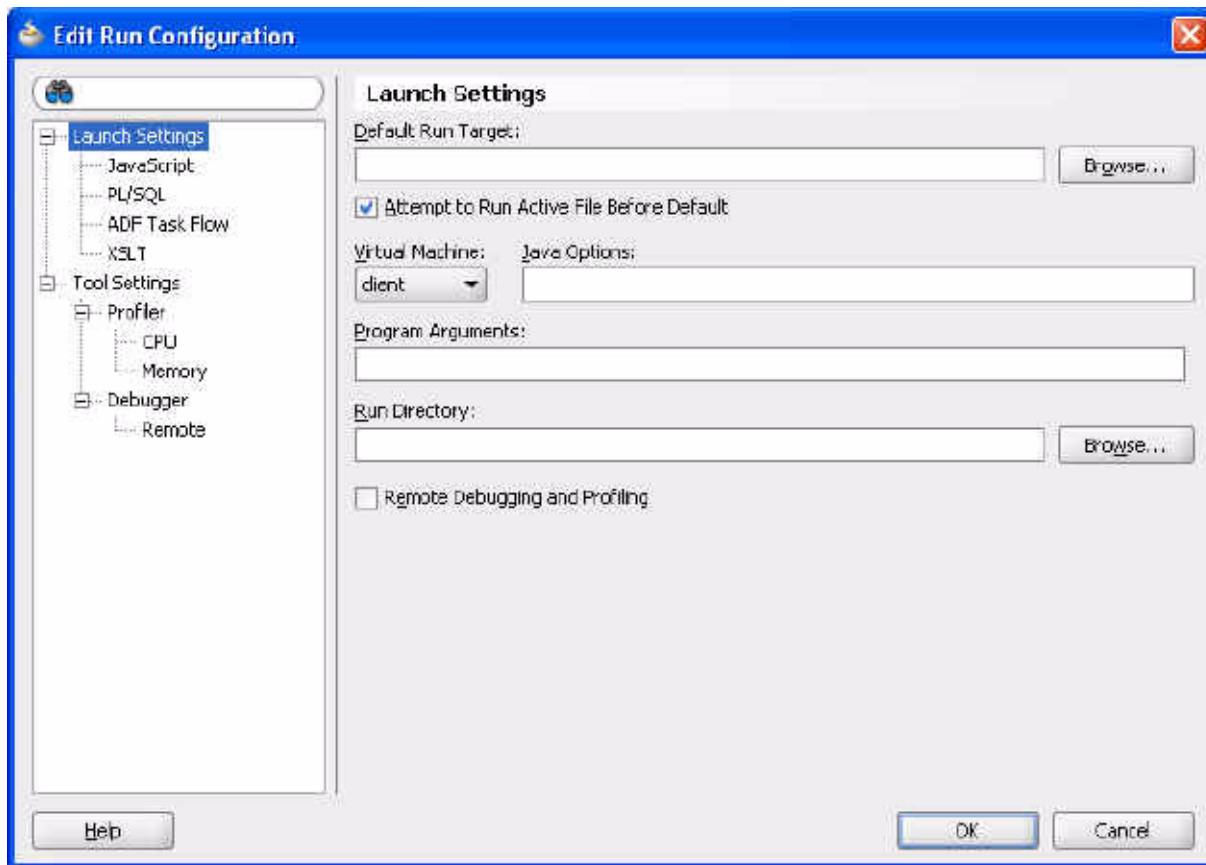
1. Right click the project and select **Project Properties**.
2. In the **Project Properties** dialog box select **Run/Debug/Profile** and click **Edit**.

Figure 2–15 Project Properties Dialog Box



3. You can set a number of values in the Edit Run Configuration dialog box:
 - Set **Virtual Machine** to either **Client** or **Server**.
 - Set values for local storage and log level in the **Java Options** field. For example:


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

Figure 2–16 Setting the Runtime Configuration

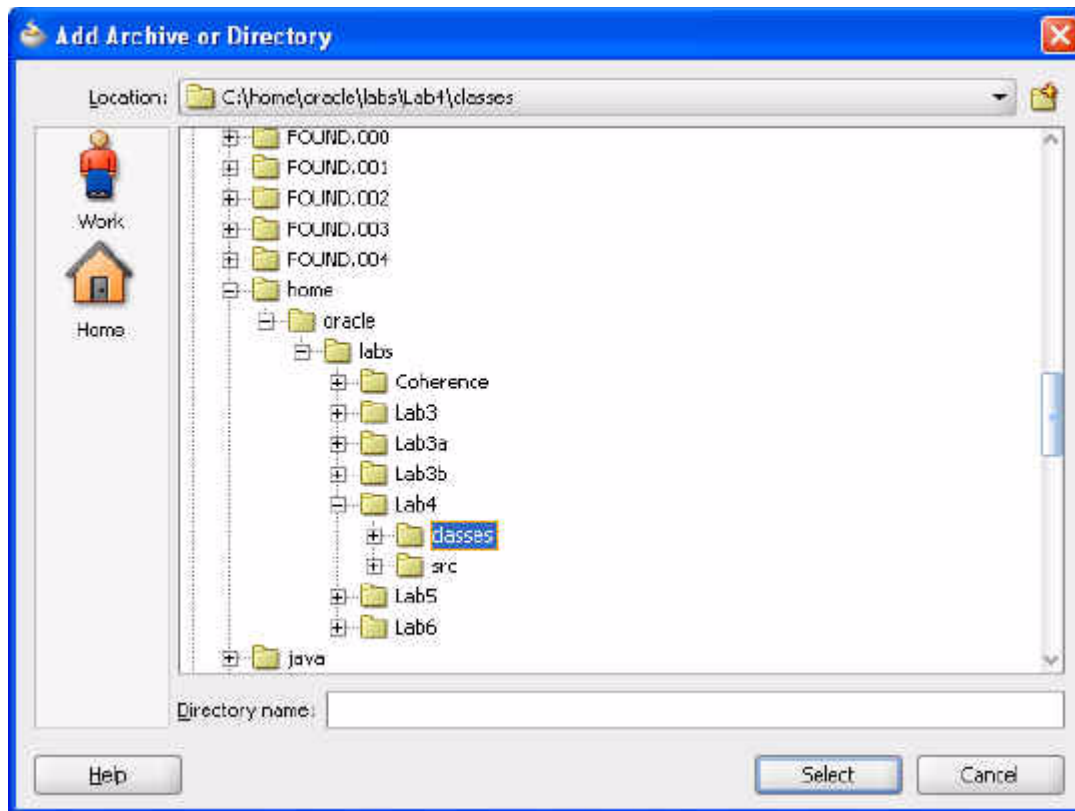
4. Click **OK** in the **Edit Run Configuration** dialog box and **OK** in the **Project Properties** dialog box to return to the JDeveloper IDE.

Adding JARS and Libraries to the Project Classpath

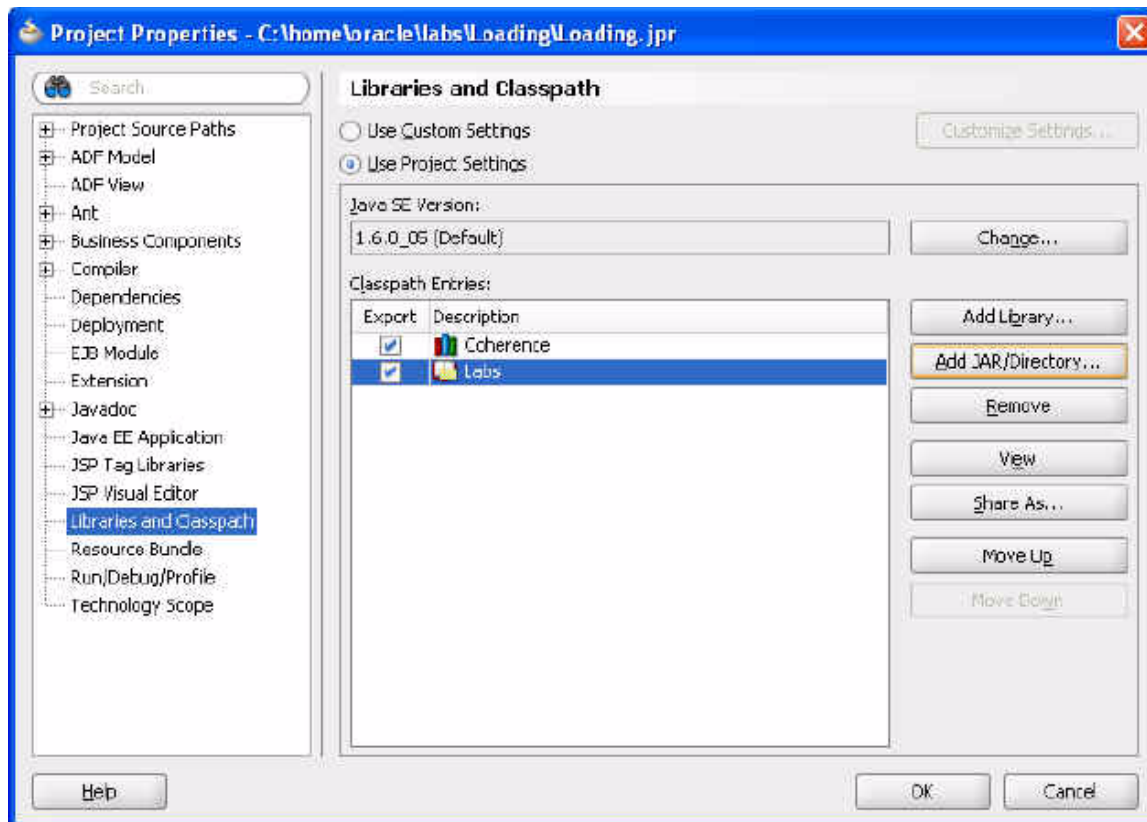
To add JAR files or libraries to the project classpath:

1. Right click the project and select **Project Properties**.
2. In the **Project Properties** dialog box select **Libraries and Classpaths** and click **Add JAR/Directory**.
3. Navigate to the JAR file or directory that you want to include in the classpath.

Figure 2–17 Adding JARS or Libraries to the Classpath



4. Click **Select** in the **Add Archive or Directory** dialog box. Click **OK** in the **Project Properties** dialog box.

Figure 2–18 Adding a JAR or Library to the Project Classpath

Accessing the Data Grid from Java

In this exercise, you will 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 exercise, 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
<code>void clear()</code>	Removes all entries from the <code>NamedCache</code> .
<code>boolean containsKey(Object key)</code>	Returns <code>true</code> if <code>NamedCache</code> contains an entry for the key.
<code>boolean containsValue(Object value)</code>	Returns <code>true</code> if there is at least one entry with this value in <code>NamedCache</code> .
<code>Object get(Object key)</code>	Gets the entry from <code>NamedCache</code> for that key.
<code>Object put(Object key, Object value)</code>	Puts an object in the cache and returns the previous value (if any).
<code>Object remove(Object key)</code>	Removes the mapping for this key from this map if present. Inherited from <code>ConcurrentMap</code> .
<code>Set entrySet()</code>	Returns a set of key/value pairs.
<code>Collection values()</code>	Gets all values back as a collection.
<code>CacheService getCacheService()</code>	Returns the <code>CacheService</code> that this <code>NamedCache</code> 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
<code>static Cluster ensureCluster()</code>	Obtains a cluster object running Coherence services.
<code>static void shutdown()</code>	Shuts down all clustered services.
<code>static NamedCache getCache(String cache)</code>	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.

See ["Creating a New Project in an Existing Application"](#) on page 2-11 if you need detailed instructions. Name the project `InsertValue`. Ensure that directory is `C:\home\oracle\labs\InsertValue`.

Check the **Project Properties>Libraries and Classpath** value for the Coherence entry. Ensure that the full path is provided for the `coherence.jar`: `C:\oracle\product\coherence\lib\coherence.jar`

2. Create your first Coherence Java program.

See ["Creating a Java Class"](#) on page 2-13 if you need detailed information. Name the class `MyFirstSample` and select the **Generate Main Method** check box.

- a. In the JDeveloper editor, 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) {

        // 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"));
    }
}
```

- b. Run the program in JDeveloper: right-click the `MyFirstSample.java` class in the editor and choose **Run**.
- c. You should see messages similar to the following:

Figure 2–19 Output for Creating a `NamedCache` and Storing and Retrieving Values

```
C:\oracle\product\jdk160_05\bin\javaw.exe -server -classpath
2009-04-20 18:29:00.757/0.937 Oracle Coherence 3.5/453 (Pr
2009-04-20 18:29:00.773/0.953 Oracle Coherence 3.5/453 (Pr

Oracle Coherence Version 3.5/453 (Pre-release)
  Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All
rights reserved.

2009-04-20 18:29:03.304/3.484 Oracle Coherence GE 3.5/453
2009-04-20 18:29:03.866/4.046 Oracle Coherence GE 3.5/453
Value in cache is Gene Smith
Process exited with exit code 0.
```

3. Now create another Java class which gets the value from your cache, instead of doing a put and then a get. Name this Java class `MyFirstSampleReader`. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information. [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 cluster
        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 2–20](#) illustrates the output from the program. Note that a null value is returned. Although `MyFirstSample` successfully created and populated the `NamedCache`, the cache only existed within the `MyFirstSample` process memory. When `MyFirstSample` terminated so did the cache.

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 2–20 Output from the Sample Reader Program

```

C:\oracle\product\jdk160_05\bin\javaw.exe -server -classpath
2009-04-20 18:37:36.335/0.937 Oracle Coherence 3.5/453 (Pre
2009-04-20 18:37:36.351/0.953 Oracle Coherence 3.5/453 (Pre

Oracle Coherence Version 3.5/453 (Pre-release)
  Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All

2009-04-20 18:37:41.804/6.406 Oracle Coherence CE 3.5/453
2009-04-20 18:37:42.195/6.797 Oracle Coherence CE 3.5/453
Value in cache is null
Process exited with exit code 0.

```

4. Start 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 2–21](#): the "Gene Smith" value stored by `MyFirstSample` is returned by `MyFirstSampleReader`. By default, unless specified otherwise, all processes start as storage-enabled. This means that the process can store data as part of the cluster.

Figure 2–21 Output from the Sample Reader Program with a Running Cache Server

```

C:\oracle\product\jdk160_05\bin\javaw.exe -server -classpath
2009-04-20 18:34:42.460/0.937 Oracle Coherence 3.5/453 (Pre
2009-04-20 18:34:42.476/0.953 Oracle Coherence 3.5/453 (Pre

Oracle Coherence Version 3.5/453 (Pre-release)
  Grid Edition: Development mode
  Copyright (c) 2000, 2009, Oracle and/or its affiliates. All

2009-04-20 18:34:44.945/3.422 Oracle Coherence GE 3.5/453
2009-04-20 18:34:45.460/3.937 Oracle Coherence GE 3.5/453
Value in cache is Gene Smith
Process exited with exit code 0.

```

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 is not 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. Change the project properties to disable local storage. See "[Changing Project Properties, Setting Runtime Configuration, and](#)" on page 2-15 for detailed information.
- b. Ensure the **Virtual Machine** field is set to server. Enter the following value in the **Java Options** field:

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

The Java option `tangosol.coherence.log.level=level` changes the level of logging produced by Coherence. The *level* can be set to a number between 0 and 9. The default is 5. A value of 0 means no logging. A value of 9 means extremely verbose. A value of 3 is often useful enough for most application development.

5. Shut down any running cache servers and rerun your `MyFirstSample` class.

You receive a message similar to the one in [Figure 2–22](#) indicating that storage is not enabled on the cluster, because you have set this member to be storage-disabled.

Figure 2–22 Output from JDeveloper if Storage is Disabled

```
Exception in thread "main" java.lang.RuntimeException: Storage is not configured
at com.tangosol.coherence.component.util.daemon.queueProcessor.service.Gr
at com.tangosol.coherence.component.util.daemon.queueProcessor.service.Gr
at com.tangosol.coherence.component.util.daemon.queueProcessor.service.Gr
at com.tangosol.coherence.component.util.daemon.queueProcessor.service.Gr
at com.tangosol.util.ConverterCollections$ConverterMap.put(ConverterColle
at com.tangosol.coherence.component.util.daemon.queueProcessor.service.Gr
at com.tangosol.coherence.component.util.SafeNamedCache.put(SafeNamedCach
at com.oracle.coherence.handson.MyFirstSample.main(MyFirstSample.java:18)
Process exited with exit code 1.
```

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

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

1. Examine the methods in the `CacheFactory` class using the Coherence Java documentation (Javadoc) that is shipped in the `\oracle\product\coherence\doc` directory.
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). See "[Creating a Java Class](#)" on page 2-13 if you need detailed information on creating a Java class.
3. Examine the methods that are available in the `NamedCache` interface using the Javadoc.
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 2-23 illustrates the output from YourFirstCoherenceApplication. The output indicates that there is no data in the cache for the message key.

Figure 2-23 Output from Coherence-Based Java Application—No Value for the Key

```

C:\oracle\product\jdk160_05\bin\javaw.exe -server -classpa
2009-04-20 18:40:32.335/0.765 Oracle Coherence 3.5/453 (Pr
2009-04-20 18:40:32.366/0.796 Oracle Coherence 3.5/453 (Pr

Oracle Coherence Version 3.5/453 (Pre-release)
  Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All
2009-04-20 18:40:34.898/3.328 Oracle Coherence GE 3.5/453
2009-04-20 18:40:35.507/3.937 Oracle Coherence GE 3.5/453
null
Process exited with exit code 0

```

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 2–24](#) illustrates that cache now holds the value `hello` for the key message.

Figure 2–24 Output from Coherence-Based Java Application—A Data Value for the Key

```
C:\oracle\product\jdk160_05\bin\javaw.exe -server -classpath
2009-04-20 18:46:44.116/0.921 Oracle Coherence 3.5/453 (Pre-
2009-04-20 18:46:44.132/0.937 Oracle Coherence 3.5/453 (Pre-

Oracle Coherence Version 3.5/453 (Pre-release)
Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All

2009-04-20 18:46:46.679/3.484 Oracle Coherence GB 3.5/453 (
2009-04-20 18:46:47.257/4.062 Oracle Coherence GB 3.5/453 (
hello
Process exited with exit code 0
```

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
```

Working with Complex Objects

In this chapter, you work with complex objects residing in the cache. Using JDeveloper, you 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](#)
- [Creating and Caching Complex Objects](#)

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 might 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 (deserializing) them into a live object at some future time. For example, objects that implement the `java.io.Serializable` interface are serializable.

As an alternative to using the Java `Serializable` interface, you can improve performance by using Coherence's own class for high-performance serialization, `com.tangosol.io.pof.PortableObject`. `PortableObject` is up to six times faster than the standard `java.io.Serializable` and the serialized result set is smaller.

The `PortableObject` interface provides two simple methods, `readExternal` and `writeExternal`, that permit you 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 dramatically reduces 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.

Creating and Caching Complex Objects

In this exercise, you create a `Contact` object that contains names, addresses, dates of birth, and telephone numbers for employees. An `Address` object type supplies the address. You will also create a program that uses `PortableObject` to put the object in the cache and retrieve it.

Creating the Data Objects

This section describes how to create two data objects that will later be incorporated another data object. The `Address` object will be used to provide employee address information. The `Phone` object will provide telephone contact information.

1. Create an `Address` object to store address information for an employee.
 - a. Create a new project in JDeveloper called `Contacts`. See ["Creating a New Project in an Existing Application"](#) on page 2-11 if you need detailed information.
 - b. Name the project `Contacts`. Ensure that the **Default Package** is `com.oracle.coherence.handson`, the **Java Source Path** is `C:\home\oracle\labs\Contacts\src` and the **Output Directory** is `C:\home\oracle\labs\Contacts\classes`.

Check the **Project Properties>Libraries and Classpath** value for the `Coherence` entry. Ensure that the full path is provided for the `coherence.jar`: `C:\oracle\product\coherence\lib\coherence.jar`
 - c. Create a new Java class called `Address`. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.

Name the Java class `Address`. *Do not* select the **Generate Main Method** check box.
 - d. Write the class to use `PortableObject` for data serialization. In the JDeveloper code editor, change your generated `Address` class to implement `com.tangosol.io.pof.PortableObject`. Add an `import` statement for the `PortableObject` class.
 - e. Import the `com.tangosol.io.pof.PofReader`, `com.tangosol.io.pof.PofWriter`, and `java.io.IOException` classes required by `PortableObject`.
 - f. Add the default public constructor for `Address` that is required by `PortableObject`.
 - g. Enter the following private attributes for your `Address` class. You can add others if you like.

```
— String Street1
— String Street2
— String City
— String State
— String Country
```

At this point, the `Address` class should look similar to the following

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

import com.tangosol.io.pof.PofReader;
```

```

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

import java.io.IOException;

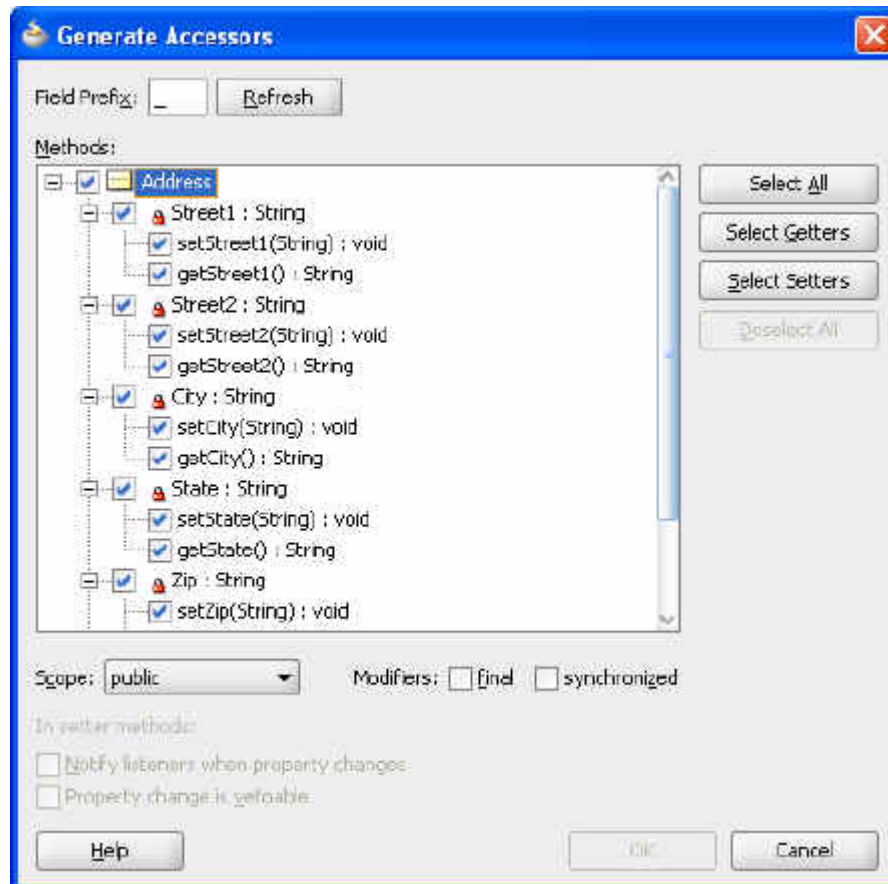
public class Address implements PortableObject
{
    private String Street1;
    private String Street2;
    private String City;
    private String State;
    private String Zip;
    private String Country;

    /**
     * Default constructor (necessary for PortableObject implementation).
     */
    public Address()
    {
    }
}

```

- 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 Address class. All of the attributes are now automatically selected. Click **OK** to continue.

Figure 3–1 Generate Accessors Dialog Box



- 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 3–2 *Generate Constructors Dialog Box*



The generated constructor should look similar to the following:

```
public Address(String Street1, String Street2, String City, String
State, String Zip, String Country) {
    super();
    this.Street1 = Street1;
    this.Street2 = Street2;
    this.City = City;
    this.State = State;
    this.Zip = Zip;
    this.Country = Country;
}
```

- j. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface.
- k. Implement the `equals`, `hashCode`, and `toString` object methods.

Note: Cache keys and values must be serializable (for example, `java.io.Serializable`). Cache keys must also provide an implementation of the `hashCode()` and `equals()` methods, and those methods must return consistent results across cluster nodes. This implies that the implementation of `hashCode()` and `equals()` must be based solely on the object's serializable state (that is, the object's non-transient fields); most built-in Java types, such as `String`, `Integer` and `Date`, meet this requirement. Some cache implementations (specifically the partitioned cache) use the serialized form of the key objects for equality testing, which means that keys for which `equals()` returns true must serialize identically; most built-in Java types meet this requirement as well.

To support these methods, import the `com.tangosol.util.Base` and `com.tangosol.util.HashHelper` classes.

- I. The resulting class should look similar to the following.

Example 3–1 Implementation of an Address Class

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

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

import com.tangosol.util.Base;
import com.tangosol.util.HashHelper;

import java.io.IOException;

public class Address implements PortableObject
{
    private String Street1;
    private String Street2;
    private String City;
    private String State;
    private String Zip;
    private String Country;

    /**
     * Default constructor (necessary for PortableObject implementation).
     */
    public Address() {
    }

    public Address(String Street1, String Street2, String City, String State,
        String Zip, String Country) {
        super();
        this.Street1 = Street1;
        this.Street2 = Street2;
        this.City    = City;
        this.State   = State;
        this.Zip     = Zip;
        this.Country = Country;
    }

    //----- accessors-----

    public void setStreet1(String Street1) {
        this.Street1 = Street1;
    }

    public String getStreet1() {
        return Street1;
    }

    public void setStreet2(String Street2) {
        this.Street2 = Street2;
    }

    public String getStreet2() {
        return Street2;
    }
}
```

```

public void setCity(String City) {
    this.City = City;
}

public String getCity() {
    return City;
}

public void setState(String State) {
    this.State = State;
}

public String getState() {
    return State;
}

public void setZip(String Zip) {
    this.Zip = Zip;
}

public String getZip() {
    return Zip;
}

public void setCountry(String Country) {
    this.Country = Country;
}

public String getCountry() {
    return Country;
}
// ----- PortableObject Interface-----

public void readExternal(PofReader reader)
    throws IOException
    {
        setStreet1(reader.readString(0));
        setStreet2(reader.readString(1));
        setCity(reader.readString(2));
        setState(reader.readString(3));
        setZip(reader.readString(4));
        setCountry(reader.readString(5));
    }

public void writeExternal(PofWriter writer)
    throws IOException
    {
        writer.writeString(0, getStreet1());
        writer.writeString(1, getStreet2());
        writer.writeString(2, getCity());
        writer.writeString(3, getState());
        writer.writeString(4, getZip());
        writer.writeString(5, getCountry());
    }
// ----- Object methods -----

public boolean equals(Object oThat)
    {
        if (this == oThat)

```

```

        {
            return true;
        }
        if (oThat == null)
        {
            return false;
        }

        Address that = (Address) oThat;
        return Base.equals(getStreet1(), that.getStreet1()) &&
            Base.equals(getStreet2(), that.getStreet2()) &&
            Base.equals(getCity(), that.getCity()) &&
            Base.equals(getState(), that.getState()) &&
            Base.equals(getZip(), that.getZip()) &&
            Base.equals(getCountry(), that.getCountry());
    }

    public int hashCode()
    {
        return HashHelper.hash(getStreet1(),
            HashHelper.hash(getStreet2(),
                HashHelper.hash(getZip(), 0)));
    }

    public String toString()
    {
        return getStreet1() + "\n" +
            getStreet2() + "\n" +
            getCity() + ", " + getState() + " " + getZip() + "\n" +
            getCountry();
    }
}

```

2. Create a Phone class to store telephone contact data.

- a. Create a new Java class called Phone. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.

Name the Java class Phone. *Do not* select the **Generate Main Method** check box.

- b. Use `PortableObject` for data serialization. In the JDeveloper code editor, change your generated Phone class to implement `com.tangosol.io.pof.PortableObject`. Add an import statement for the `PortableObject` class.
- c. Import the `com.tangosol.io.pof.PofReader` and `com.tangosol.io.pof.PofWriter` classes required by `PortableObject`.
- d. Add the default public constructor for Phone that is required by `PortableObject`.
- e. Enter the following private attributes for your Phone class. You can add others if you like.

```

—short AccessCode
—short CountryCode
—short AreaCode

```

```
—int LocalNumber
```

- f. 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 Phone class. All of the attributes are now automatically selected. Click **OK** to continue.
- g. You can generate the default constructor automatically. From the **Source** menu, select **Generate Constructor from Fields**, click **Select All**, and then click **OK**.

The generated constructor should look similar to the following:

```
public Phone(short AccessCode, short CountryCode, short AreaCode,
             int LocalNumber) {
    super();
    this.AccessCode = AccessCode;
    this.CountryCode = CountryCode;
    this.AreaCode = AreaCode;
    this.LocalNumber = LocalNumber;
}
```

- h. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface.
- i. Implement the `equals`, `hashCode` and `toString` object methods.
- j. The resulting class should look similar to the following.

Example 3–2 Implementation of a Phone 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 com.tangosol.util.HashHelper;

import java.io.IOException;

public class Phone implements PortableObject
{
    private short AccessCode;
    private short CountryCode;
    private short AreaCode;
    private int LocalNumber;

    /**
     * Default constructor (necessary for PortableObject implementation).
     */
    public Phone() {
    }

    public Phone(short AccessCode, short CountryCode, short AreaCode,
                int LocalNumber) {
        super();
        this.AccessCode = AccessCode;
        this.CountryCode = CountryCode;
        this.AreaCode = AreaCode;
        this.LocalNumber = LocalNumber;
    }
}
```

```

}

//----- accessors-----

public void setAccessCode(short AccessCode) {
    this.AccessCode = AccessCode;
}

public short getAccessCode() {
    return AccessCode;
}

public void setCountryCode(short CountryCode) {
    this.CountryCode = CountryCode;
}

public short getCountryCode() {
    return CountryCode;
}

public void setAreaCode(short AreaCode) {
    this.AreaCode = AreaCode;
}

public short getAreaCode() {
    return AreaCode;
}

public void setLocalNumber(int LocalNumber) {
    this.LocalNumber = LocalNumber;
}

public int getLocalNumber() {
    return LocalNumber;
}

// ----- PortableObject Interface-----

public void readExternal(PofReader reader)
    throws IOException
    {
        setAccessCode(reader.readShort(0));
        setCountryCode(reader.readShort(1));
        setAreaCode(reader.readShort(2));
        setLocalNumber(reader.readInt(3));
    }

public void writeExternal(PofWriter writer)
    throws IOException
    {
        writer.writeShort(0, getAccessCode());
        writer.writeShort(1, getCountryCode());
        writer.writeShort(2, getAreaCode());
        writer.writeInt(3, getLocalNumber());
    }

// ----- Object methods -----

/**
 * {@inheritDoc}
 */

```

```
public boolean equals(Object oThat)
{
    if (this == oThat)
    {
        return true;
    }
    if (oThat == null)
    {
        return false;
    }

    Phone that = (Phone) oThat;
    return getAccessCode() == that.getAccessCode() &&
        getCountryCode() == that.getCountryCode() &&
        getAreaCode() == that.getAreaCode() &&
        getLocalNumber() == that.getLocalNumber();
}

/**
 * {@inheritDoc}
 */
public int hashCode()
{
    return HashHelper.hash(getAreaCode(),
        HashHelper.hash(getLocalNumber(), 0));
}

/**
 * {@inheritDoc}
 */
public String toString()
{
    return "+" + getAccessCode() + " " + getCountryCode() + " "
        + getAreaCode() + " " + getLocalNumber();
}
}
```

Creating the Complex Object

The Contact object will provide the name, address, and telephone information of employees by incorporating the Address and Phone data objects.

1. Create a new Java class called Contact. See ["Creating a Java Class"](#) on page 2-13 if you need more information.

Name the Java class Contact. *Do not* select the **Generate Main Method** check box.

2. Since the class will use PortableObject for data serialization, change your generated Contact class to implement `com.tangosol.io.pof.PortableObject` in the JDeveloper code editor. Add an import statement for the PortableObject class.
3. Import the `com.tangosol.io.pof.PofReader` and `com.tangosol.io.pof.PofWriter` classes required by PortableObject.
4. Add the default public constructor for Contact that is required by PortableObject.

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

```

—String FirstName
—String LastName
—Address HomeAddress
—Address WorkAddress
—Map TelephoneNumbers
—java.sql.Date BirthDate

```

6. 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 `Phone` class. All of the attributes are now automatically selected. Click **OK** to continue.
7. Create an accessor, `getAge`, to calculate the age of an employee:

```

public int getAge()
{
    return (int) ((System.currentTimeMillis() - BirthDate.getTime()) /
MILLIS_IN_YEAR);
}

```

8. Add a definition for `MILLIS_IN_YEAR`

```

public static final long MILLIS_IN_YEAR = 1000L * 60L * 60L * 24L * 365L;

```

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

The generated constructor should look similar to the following:

```

public Contact(String FirstName, String LastName, Address HomeAddress,
Address WorkAddress, Map TelephoneNumbers, Date BirthDate) {
    super();
    this.FirstName = FirstName;
    this.LastName = LastName;
    this.HomeAddress = HomeAddress;
    this.WorkAddress = WorkAddress;
    this.TelephoneNumbers = TelephoneNumbers;
    this.BirthDate = BirthDate;
}

```

10. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface.
11. Implement the `equals`, `hashCode`, and `toString` object methods.
12. The resulting class should look similar to the following.

Example 3-3 Sample Contact Class

```

package com.oracle.coherence.handson;

package com.oracle.coherence.handson;

import com.tangosol.io.pof.PortableObject;

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

```

```

import java.io.IOException;

import java.sql.Date;

import java.util.Iterator;
import java.util.Map;

public class Contact implements PortableObject
{

    private String FirstName;
    private String LastName;
    private Address HomeAddress;
    private Address WorkAddress;
    private Map TelephoneNumbers;
    private java.sql.Date BirthDate;

    // ----- constructors -----

    /**
     * Default constructor (necessary for PortableObject implementation).
     */
    public Contact()
    {
    }

    public Contact(String FirstName, String LastName, Address HomeAddress,
        Address WorkAddress, Map TelephoneNumbers, Date BirthDate) {
        super();
        this.FirstName = FirstName;
        this.LastName = LastName;
        this.HomeAddress = HomeAddress;
        this.WorkAddress = WorkAddress;
        this.TelephoneNumbers = TelephoneNumbers;
        this.BirthDate = BirthDate;
    }

    // ----- accessors -----

    public void setFirstName(String FirstName) {
        this.FirstName = FirstName;
    }

    public String getFirstName() {
        return FirstName;
    }

    public void setLastName(String LastName) {
        this.LastName = LastName;
    }

    public String getLastName() {
        return LastName;
    }

    public void setHomeAddress(Address HomeAddress) {
        this.HomeAddress = HomeAddress;
    }

    public Address getHomeAddress() {

```



```

        return HomeAddress;
    }

    public void setWorkAddress(Address WorkAddress) {
        this.WorkAddress = WorkAddress;
    }

    public Address getWorkAddress() {
        return WorkAddress;
    }

    public void setTelephoneNumbers(Map TelephoneNumbers) {
        this.TelephoneNumbers = TelephoneNumbers;
    }

    public Map getTelephoneNumbers() {
        return TelephoneNumbers;
    }

    public void setBirthDate(Date BirthDate) {
        this.BirthDate = BirthDate;
    }

    public Date getBirthDate() {
        return BirthDate;
    }
}
/**
 * Get age.
 *
 * @return age
 */
public int getAge()
{
    return (int) ((System.currentTimeMillis() - BirthDate.getTime()) /
        MILLIS_IN_YEAR);
}

// ----- PortableObject interface -----

/**
 * {@inheritDoc}
 */
public void readExternal(PofReader reader)
    throws IOException
{
    setFirstName(reader.readString(0));
    setLastName(reader.readString(1));
    setHomeAddress((Address) reader.readObject(2));
    setWorkAddress((Address) reader.readObject(3));
    setTelephoneNumbers(reader.readMap(4, null));
    setBirthDate(new Date(reader.readLong(5)));
}

/**
 * {@inheritDoc}
 */
public void writeExternal(PofWriter writer)
    throws IOException
{

```

```

        writer.writeString(0, getFirstName());
        writer.writeString(1, getLastName());
        writer.writeObject(2, getHomeAddress());
        writer.writeObject(3, getWorkAddress());
        writer.writeMap(4, getTelephoneNumbers());
        writer.writeLong(5, getBirthDate().getTime());
    }

    // ----- Object methods -----

    /**
     * {@inheritDoc}
     */
    public String toString()
    {
        StringBuffer sb = new StringBuffer(getFirstName())
            .append(" ")
            .append(getLastName())
            .append("\nAddresses")
            .append("\nHome: ").append(getHomeAddress())
            .append("\nWork: ").append(getWorkAddress())
            .append("\nTelephone Numbers");

        for (Iterator iter = TelephoneNumbers.entrySet().iterator();
             iter.hasNext(); )
        {
            Map.Entry entry = (Map.Entry) iter.next();
            sb.append("\n")
                .append(entry.getKey()).append(": ").append(entry.getValue());
        }
        return sb.append("\nBirth Date: ").append(getBirthDate()).toString();
    }

    /**
     * Approximate number of millis in a year ignoring things such as leap
     * years. Suitable for example use only.
     */
    public static final long MILLIS_IN_YEAR = 1000L * 60L * 60L * 24L * 365L;
}

```

Creating the Driver Class

Create a driver class called `ContactDriver` to put `Contact` entries into the cache and retrieve them.

1. Create a new Java class called `ContactDriver` in the `Contacts` project. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.
Name the Java class `ContactDriver`. Select the **Generate Main Method** check box.
2. In the `ContactDriver` file, create a new `NamedCache` called `contact` and put a new instance of the `Contact` object in it. Get the `Contact` object from the cache and ensure that the two objects are identical.

Example 3–4 Sample ContactDriver Class

```

package com.oracle.coherence.handson;

import com.tangosol.net.CacheFactory;

```

```

import com.tangosol.net.NamedCache;

import java.sql.Date;

import java.util.Map;
import java.util.HashMap;

public class ContactDriver {
    public ContactDriver() {
    }

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

        Address homeAddress = new Address ("4157 Wash Ave", "Suite 4",
                                           "Burlingame", "CA", "94407", "USA");
        Address workAddress = new Address ("500 Oracle Pkwy", "MS989",
                                           "Redwood Shores", "CA", "94065", "USA");
        Date date = new Date(2009, 04, 01);
        Phone phone = new Phone ((short)11, (short)650, (short)506, 7000);
        Map map = new HashMap();
        map.put("home", phone);

        Contact con1 = new Contact("Tom", "Dunn", homeAddress, workAddress,
                                   map, date);

        contact.put(con1.getFirstName(), con1);

        Contact con2 = (Contact)contact.get(con1.getFirstName());

        if (con2.getFirstName().equals(con1.getFirstName())) {
            System.out.println("They are the same!!");
        }
    }
}

```

Creating Configuration Files and the Cache Server Executable

To use POF serialization, you must register your user-defined objects in a POF configuration file. The configuration associates the class of a user-defined object with a numeric value. In addition, you must specify POF serialization and the name of the POF configuration file as part of the cache configuration file.

1. Create a POF configuration file for the `Contact`, `Address`, and `Phone` objects.

Create a POF configuration file for your data types. You can use the `coherence-pof-config.xml` file as a model. Define `<user-type>` elements for the `Contact`, `Address`, and `Phone` objects, assign them type IDs 1001, 1002, and 1003 to them and provide their full class names. The file should include the `coherence-pof-config.xml` file which reserves the first 1000 IDs for Coherence data types.

Save the file as `contacts-pof-config.xml` in the `home\oracle\labs` directory. Save a copy of the `coherence-pof-config.xml` file in the `labs` directory as well. [Example 3-5](#) illustrates a sample `contacts-pof-config.xml` file.

Example 3–5 POF Configuration File

```

<?xml version="1.0"?>

<!DOCTYPE pof-config SYSTEM "pof-config.dtd">

<pof-config>
  <user-type-list>

    <!-- coherence POF user types -->
    <include>coherence-pof-config.xml</include>

    <!-- com.tangosol.examples package -->
    <user-type>
      <type-id>1002</type-id>
      <class-name>com.tangosol.examples.model.Contact</class-name>
    </user-type>
    <user-type>
      <type-id>1003</type-id>
      <class-name>com.tangosol.examples.model.Address</class-name>
    </user-type>
    <user-type>
      <type-id>1004</type-id>
      <class-name>com.tangosol.examples.model.Phone</class-name>
    </user-type>
  </user-type-list>
  <allow-interfaces>true</allow-interfaces>
  <allow-subclasses>true</allow-subclasses>
</pof-config>

```

2. Create a cache configuration file. You can use the `coherence-cache-config.xml` file, which is based on the `cache-config.dtd` as a model.

Use `ExamplesPartitionedPocScheme` as the `scheme-name` and `PartitionedPofCache` as the `service-name`. The `serializer` section identifies the full path of the serializer class that will be used to transform the data. In this case, use the `com.tangosol.io.pof.ConfigurablePofContext` class. The `<init-param>` section points to the name of the POF configuration file, in this case `examples-pof-config.xml`.

Save the file as `contacts-cache-config.xml` in the `home\oracle\labs` directory. Save a copy of the `cache-config.dtd` file in the `labs` directory as well. [Example 3–6](#) illustrates a sample `contacts-cache-config.xml` file.

Example 3–6 Cache Configuration File

```

<?xml version="1.0"?>

<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
    </distributed-scheme>
  </caching-schemes>

```

```

<service-name>PartitionedPofCache</service-name>
<serializer>
  <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
  <init-params>
    <init-param>
      <param-type>String</param-type>
      <param-value>contacts-pof-config.xml</param-value>
    </init-param>
  </init-params>
</serializer>
<backing-map-scheme>
  <local-scheme>
    <!-- each node will be limited to 250MB -->
    <high-units>250M</high-units>
    <unit-calculator>binary</unit-calculator>
  </local-scheme>
</backing-map-scheme>
<autostart>true</autostart>
</distributed-scheme>
</caching-schemes>
</cache-config>

```

3. Create an executable to start the cache server.

Add the following features to the file:

- the location of the `coherence.jar` file to the classpath
- the location of the data classes to the classpath
- the location of the application class files
- the location of the server configuration with `-Dtangosol.coherence.cacheconfig`
- the command to start the default cache server `com.tangosol.net.DefaultCacheServer`

[Example 3-7](#) illustrates a sample cache server executable file.

Example 3-7 Sample Cache Server Executable

```

@echo off
setlocal

if (%COHERENCE_HOME%)==() (
  set COHERENCE_HOME=c:\oracle\product\coherence
)

set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.
jar;C:\home\oracle\labs\Contacts\classes
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.
cacheconfig=%CONFIG%\contacts-cache-config.xml

java %COH_OPTS% -Xms1g -Xmx1g -Xloggc: com.tangosol.net.DefaultCacheServer %2 %3
%4 %5 %6 %7

:exit

```

Running the Sample

1. Stop any cache servers that may be running.

2. Start the Contacts cache server by executing the following command:

```
C:\oracle\product\coherence\examples\java>contacts-cache-server.cmd
```

Example 3-8 lists sample output from running `contacts-cache-server.cmd`.

Example 3-8 Starting the POF Cache Server

```
C:\oracle\product\coherence\examples\java>contacts-cache-server.cmd 2009-03-30 15:31:33.741/0.656
Oracle Coherence 3.5/450b (Internal) <Info> (thread=main, member=n/a): Loaded operational
configuration from resource "jar:f.jar!/tangosol-coherence.xml"
2009-03-30 15:31:33.756/0.671 Oracle Coherence 3.5/450b (Internal) <Info> (thread=main,
member=n/a): Loaded operational overrides from resource
"jar:file:!/tangosol-coherence-override-dev.xml"
2009-03-30 15:31:33.756/0.671 Oracle Coherence 3.5/450b (Internal) <D5> (thread=main, member=n/a):
Optional configuration override "/tangosol-coherence-ov
2009-03-30 15:31:33.756/0.671 Oracle Coherence 3.5/450b (Internal) <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is
```

```
Oracle Coherence Version 3.5/450b (Internal)
Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.
```

```
2009-03-30 15:31:34.350/1.265 Oracle Coherence GE 3.5/450b (Internal) <Info> (thread=main,
member=n/a): Loaded cache configuration from file "C:\oracle\practs-cache-config.xml"
2009-03-30 15:31:35.319/2.234 Oracle Coherence GE 3.5/450b (Internal) <D5> (thread=Cluster,
member=n/a): Service Cluster joined the cluster with senior se
2009-03-30 15:31:38.569/5.484 Oracle Coherence GE 3.5/450b (Internal) <Info> (thread=Cluster,
member=n/a): Created a new cluster "cluster:0x2EDB" with Mem
ddress=130.35.99.50:8088, MachineId=49714, Location=site:us.oracle.
com,machine:tpfaeffl-pc,process:1204, Role=CoherenceServer, Edition=Grid Edition, Mode=
82236332000001205982554FC2321F98
2009-03-30 15:31:38.600/5.515 Oracle Coherence GE 3.5/450b (Internal) <D5>
(thread=Invocation:Management, member=1): Service Management joined the cluster
```

```
2009-03-30 15:31:39.116/6.031 Oracle Coherence GE 3.5/450b (Internal) <D5>
(thread=DistributedCache:PartitionedPofCache, member=1): Service PartitionedPofember 1
2009-03-30 15:31:39.131/6.046 Oracle Coherence GE 3.5/450b (Internal) <Info>
(thread=DistributedCache:PartitionedPofCache, member=1): Loading POF configur
oherence/examples/resource/config2/contacts-pof-config.xml"
2009-03-30 15:31:39.163/6.078 Oracle Coherence GE 3.5/450b (Internal) <Info>
(thread=DistributedCache:PartitionedPofCache, member=1): Loading POF
configurct/coherence/lib/coherence.jar!/coherence-pof-config.xml"
2009-03-30 15:31:39.506/6.421 Oracle Coherence GE 3.5/450b (Internal) <Info> (thread=main,
member=1): Started DefaultCacheServer...
```

```
SafeCluster: Name=cluster:0x2EDB
```

```
Group{Address=224.3.5.0, Port=35450, TTL=4}
```

```
MasterMemberSet
```

```
(
  ThisMember=Member(Id=1, Timestamp=2009-03-30 15:31:34.991, Address=130.35.99.50:8088,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc,p
  OldestMember=Member(Id=1, Timestamp=2009-03-30 15:31:34.991, Address=130.35.99.50:8088,
MachineId=49714, Location=site:us.oracle.com,machine:tpfaeffl-pc
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2009-03-30 15:31:34.991, Address=130.35.99.50:8088, MachineId=49714,
Location=site:us.oracle.com,machine:tpfaeffl-pc,process:12
  )
  RecycleMillis=120000
```

```

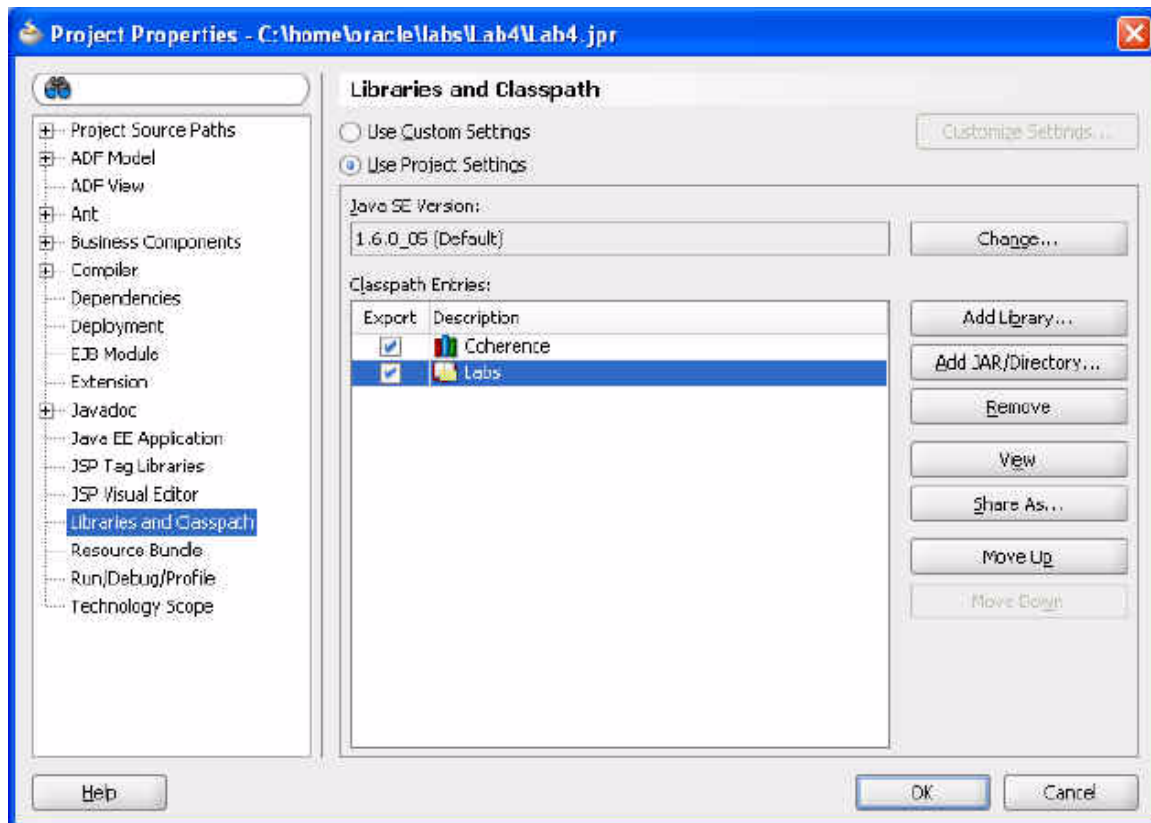
RecycleSet=MemberSet (Size=0, BitSetCount=0
)
)
Services
(
  TcpRing{TcpSocketAcceptor{State=STATE_OPEN, ServerSocket=130.35.99.50:8088}, Connections=[]}
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.4,
  OldestMemberId=1}
  InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
  DistributedCache{Name=PartitionedPofCache, State=(SERVICE_STARTED), LocalStorage=enabled,
  PartitionCount=257, BackupCount=1, AssignedPartitions=257, Bac
)

```

3. Set up the POF cache server to run in JDeveloper.

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 3–3 Adding Labs and Configuration Files to the Classpath



4. 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\contacts-cache-config.xml
```

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

5. Rerun `ContactDriver.java` from the JDeveloper IDE to ensure that it works.

In this example, the Contact object is converted to POF at run time. Using POF should provide significant performance improvements in terms of CPU time and the size of the binary generated.

Figure 3–4 *Contacts Example Output Run from JDeveloper*

```
Oracle Coherence Version 3.5/450b (Internal)
Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.

2009-03-30 16:53:08.303/1.047 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:09.319/2.063 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:09.538/2.282 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:09.569/2.313 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:09.569/2.313 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:09.678/2.422 Oracle Coherence CE 3.5/450b (Internal)
They are the same!!
2009-03-30 16:53:10.147/2.891 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:10.147/2.891 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:10.288/3.032 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:10.335/3.079 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:10.741/3.485 Oracle Coherence CE 3.5/450b (Internal)
2009-03-30 16:53:10.741/3.485 Oracle Coherence CE 3.5/450b (Internal)
Process exited with exit code 0.
```

Loading Data Into a Cache

In this chapter, you will learn how to populate a Coherence cache with domain objects that are read from text files.

This chapter contains the following sections:

- [Introduction](#)
- [Populating a Cache with Domain Objects](#)
- [Querying and Aggregating Data in the Cache](#)

Introduction

Until now, you have been putting and retrieving objects individually. Each `put` can result in increased network traffic, especially for partitioned and replicated caches. Additionally, each call to `put` returns the object it just replaced in the cache, which adds more unnecessary overhead. Loading the cache can be made much more efficient by using the `putAll` method instead.

This chapter assumes that you have completed "[Creating and Caching Complex Objects](#)" on page 3-2. It 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.

Populating a Cache with Domain Objects

This exercise demonstrates how to create a console application that populates a Coherence cache with domain objects. The application can use the Coherence `com.tangosol.io.pof.PortableObject` implementation.

This exercise has three parts. You will create a key that will help to obtain the `Contact` object, a generator to provide data for the cache, and a loader to load the cache.

To populate a cache with domain objects:

1. Create a new project called `Loading`.
See "[Creating and Caching Complex Objects](#)" on page 3-2 for information on creating a new project.
2. Use the `Address`, `Phone`, and `Contact` classes that you created in an earlier exercise (`Contacts`). Add `c:\home\oracle\labs\Contacts\classes` to the project classpath (Hint: right-click the project and choose **Project Properties>Libraries and Classpath**. You can find more detailed information in "[Adding JARS and Libraries to the Project Classpath](#)" on page 2-16.)

3. Create a contact ID class that provides a key to the employee for whom information is tracked. See "[Creating a Java Class](#)" on page 2-13 if you need detailed information on creating a Java class.

Create the contact ID based on the employee's first name and last name. This object acts as the key to obtain the `Contact` object.

Since this class will use POF serialization, it must implement `PortableObject` and provide implementations for the `writeExternal` and `readExternal` `PortableObject` methods and the `equals`, `hashCode`, and `toString` object methods.

Note: Cache keys and values must be serializable (for example, `java.io.Serializable`). Cache keys must also provide an implementation of the `hashCode()` and `equals()` methods, and those methods must return consistent results across cluster nodes. This implies that the implementation of `hashCode()` and `equals()` must be based solely on the object's serializable state (that is, the object's non-transient fields); most built-in Java types, such as `String`, `Integer` and `Date`, meet this requirement. Some cache implementations (specifically the partitioned cache) use the serialized form of the key objects for equality testing, which means that keys for which `equals()` returns true must serialize identically; most built-in Java types meet this requirement as well.

[Example 4-1](#) illustrates a possible solution.

Example 4-1 Simple Contact ID 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 com.tangosol.util.Base;
import com.tangosol.util.HashHelper;

import java.io.IOException;

/**
 * ContactId is a key to the person for whom information is
 * tracked.
 */
public class ContactId
    implements PortableObject
    {
    // ----- constructors -----

    /**
     * Default constructor (necessary for PortableObject implementation).
     */
    public ContactId() {
        }

    /**
     * Construct a contact person.
     */
}
```

```

*/
public ContactId(String FirstName, String LastName)
{
    super();
    this.FirstName = FirstName;
    this.LastName = LastName;
}

// ----- accessors -----

/**
 * Return the first name.
 */
public String getFirstName()
{
    return FirstName;
}

/**
 * Return the last name.
 */
public String getLastName()
{
    return LastName;
}

// ----- PortableObject interface -----

public void readExternal(PofReader reader)
    throws IOException
{
    FirstName = reader.readString(0);
    LastName = reader.readString(1);
}

public void writeExternal(PofWriter writer)
    throws IOException
{
    writer.writeString(0, FirstName);
    writer.writeString(1, LastName);
}

// ----- Object methods -----

public boolean equals(Object oThat)
{
    if (this == oThat)
    {
        return true;
    }
    if (oThat == null)
    {
        return false;
    }

    ContactId that = (ContactId) oThat;
    return Base.equals(getFirstName(), that.getFirstName()) &&
        Base.equals(getLastName(), that.getLastName());
}

```

```

    }

    public int hashCode()
    {
        return HashHelper.hash(getFirstName(),
                               HashHelper.hash(getLastName(), 0));
    }

    public String toString()
    {
        return getFirstName() + " " + getLastName();
    }

    // ----- data members -----

    /**
     * First name.
     */
    private String FirstName;

    /**
     * Last name.
     */
    private String LastName;
}

```

4. Add a <user-type> entry for ContactID to the contacts-pof-config.xml file.

Example 4-2 POF Configuration File with the ContactId Entry

```

<?xml version="1.0"?>

<!DOCTYPE pof-config SYSTEM "pof-config.dtd">

<pof-config>
  <user-type-list>

    <!-- coherence POF user types -->
    <include>coherence-pof-config.xml</include>

    <!-- com.tangosol.examples package -->
    <user-type>
      <type-id>1001</type-id>
      <class-name>com.oracle.coherence.handson.Contact</class-name>
    </user-type>
    <user-type>
      <type-id>1002</type-id>
      <class-name>com.oracle.coherence.handson.Address</class-name>
    </user-type>
    <user-type>
      <type-id>1003</type-id>
      <class-name>com.oracle.coherence.handson.Phone</class-name>
    </user-type>
    <user-type>
      <type-id>1004</type-id>
      <class-name>com.oracle.coherence.handson.ContactId</class-name>
    </user-type>
  </user-type-list>
</pof-config>
<allow-interfaces>>true</allow-interfaces>

```

```
<allow-subclasses>true</allow-subclasses>
</pof-config>
```

5. Create a Java class named `DataGenerator` to generate random contact names and addresses. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information on creating a Java class.

Hint: Use the `Address`, `Phone`, and `Contact` classes that you created in the earlier exercise. Use `java.util.Random` to generate some random names, addresses, telephone numbers, and ages.

[Example 4-3](#) illustrates a possible solution. This solution creates a text file, `contacts.csv`, that contains the employee contact information. The file will be stored in the root of the project directory, in this case, `C:\home\oracle\labs>Loading`

You might see an error in JDeveloper caused by the `LoaderExample.FILENAME` declaration. This can be ignored—you will create `LoaderExample` in the next step.

Example 4-3 Sample Data Generation Class

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

import com.oracle.coherence.handson.Address;
import com.oracle.coherence.handson.Contact;
import com.oracle.coherence.handson.Phone;
import com.tangosol.util.Base;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;

import java.sql.Date;

import java.util.Collections;
import java.util.Random;

/**
 * DataGenerator is a generator of sample contacts.
 */
public class DataGenerator
{
    // ----- static methods -----

    /**
     * Generate contacts.
     */
    public static void main(String[] asArg)
        throws IOException
    {
        String      sFile = asArg.length > 0 ? asArg[0] : LoaderExample.FILENAME;
        int         cCon  = asArg.length > 1 ? Integer.parseInt(asArg[1]) : 1000;
        OutputStream out  = new FileOutputStream(sFile);

        generate(out, cCon);
        out.close();
    }
}
```

```
    }

    /**
     * Generate the contacts and write them to a file.
     */
    public static void generate(OutputStream out, int cContacts)
        throws IOException
    {
        PrintWriter writer = new PrintWriter(new BufferedWriter(
            new OutputStreamWriter(out)));

        for (int i = 0; i < cContacts; ++i)
        {
            StringBuffer sb = new StringBuffer(256);

            //contact person
            sb.append("John, ")
              .append(getRandomName())
              .append(',');

            // home and work addresses
            sb.append(Integer.toString(Base.getRandom().nextInt(999)))
              .append(" Beacon St., ") /*street1,empty street2*/
              .append(getRandomName()) /*random city name*/
              .append(',')
              .append(getRandomState())
              .append(',')
              .append(getRandomZip())
              .append(",US,Yoyodyne Propulsion Systems,")
              .append("330 Lectroid Rd.,Grover's Mill,")
              .append(getRandomState())
              .append(',')
              .append(getRandomZip())
              .append(",US,");

            // home and work telephone numbers
            sb.append("home, ")
              .append(Base.toDelimitedString(getRandomPhoneDigits(), ","))
              .append(",work, ")
              .append(Base.toDelimitedString(getRandomPhoneDigits(), ","))
              .append(',');

            // random birth date in millis before or after the epoch
            sb.append(getRandomDateInMillis());

            writer.println(sb);
        }
        writer.flush();
    }

    /**
     * Return a random name.
     */
    private static String getRandomName()
    {
        Random rand = Base.getRandom();
        int cCh = 4 + rand.nextInt(7);
        char[] ach = new char[cCh];
```

```

        ach[0] = (char) ('A' + rand.nextInt(26));
        for (int of = 1; of < cCh; ++of)
        {
            ach[of] = (char) ('a' + rand.nextInt(26));
        }
        return new String(ach);
    }

/**
 * Return a random phone number.
 * The phone number includes access, country, area code, and local
 * number.
 */
private static int[] getRandomPhoneDigits()
{
    Random rand = Base.getRandom();
    return new int[] {
        11, // access code
        rand.nextInt(99), // country code
        rand.nextInt(999), // area code
        rand.nextInt(9999999) // local number
    };
}

/**
 * Return a random Phone.
 */
private static Phone getRandomPhone()
{
    int[] anPhone = getRandomPhoneDigits();

    return new Phone((short)anPhone[0], (short)anPhone[1],
        (short)anPhone[2], anPhone[3]);
}

/**
 * Return a random Zip code.
 */
private static String getRandomZip()
{
    return Base.toDecString(Base.getRandom().nextInt(99999), 5);
}

/**
 * Return a random state.
 */
private static String getRandomState()
{
    return STATE_CODES[Base.getRandom().nextInt(STATE_CODES.length)];
}

/**
 * Return a random date in millis before or after the epoch.
 */
private static long getRandomDateInMillis()

```

```

        {
            return (Base.getRandom().nextInt(40) - 20) * Contact.MILLIS_IN_YEAR;
        }

/**
 * Generate a Contact with random information.
 *
 */
public static Contact getRandomContact()
{
    return new Contact("John",
        getRandomName(),
        new Address("1500 Boylston St.", null, getRandomName(),
            getRandomState(), getRandomZip(), "US"),
        new Address("8 Yawkey Way", null, getRandomName(),
            getRandomState(), getRandomZip(), "US"),
        Collections.singletonMap("work", getRandomPhone()),
        new Date(getRandomDateInMillis()));
}

// ----- constants -----

/**
 * US Postal Service two letter postal codes.
 */
private static final String[] STATE_CODES = {
    "AL", "AK", "AS", "AZ", "AR", "CA", "CO", "CT", "DE", "OF", "DC",
    "FM", "FL", "GA", "GU", "HI", "ID", "IL", "IN", "IA", "KS", "KY",
    "LA", "ME", "MH", "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE",
    "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "MP", "OH", "OK", "OR",
    "PW", "PA", "PR", "RI", "SC", "SD", "TN", "TX", "UT", "VT", "VI",
    "VA", "WA", "WV", "WI", "WY"
};
}

```

6. Create a console application (Java class) called `LoadingExample`. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.

Use input streams and buffered readers to load the entire employee information contained in the `contacts.csv` file into a single Coherence cache.

The application should contain the code to parse the employee information in the data file. Once you have this information, construct the individual contacts to put into the cache. To conserve processing effort and minimize network traffic, use the `putAll` method to load the cache.

[Example 4-4](#) illustrates a possible solution.

Example 4-4 Sample Cache Loading Program

```

package com.tangosol.examples.contacts;

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

import com.tangosol.examples.model.ContactId;
import com.tangosol.examples.model.Address;
import com.tangosol.examples.model.Phone;
import com.tangosol.examples.model.Contact;

import java.io.BufferedReader;

```



```

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import java.sql.Date;

import java.util.HashMap;
import java.util.Map;

/**
 * LoaderExample loads contacts into the cache from a files.
 *
 */
public class LoaderExample
{
    // ----- static methods -----

    /**
     * Load contacts.
     */
    public static void main (String[] asArg)
        throws IOException
    {
        String sFile = asArg.length > 0 ? asArg[0] : FILENAME;
        String sCache = asArg.length > 1 ? asArg[1] : CACHENAME;

        System.out.println("input file: " + sFile);
        System.out.println("cache name: " + sCache);

        new LoaderExample().load(CacheFactory.getCache(sCache),
            new FileInputStream(sFile));
        CacheFactory.shutdown();
    }

    /**
     * Load cache from stream.
     */
    public void load(NamedCache cache, InputStream in)
        throws IOException
    {
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        Map mapBatch = new HashMap(1024);
        String sRecord;
        int cRecord = 0;

        while ((sRecord = reader.readLine()) != null)
        {
            // parse record
            String[] asPart = sRecord.split(",");
            int ofPart = 0;
            String sFirstName = asPart[ofPart++];
            String sLastName = asPart[ofPart++];
            ContactId id = new ContactId(sFirstName, sLastName);
            Address addrHome = new Address(
                /*streetline1*/ asPart[ofPart++],
                /*streetline2*/ asPart[ofPart++],
                /*city*/ asPart[ofPart++],

```

```

                                /*state*/      asPart[ofPart++],
                                /*zip*/        asPart[ofPart++],
                                /*country*/    asPart[ofPart++]);
Address    addrWork    = new Address(
                                /*streetline1*/ asPart[ofPart++],
                                /*streetline2*/ asPart[ofPart++],
                                /*city*/       asPart[ofPart++],
                                /*state*/      asPart[ofPart++],
                                /*zip*/        asPart[ofPart++],
                                /*country*/    asPart[ofPart++]);
Map        mapTelNum   = new HashMap();

for (int c = asPart.length - 1; ofPart < c; )
{
    mapTelNum.put(/*type*/ asPart[ofPart++], new Phone(
        /*access code*/ Short.parseShort(asPart[ofPart++]),
        /*country code*/ Short.parseShort(asPart[ofPart++]),
        /*area code*/   Short.parseShort(asPart[ofPart++]),
        /*local num*/   Integer.parseInt(asPart[ofPart++])));
}
Date dtBirth = new Date(Long.parseLong(asPart[ofPart]));

// Construct Contact and add to batch
mapBatch.put(id, new Contact(sFirstName, sLastName, addrHome,
    addrWork, mapTelNum, dtBirth));

++cRecord;
if (cRecord % 1024 == 0)
{
    // load batch
    cache.putAll(mapBatch);
    mapBatch.clear();
    System.out.print('.');
    System.out.flush();
}
}

if (!mapBatch.isEmpty())
{
    // load final batch
    cache.putAll(mapBatch);
}

System.out.println("Added " + cRecord + " entries to cache");
}

// ----- constants -----
/**
 * Default cache name.
 */
public static final String CACHENAME = "ContactsCache";

/**
 * Default file name.
 */
public static final String FILENAME = "contacts.csv";
}

```

7. Add the cache configuration file (C:\home\oracle\labs) to the project's classpath. See ["Adding JARS and Libraries to the Project Classpath"](#) on page 2-16 if you need detailed information.

8. Add the path to the cache configuration to the **Java Options** field of the **Project Properties>Run/Debug/Profile**.

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

9. Edit the `contacts-cache-server.cmd` file to add the compiled classes for the Loading project (in this case, C:\home\oracle\labs>Loading\classes). Ensure that the path to the directory where the cache configuration file is stored (in this case, C:\home\oracle\labs) is already listed.

The `contacts-cache-server.cmd` file should look similar to [Example 4-5](#):

Example 4-5 Sample contacts-cache-server.cmd File

```
@echo off
setlocal

if (%COHERENCE_HOME%)==( ) (
    set COHERENCE_HOME=c:\oracle\product\coherence
)

set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.
jar;C:\home\oracle\labs\Contacts\classes;C:\home\oracle\labs>Loading\classes;C:\home\oracle\labs;
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.
cacheconfig=\home\oracle\labs\contacts-cache-config.xml

java %COH_OPTS% -Xms1g -Xmx1g -Xloggc: com.tangosol.net.DefaultCacheServer %2 %3
%4 %5 %6 %7

:exit
```

10. Start the cache server with the `contacts-cache-server.cmd` file.
11. Run `DataGenerator` from `JDeveloper`, then run `LoaderExample`.

A stream of employee contact information should appear in the `JDeveloper` output window. [Figure 4-1](#) illustrates an example.

Figure 4–1 Output from the Sample Cache Loading Program

```

Running: Loading.jpr - Log X
work: +11 5 349 1940267
home: +11 24 370 404073
Birth Date: 1971-12-31
John Inqikfxqx
Addresses
Home: 955 Beacon St.

Virfqjf, WA 16897
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, MP 12840
US
Telephone Numbers
work: +11 88 723 4066276
home: +11 16 626 406497
Birth Date: 1969-12-31
John Uvdsjq
Addresses
Home: 572 Beacon St.

Dkedtffwps, NJ 04172
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, DC 23540
US
Telephone Numbers
work: +11 21 657 6513801
home: +11 77 479 7986221
Birth Date: 1958-01-03
2009-04-21 11:31:48.413/5.734 Oracle Coherence GE 3.5/453 (Pre-rel:
Added 1000 entries to cache
Process exited with exit code 0.

```

Querying and Aggregating Data in the Cache

This exercise introduces the concept of querying and aggregating data in a cache. This exercise demonstrates how to:

- query the cache for specific data
- aggregate information within the cache

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 also have the overhead of deserializing when querying. When indexes are added, the *values* kept in the index

itself are deserialized and therefore, offer quicker access time. The *objects* that are indexed are always kept serialized.

Some of the more useful methods in the `QueryMap` interface are:

- `Set entrySet(Filter filter)`—Returns a set of entries that are contained in the map that satisfy the filter
- `addIndex(ValueExtractor extractor, boolean fOrdered, 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 be 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 most 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 `getHomeAddress.getState` method on an object in a cache is for Massachusetts (MA):

```
(new EqualsFilter("getHomeAddress.getState", "MA"));
```

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 who live in a city whose name begins with the letter "S":


```
Set sPeople = cache.entrySet(new LikeFilter("getHomeAddress.getCity", "S%"));
```
- Return a set containing people over age 42:


```
final int nAge = 42;
// Find all contacts who are older than nAge
Set sSeniors = cache.entrySet(new GreaterFilter("getAge", nAge));
```

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("getCity")
```

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

An especially useful type of extractor is the `ChainedExtractor`. This is a composite `ValueExtractor` implementation based on an array of extractors. Extractors in the array are applied sequentially left-to-right, so a result of a previous extractor serves as a target object for a next one. For example:

```
new ChainedExtractor(new ReflectionExtractor("getHomeAddress"), new
ReflectionExtractor("getState"))
```

This example assumes that `HomeAddress` and `State` belong to a complex `Contact` object. `ChainedExtractor` first uses reflection to call `getHomeAddress` on each cached `Contact` object, and then uses reflection to call `getState` on the set of returned `HomeAddress`.

To aggregate and query data in the cache:

1. Create a new Java class called `QueryExample` to perform your queries. Ensure that the class has a `main` method. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.

Use the `entrySet` method to get the employee contact information for:

- all employees who live in Massachusetts. Hint:

```
cache.entrySet(new EqualsFilter("getHomeAddress.getState", "MA"));
```

- all employees who live in Massachusetts and work elsewhere. Hint:

```
cache.entrySet(new AndFilter(
    new EqualsFilter("getHomeAddress.getState", "MA"),
    new NotEqualsFilter("getWorkAddress.getState", "MA")));
```

- all employees whose city name begins with 'S'. Hint:

```
cache.entrySet(new LikeFilter("getHomeAddress.getCity", "S%"));
```

- all employees with last name beginning with 'S' that live in Massachusetts. Use both key and value in the query. Hint:

```
cache.entrySet(new AndFilter(
    new LikeFilter(new KeyExtractor("getLastName"), "S%",
```

```
(char) 0, false),
    new EqualsFilter("getHomeAddress.getState", "MA")));
```

- all employees who are older than a specified age. Hint:

```
final int nAge = 42;
    setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
```

Use indexes to improve performance. Hints: Find the `addIndex` method in the Javadoc for the `QueryMap` interface.

[Example 4–6](#) illustrates a possible solution.

Example 4–6 Sample QueryExample Class

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

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

import com.tangosol.util.extractor.ChainedExtractor;
import com.tangosol.util.extractor.KeyExtractor;
import com.tangosol.util.extractor.ReflectionExtractor;

import com.tangosol.util.filter.AlwaysFilter;
import com.tangosol.util.filter.AndFilter;
import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.filter.GreaterFilter;
import com.tangosol.util.filter.LikeFilter;
import com.tangosol.util.filter.NotEqualsFilter;

import java.util.Iterator;
import java.util.Set;

/**
 * QueryExample runs sample queries for contacts.
 *
 */
public class QueryExample{

    // ----- QueryExample methods -----

    public static void main(String[] args) {
        NamedCache cache = CacheFactory.getCache("ContactsCache");
        query(cache);
    }
    /**
     * Perform the example queries
     *
     */
    public static void query(NamedCache cache)
    {
        // Add indices to make queries more efficient
        ReflectionExtractor reflectAddrHome =
            new ReflectionExtractor("getHomeAddress");

        // Add an index for the age
        cache.addIndex(new ReflectionExtractor("getAge"), true, null);

        // Add index for state within home address
```

```

        cache.addIndex(new ChainedExtractor(reflectAddrHome,
            new ReflectionExtractor("getState")), true, null);

// Add index for state within work address
    cache.addIndex(new ChainedExtractor(
        new ReflectionExtractor("getWorkAddress"),
        new ReflectionExtractor("getState")), true, null);

// Add index for city within home address
    cache.addIndex(new ChainedExtractor(reflectAddrHome,
        new ReflectionExtractor("getCity")), true, null);

// Find all contacts who live in Massachusetts
    Set setResults = cache.entrySet(new EqualsFilter(
        "getHomeAddress.getState", "MA"));
    printResults("MA Residents", setResults);

// Find all contacts who live in Massachusetts and work elsewhere
    setResults = cache.entrySet(new AndFilter(
        new EqualsFilter("getHomeAddress.getState", "MA"),
        new NotEqualsFilter("getWorkAddress.getState", "MA")));
    printResults("MA Residents, Work Elsewhere", setResults);

// Find all contacts whose city name begins with 'S'
    setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity",
        "S%"));
    printResults("City Begins with S", setResults);

    final int nAge = 42;
// Find all contacts who are older than nAge
    setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
    printResults("Age > " + nAge, setResults);

// Find all contacts with last name beginning with 'S' that live
// in Massachusetts. Uses both key and value in the query.
    setResults = cache.entrySet(new AndFilter(
        new LikeFilter(new KeyExtractor("getLastName"), "S%",
            (char) 0, false),
        new EqualsFilter("getHomeAddress.getState", "MA")));
    printResults("Last Name Begins with S and State Is MA", setResults);

    }

/**
 * Print results of the query
 *
 * @param sTitle    the title that describes the results
 * @param setResults a set of query results
 */
private static void printResults(String sTitle, Set setResults)
    {
        System.out.println(sTitle);
        for (Iterator iter = setResults.iterator(); iter.hasNext(); )
            {
                System.out.println(iter.next());
            }
    }
}

```


2. Stop all running cache servers. Restart the Contacts cache server with `contacts-cache-server.cmd`. Run the `DataGenerator` file, the `LoadExample` file, then the `QueryExample` file. After printing all of the contact information in the cache, it displays the results of the queries. The results should look similar to [Example 4-1](#).

Figure 4-2 Output of the `QueryExample` Class

```

2009-04-02 16:31:47.881/1.312 Oracle Coherence CE 3.5/
2009-04-02 16:31:49.335/2.766 Oracle Coherence CE 3.5/
2009-04-02 16:31:49.960/3.391 Oracle Coherence CE 3.5/
2009-04-02 16:31:49.975/3.406 Oracle Coherence CE 3.5/
MA Residents
ConverterEntry{Key="John Pwfe", Value="John Pwfe
Addresses
Home: 479 Beacon St.

Wmurdqs, MA 47981
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, RI 84405
US
Telephone Numbers
work: +11 2 816 4697504
home: +11 74 7 846471
Birth Date: 1971-12-31"}
ConverterEntry{Key="John Kvimuvfv", Value="John Kvimuv
Addresses
Home: 987 Beacon St.

Frvnn, MA 51425
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, RI 05553
US
Telephone Numbers
work: +11 17 159 558433
home: +11 96 257 5295402
Birth Date: 1976-12-29"}
ConverterEntry{Key="John Epkjxak", Value="John Epkjxak
Addresses
Home: 181 Beacon St.

Mchigusz, MA 97665
US
Work: Yoyodyne Propulsion Systems

```

3. Add code in `QueryExample.java` to perform aggregations on the data in the cache.

An `EntryAggregator` (`com.tangosol.util.InvokeableMap.EntryAggregator`) enables you to perform operations on all or a specific set of objects and return an aggregation. `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 4–7](#) illustrates the methods that perform these aggregations.

Example 4–7 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. Using aggregations, write code in the `QueryExample` class to calculate the following:

—the number of employees that are greater than a specified age. Hint: use the `GreaterFilter` and the `Count` class:

```
cache.aggregate(new GreaterFilter("getAge", nAge), new Count())
```

—lowest age of the set of employees. Hint: use the `AlwaysFilter` and the `LongMin` class:

```
cache.aggregate(AlwaysFilter.INSTANCE, new LongMin("getAge"))
```

—highest age of the set of employees. Hint: use the `AlwaysFilter` and the `LongMax` class:

```
cache.aggregate(AlwaysFilter.INSTANCE, new LongMax("getAge"))
```

—average age of employees. Hint: use the `AlwaysFilter` and the `DoubleAverage` class:

```
cache.aggregate(AlwaysFilter.INSTANCE, new DoubleAverage("getAge"))
```

- b. Import the `Count`, `DoubleAverage`, `LongMax`, and `LongMin` aggregator classes.

```
import com.tangosol.util.aggregator.Count;
import com.tangosol.util.aggregator.DoubleAverage;
import com.tangosol.util.aggregator.LongMax;
import com.tangosol.util.aggregator.LongMin;
```

The `QueryExample.java` file should now look similar to this:

Example 4–8 QueryExample with Aggregation

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

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

import com.tangosol.util.aggregator.Count;
import com.tangosol.util.aggregator.DoubleAverage;
import com.tangosol.util.aggregator.LongMax;
import com.tangosol.util.aggregator.LongMin;

import com.tangosol.util.extractor.ChainedExtractor;
import com.tangosol.util.extractor.KeyExtractor;
import com.tangosol.util.extractor.ReflectionExtractor;
```

```

import com.tangosol.util.filter.AlwaysFilter;
import com.tangosol.util.filter.AndFilter;
import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.filter.GreaterFilter;
import com.tangosol.util.filter.LikeFilter;
import com.tangosol.util.filter.NotEqualsFilter;

import java.util.Iterator;
import java.util.Set;

/**
 * QueryExample runs sample queries for contacts.
 */
public class QueryExample{

    // ----- QueryExample methods -----

    public static void main(String[] args) {
        NamedCache cache = CacheFactory.getCache("ContactsCache");
        query(cache);
    }
    /**
     * Perform the example queries
     */
    public static void query(NamedCache cache)
    {
        // Add indices to make queries more efficient
        ReflectionExtractor reflectAddrHome =
            new ReflectionExtractor("getHomeAddress");

        cache.addIndex(new ReflectionExtractor("getAge"), true, null);
        cache.addIndex(new ChainedExtractor(reflectAddrHome,
            new ReflectionExtractor("getState")), true, null);
        cache.addIndex(new ChainedExtractor(
            new ReflectionExtractor("getWorkAddress"),
            new ReflectionExtractor("getState")), true, null);
        cache.addIndex(new ChainedExtractor(reflectAddrHome,
            new ReflectionExtractor("getCity")), true, null);

        // Find all contacts who live in Massachusetts
        Set setResults = cache.entrySet(new EqualsFilter(
            "getHomeAddress.getState", "MA"));
        printResults("MA Residents", setResults);

        // Find all contacts who live in Massachusetts and work elsewhere
        setResults = cache.entrySet(new AndFilter(
            new EqualsFilter("getHomeAddress.getState", "MA"),
            new NotEqualsFilter("getWorkAddress.getState", "MA")));
        printResults("MA Residents, Work Elsewhere", setResults);

        // Find all contacts whose city name begins with 'S'
        setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity",
            "S%"));
        printResults("City Begins with S", setResults);

        final int nAge = 42;
        // Find all contacts who are older than nAge
        setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
    }
}

```

```

printResults("Age > " + nAge, setResults);

// Find all contacts with last name beginning with 'S' that live
// in Massachusetts. Uses both key and value in the query.
setResults = cache.entrySet(new AndFilter(
    new LikeFilter(new KeyExtractor("getLastName"), "S%",
        (char) 0, false),
    new EqualsFilter("getHomeAddress.getState", "MA")));
printResults("Last Name Begins with S and State Is MA", setResults);

// Count contacts who are older than nAge
System.out.println("count > " + nAge + ": " + cache.aggregate(
    new GreaterFilter("getAge", nAge), new Count());

// Find minimum age
System.out.println("min age: " + cache.aggregate(AlwaysFilter.INSTANCE,
    new LongMin("getAge")));

// Calculate average age
System.out.println("avg age: " + cache.aggregate(AlwaysFilter.INSTANCE,
    new DoubleAverage("getAge")));

// Find maximum age
System.out.println("max age: " + cache.aggregate(AlwaysFilter.INSTANCE,
    new LongMax("getAge"));
}

/**
 * Print results of the query
 *
 */
private static void printResults(String sTitle, Set setResults)
{
    System.out.println(sTitle);
    for (Iterator iter = setResults.iterator(); iter.hasNext(); )
    {
        System.out.println(iter.next());
    }
}
}

```

- c. Stop and start the cache server after making this change.
- d. Start the Contacts cache server with the `contacts-cache-server.cmd` file. Run the `QueryExample` application.

You should see out output similar to [Example 4-3](#) in JDeveloper.

Figure 4-3 Output from the Aggregators

```
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, KY 01694
US
Telephone Numbers
work: +11 88 852 4260648
home: +11 20 340 7345385
Birth Date: 1968-12-31"}
count > 42: 435 ← output from
min age: 20          aggregators
avg age: 39.73
max age: 59
Process exited with exit code 0.
```

Listening for Changes and Modifying Data

In this chapter, you set up listeners to observe data changes within a `NamedCache`. You also learn how `EntryProcessors` can be used to modify and perform processing on entries in the Coherence cache. This chapter contains the following sections:

- [Introduction](#)
- [Creating a Cache Listener](#)
- [Responding to Changes in the Cache](#)

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. `MapListeners` are called on the client; that is, the listener code is executed in your client process.

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

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

1. In the Loading project, create a new class that listens for a new `Contact` object entry.

Name the class `ObserverExample` and ensure that it has a `main` method. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.

2. Within this class, add a listener to display a message whenever a new `Contact` is updated 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.NamedCache;

import com.tangosol.util.AbstractMapListener;
import com.tangosol.util.MapEvent;

import com.oracle.coherence.handson.Contact;

import com.tangosol.net.CacheFactory;

import java.io.IOException;

/**
 * ObserverExample observes changes to contacts.
 */
public class ObserverExample
{
    public ObserverExample() {
    }
    // ----- ObserverExample methods -----

```



```

public static void main(String[] args) {
    NamedCache cache = CacheFactory.getCache("ContactsCache");
    new ObserverExample().observe(cache);
    try {
        System.in.read();
    } catch (IOException e) {
    }
}
/**
 * Observe changes to the contacts.
 *
 * @param cache target cache
 */
public void observe(NamedCache cache)
{
    cache.addMapListener(new ContactChangeListener());
}

// ----- inner class: ContactChangeListener -----

public class ContactChangeListener
    extends AbstractMapListener
{
    // ----- MapListener interface -----

    public void entryInserted(MapEvent event)
    {
        System.out.println(event);
    }

    public void entryUpdated(MapEvent event)
    {
        Contact contactOld = (Contact)event.getOldValue();
        Contact contactNew = (Contact)event.getNewValue();
        StringBuffer sb = new StringBuffer();

        if (!contactOld.getHomeAddress().equals(
            contactNew.getHomeAddress()))
        {
            sb.append("Home address ");
        }

        if (!contactOld.getWorkAddress().equals(
            contactNew.getWorkAddress()))
        {
            sb.append("Work address ");
        }

        if (!contactOld.getTelephoneNumbers().equals(
            contactNew.getTelephoneNumbers()))
        {
            sb.append("Telephone ");
        }

        if (contactOld.getAge() != contactNew.getAge())
        {
            sb.append("Birthdate ");
        }

        sb.append("was updated for ").append(event.getKey());
        System.out.println(sb);
    }

    public void entryDeleted(MapEvent event)

```

```

        {
            System.out.println(event.getKey());
        }
    }
}

```

3. To enable the console input, you must perform the following:
 - Right-click the `Loading` 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.
4. Turn off local storage if it is not already disabled.
 - Right-click the `Loading` project and select **Project Properties**.
 - Select **Run/Debug/Profile** at the left.
 - Select the **Default** run configuration and click **Edit**. In **Java Options** field, add the command to disable local storage.

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

5. Edit the `contacts-cache-server.cmd` file to add the classes for the `Loading` project to the classpath if it is not already there.

```
C:\home\oracle\labs>Loading\classes
```

6. Edit the `contacts-pof-config.xml` file to add a user type ID for the `OfficeUpdater` entries

```

...
<user-type>
    <type-id>1006</type-id>
    <class-name>com.oracle.coherence.handson.
    ProcessorExample$OfficeUpdater</class-name>
</user-type>
...

```

7. Start the cache server if it is not already running.
8. Load the cache by running the `LoaderExample` program from JDeveloper. If you now run the `ObserverExample`, you will see the program wait for input.

In the next section, you will create a program that modifies entries in the cache and returns the changed records.

Figure 5-1 Listener Program Waiting for Events

```

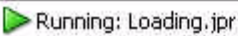
C:\oracle\product\jdk160_05\bin\javaw.exe -client -classpath C:\home
2009-04-21 11:43:08.288/0.625 Oracle Coherence 3.5/453 (Pre-release)
2009-04-21 11:43:08.288/0.625 Oracle Coherence 3.5/453 (Pre-release)
2009-04-21 11:43:08.288/0.625 Oracle Coherence 3.5/453 (Pre-release)
2009-04-21 11:43:08.304/0.641 Oracle Coherence 3.5/453 (Pre-release)

Oracle Coherence Version 3.5/453 (Pre-release)
Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.

2009-04-21 11:43:08.710/1.047 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:09.820/2.157 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.038/2.375 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.070/2.407 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.070/2.407 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.148/2.485 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.679/3.016 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.695/3.032 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.913/3.250 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:10.976/3.313 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 11:43:11.163/3.500 Oracle Coherence GE 3.5/453 (Pre-release)

```

Input:

Messages 

Responding to Changes in the Cache

In this section, you create a Java class to modify entries in the cache and return the changed records.

Until now, to perform actions on the entries in a cache, you used the `put` and `get` operations. 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. The entries will be processed directly where the data is being held. The processing you perform may change the data: it may create, update, remove data, or only perform calculations. The processing can occur in parallel in a partitioned cache with multiple nodes, so it is scalable. Processing in the cache also saves the expense of I/O because data does not have to be retrieved to the client for processing.

`EntryProcessors` that work against the same key are logically queued. This allows lock-free (high performance) processing. 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.

Note: `EntryProcessor` classes must be available in the classpath for each cluster node.

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 change the work address of employees in the `Contacts` data set:

```
public static class OfficeUpdater extends AbstractProcessor
    implements PortableObject
    ...
    public Object process(InvocableMap.Entry entry) {
        Contact contact = (Contact) entry.getValue();
        contact.setWorkAddress(m_addrWork);
        entry.setValue(contact);
        return null;
    }
}
```

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

```
cache.invokeAll(new EqualsFilter("getHomeAddress.getState", "MA"),
    new OfficeUpdater(addrWork));
```

In this exercise, you create a Java class with `EntryProcessors` that updates entries in the cache. The `ObserverExample` class created in the previous exercise will detect these changes and display the changed records.

1. Create a class that updates entries in the cache.

In the `Loading` project, create a class called `ProcessorExample` with a `main` method that updates the address of a `Contact` object in the cache. See "[Creating a Java Class](#)" on page 2-13 if you need detailed information.

2. Write code to find the records of `Contacts` that live in Massachusetts and update their work address to an in-state office.

Hint: include a subclass that implements `PortableObject` (for serializing and deserializing data from the cache) and contains an `EntryProcessor` to set the work addresses. Use methods from the `Filter` class to isolate the `Contacts` whose home address is in Massachusetts.

[Example 5-4](#) illustrates possible code for the `PersonEventTester` class.

Example 5-4 Sample Program to Update an Object in the Cache

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

import com.tangosol.net.NamedCache;

import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.processor.AbstractProcessor;
import com.tangosol.util.InvocableMap;

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

```

import com.oracle.coherence.handson.Address;
import com.oracle.coherence.handson.Contact;

import com.tangosol.net.CacheFactory;

import java.io.IOException;

/**
 * ProcessorExample executes an example EntryProcessor.
 *
 */
public class ProcessorExample
{
    public ProcessorExample() {
    }

    public static void main(String[] args) {
        NamedCache cache = CacheFactory.getCache("ContactsCache");
        new ProcessorExample().execute(cache);
    }
    // ----- ProcessorExample methods -----

    public void execute(NamedCache cache)
    {
        // People who live in Massachusetts moved to an in-state office
        Address addrWork = new Address("200 Newbury St.", "Yoyodyne, Ltd.",
            "Boston", "MA", "02116", "US");

        cache.invokeAll(new EqualsFilter("getHomeAddress.getState", "MA"),
            new OfficeUpdater(addrWork));
    }
    // ----- nested class: OfficeUpdater -----

    /**
     * OfficeUpdater updates a contact's office address.
     */
    public static class OfficeUpdater
        extends AbstractProcessor
        implements PortableObject
    {
        // ----- constructors -----

        /**
         * Default constructor (necessary for PortableObject implementation).
         */
        public OfficeUpdater() {
        }

        public OfficeUpdater(Address addrWork) {
            m_addrWork = addrWork;
        }

        // ----- InvocableMap.EntryProcessor interface -----

        public Object process(InvocableMap.Entry entry)
        {
            Contact contact = (Contact) entry.getValue();

            contact.setWorkAddress(m_addrWork);
        }
    }
}

```

```

        entry.setValue(contact);
        return null;
    }

    // ----- PortableObject interface -----

    public void readExternal(PofReader reader)
        throws IOException
    {
        m_addrWork = (Address) reader.readObject(0);
    }

    public void writeExternal(PofWriter writer)
        throws IOException
    {
        writer.writeObject(0, m_addrWork);
    }

    // ----- data members -----

    private Address m_addrWork;
    }

```

3. Perform the following steps to test the `ObserverExample` and `ProcessorExample` classes.
 - Restart your cache server.
 - Run the `LoaderExample` class to load the cache.
 - Run the `ObserverExample` class. Do not input any value through the **Input** area at the bottom of the messages window.
 - Run the `ProcessorExample` to update records in the cache. You should see messages in `JDeveloper` indicating that addresses have been updated. This is illustrated in [Figure 5-2](#).

Figure 5–2 Output from the ObserverExample and ProcessorExample Classes

```
2009-04-21 11:49:15.976/3.031 Oracle Coherence GE 3.5/453 (Pre-relea
2009-04-21 11:49:15.976/3.031 Oracle Coherence GE 3.5/453 (Pre-relea
2009-04-21 11:49:16.163/3.218 Oracle Coherence GE 3.5/453 (Pre-relea
2009-04-21 11:49:16.195/3.250 Oracle Coherence GE 3.5/453 (Pre-relea
2009-04-21 11:49:16.773/3.828 Oracle Coherence GE 3.5/453 (Pre-relea
2009-04-21 11:49:26.710/13.765 Oracle Coherence GE 3.5/453 (Pre-rele
2009-04-21 11:49:26.835/13.890 Oracle Coherence GE 3.5/453 (Pre-rele
2009-04-21 11:49:27.351/14.406 Oracle Coherence GE 3.5/453 (Pre-rele
Work address was updated for John Kwrddulrq
Work address was updated for John Skya
Work address was updated for John Tesj
Work address was updated for John Ylnklszjb
Work address was updated for John Zzgwkf
Work address was updated for John Tswqc
Work address was updated for John Xkdek
Work address was updated for John Dauzs
Work address was updated for John Gtackyczo
2009-04-21 11:49:28.866/15.921 Oracle Coherence GE 3.5/453 (Pre-rele
2009-04-21 11:49:28.866/15.921 Oracle Coherence GE 3.5/453 (Pre-rele
2009-04-21 11:49:28.866/15.921 Oracle Coherence GE 3.5/453 (Pre-rele
2009-04-21 11:49:28.866/15.921 Oracle Coherence GE 3.5/453 (Pre-rele
Process exited.
```

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 have the database, you can download it 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, such as persistence for Java SE, 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 Oracle TopLink (based on the open source EclipseLink project).

Mapping Relational Data to Java Objects with JPA

In this exercise, 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`.

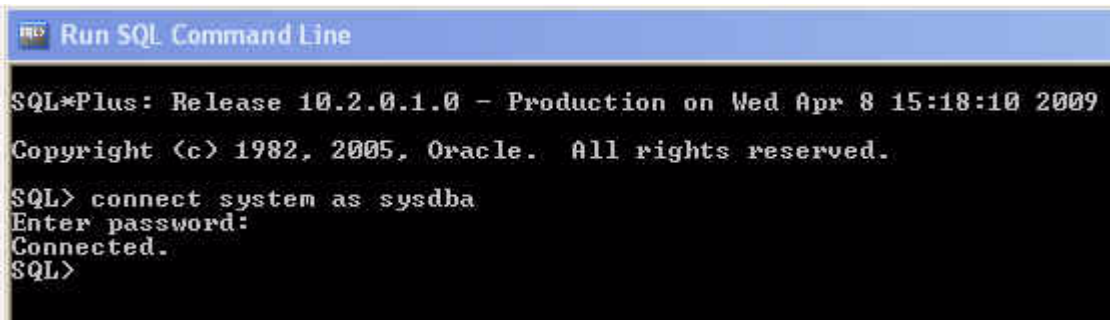
To use the Java Persistence API (JPA) to demonstrate data mapping with Coherence:

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 6–1 Connecting to the Database



```

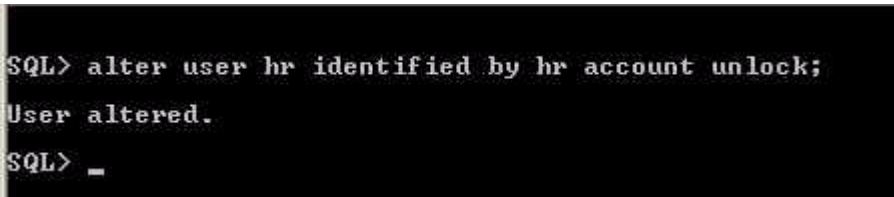
Run SQL Command Line
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Apr 8 15:18:10 2009
Copyright (c) 1982, 2005, Oracle. All rights reserved.
SQL> connect system as sysdba
Enter password:
Connected.
SQL>

```

- c. Enter the command to unlock the account:

```
alter user hr identified by hr account unlock;
```

Figure 6–2 Unlocking the Database Account



```

SQL> alter user hr identified by hr account unlock;
User altered.
SQL> _

```

2. Create a new project in JDeveloper called JPA. See "[Creating a New Project in an Existing Application](#)" on page 2-11 if you need detailed information.
 - a. Set the default **Java Options** in **Run/Debug/Profile** of the **Project Properties** to the appropriate log level and to disable local storage.

```

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

```
 - b. Ensure that the classpath points to the full path for the `coherence.jar` file:

```

C:\oracle\product\coherence\lib\coherence.jar

```
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

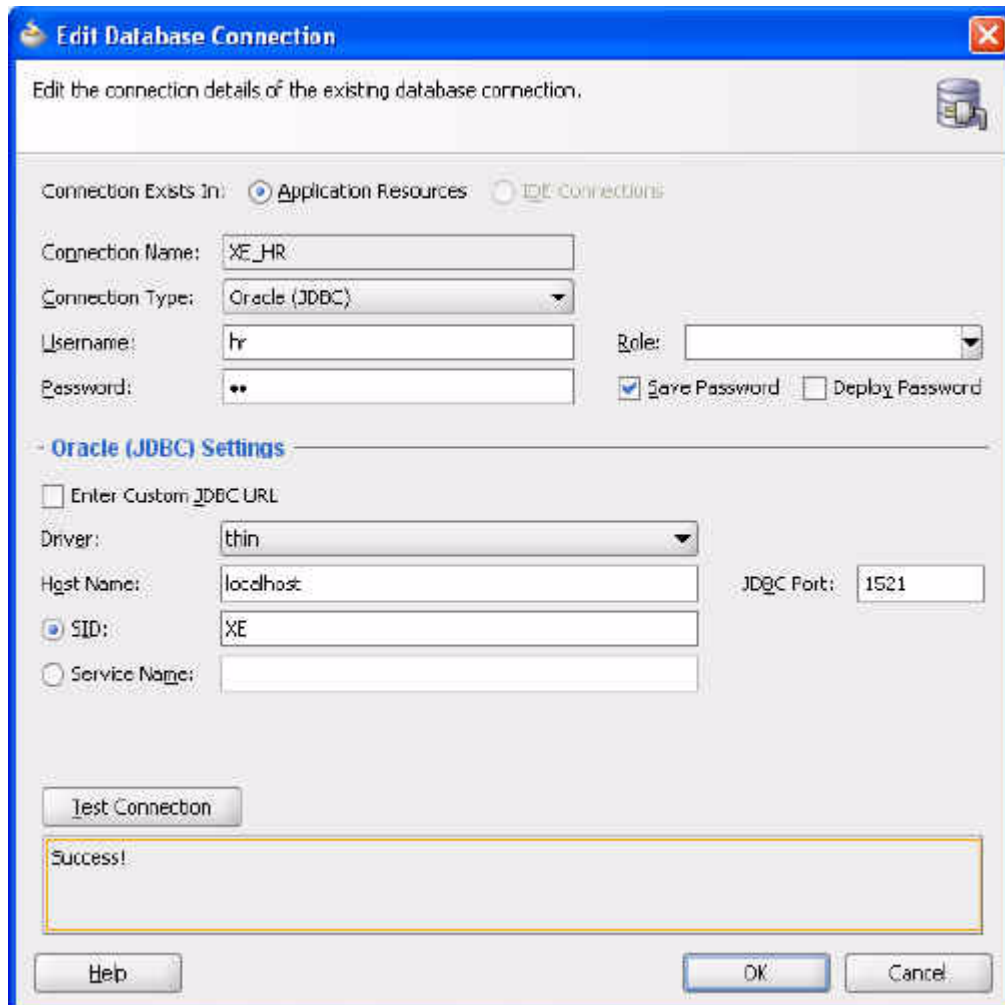
—**Password:** hr

—Click **Test Connection**.

This should display "Success!"

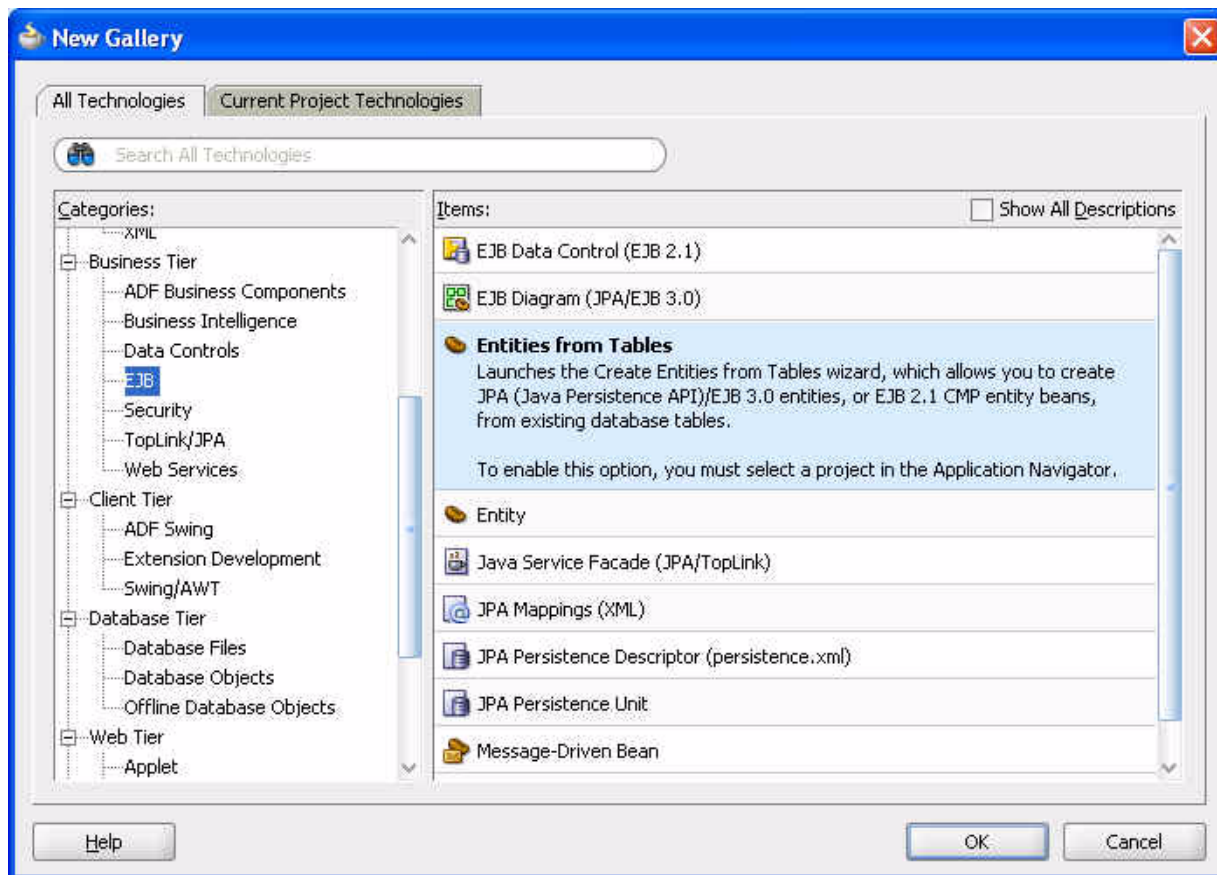
Click **OK**.

Figure 6–3 Defining the Database Connection



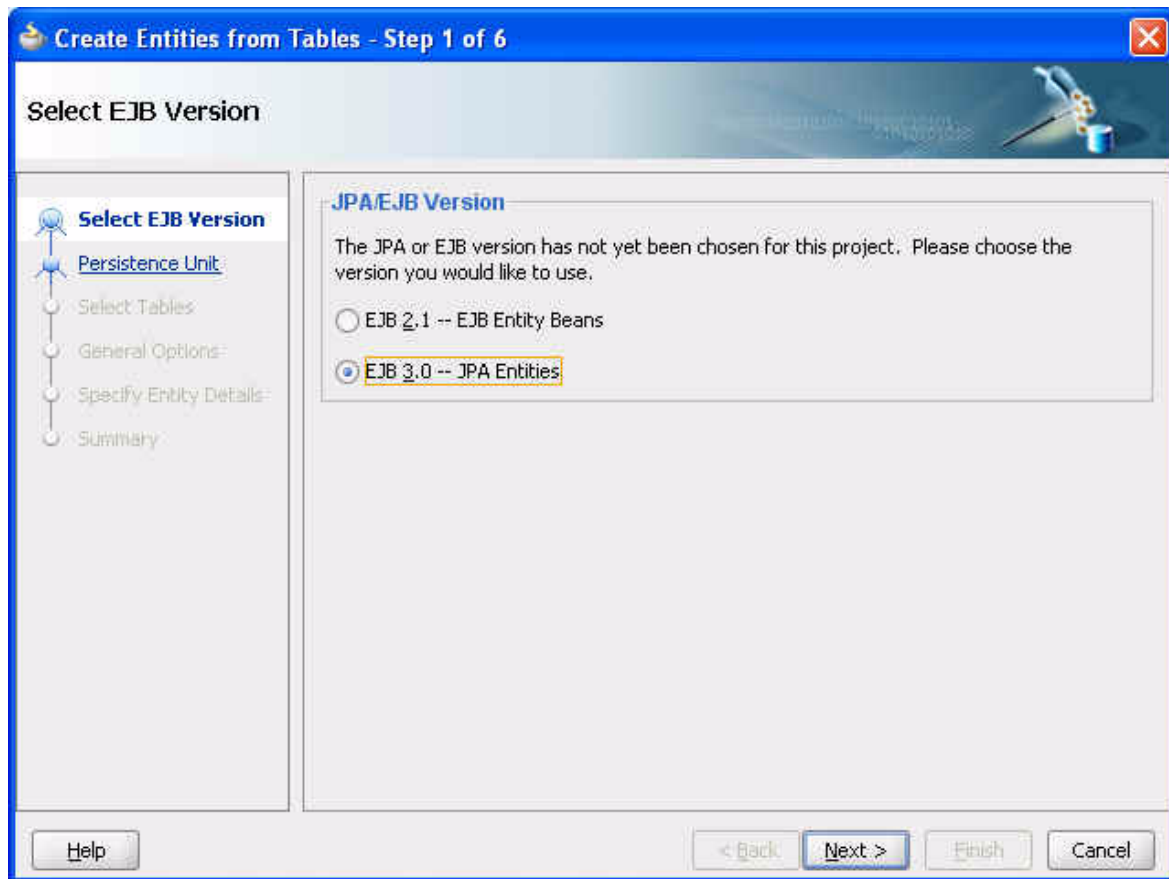
- c. Right-click the JPA project and select **New**. Under **Business Tier**, select **EJB**, and then select **Entities from Tables**. Click **OK**.

Figure 6-4 Creating EJB Entity Beans



- d. In the **Create Entities from Tables** window, select **EJB 3.0 --JPA Entities**, and then click **Next**.

Figure 6-5 Specifying the EJB Version



- e. Click **New** to create a 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: <leave blank>

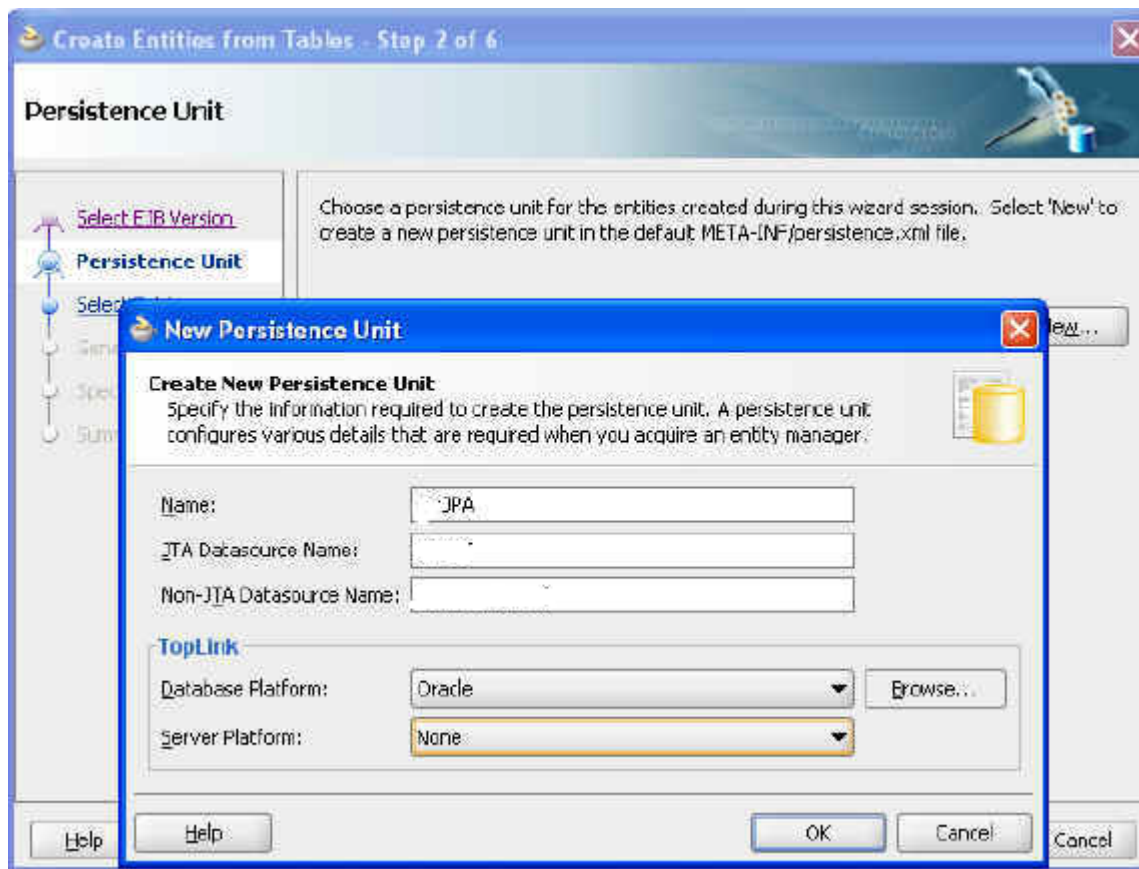
Non-JTA Datasource Name: <leave blank>

Database Platform: Oracle

Platform: None

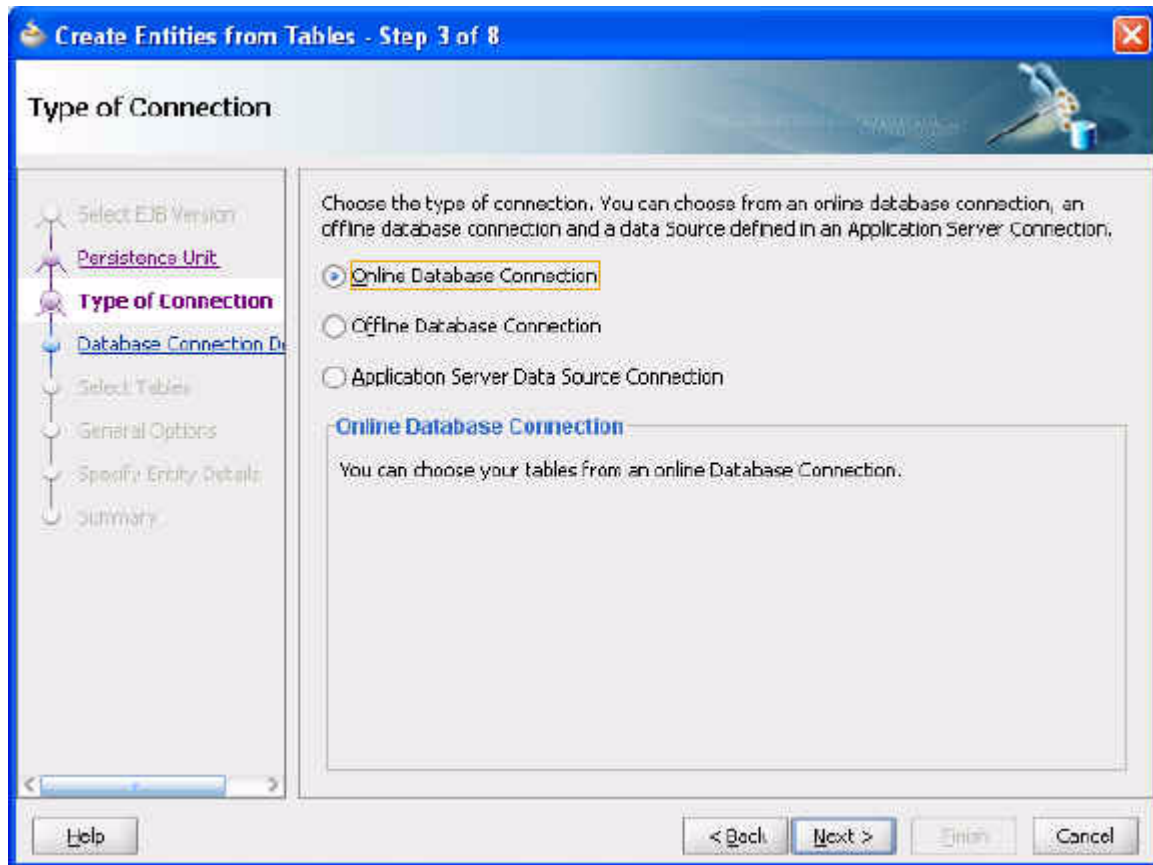
Click **OK**. When the **Create Entities from Tables** screen returns, click **Next**.

Figure 6–6 Defining the Persistence Unit

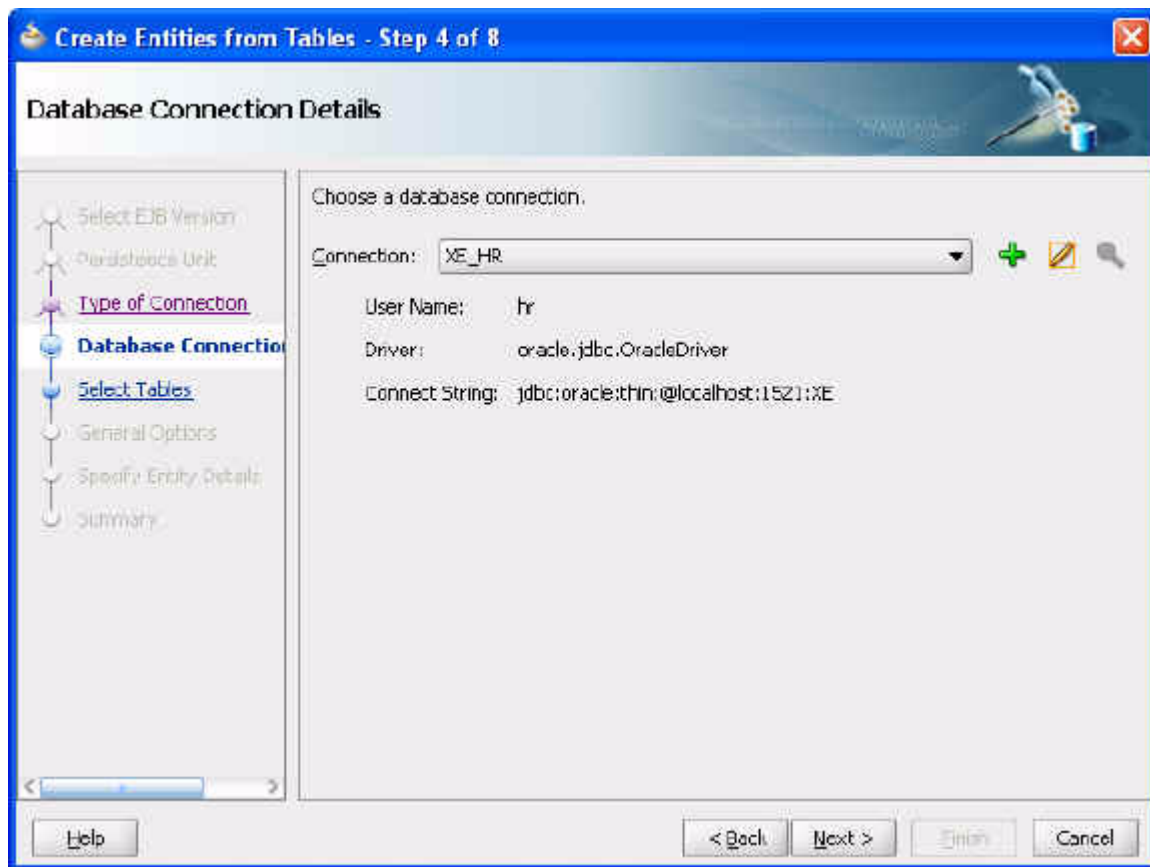


- f. Select the **Online Database Connection** option and click **Next**.

Figure 6-7 Creating Entity Beans from Table Data

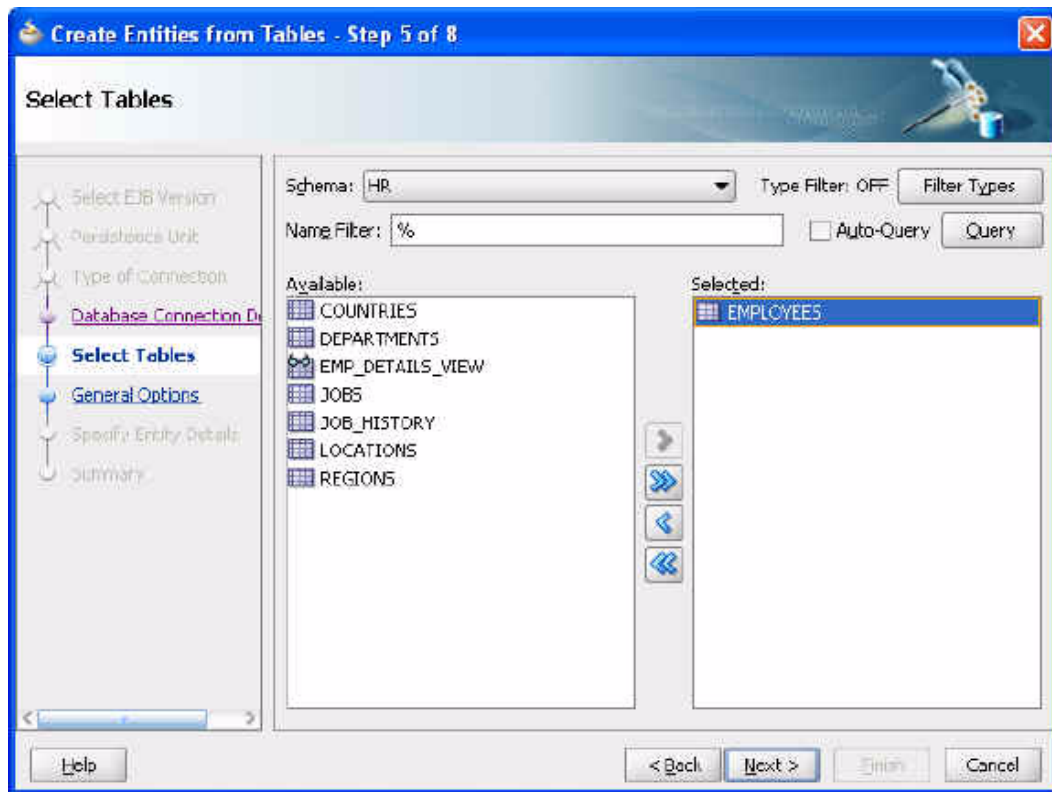


- g. In the Database Connection Details window, click Next.

Figure 6–8 Choosing the Database Connection

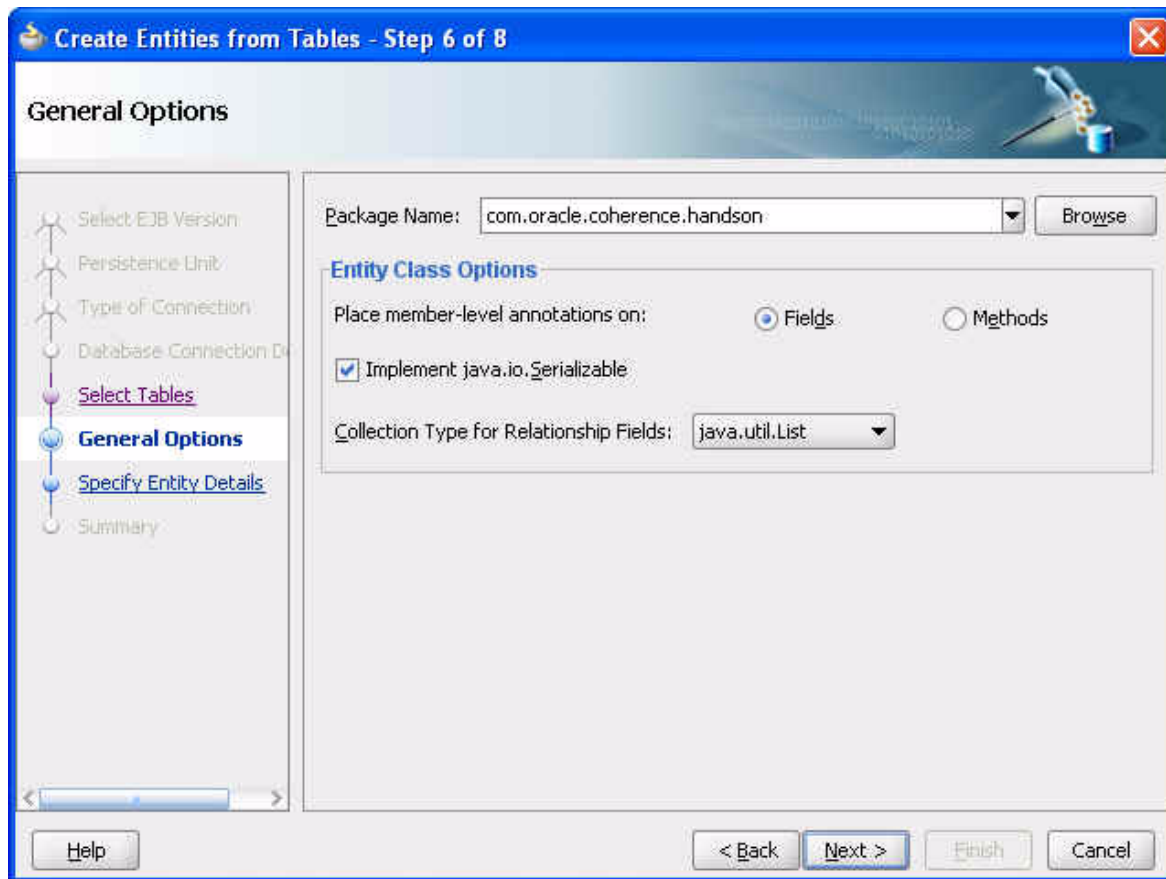
- h. Query for the EMPLOYEES table and select it as shown in [Figure 6–9](#). Click Next.

Figure 6–9 Choosing the Table Data for the Entity Bean

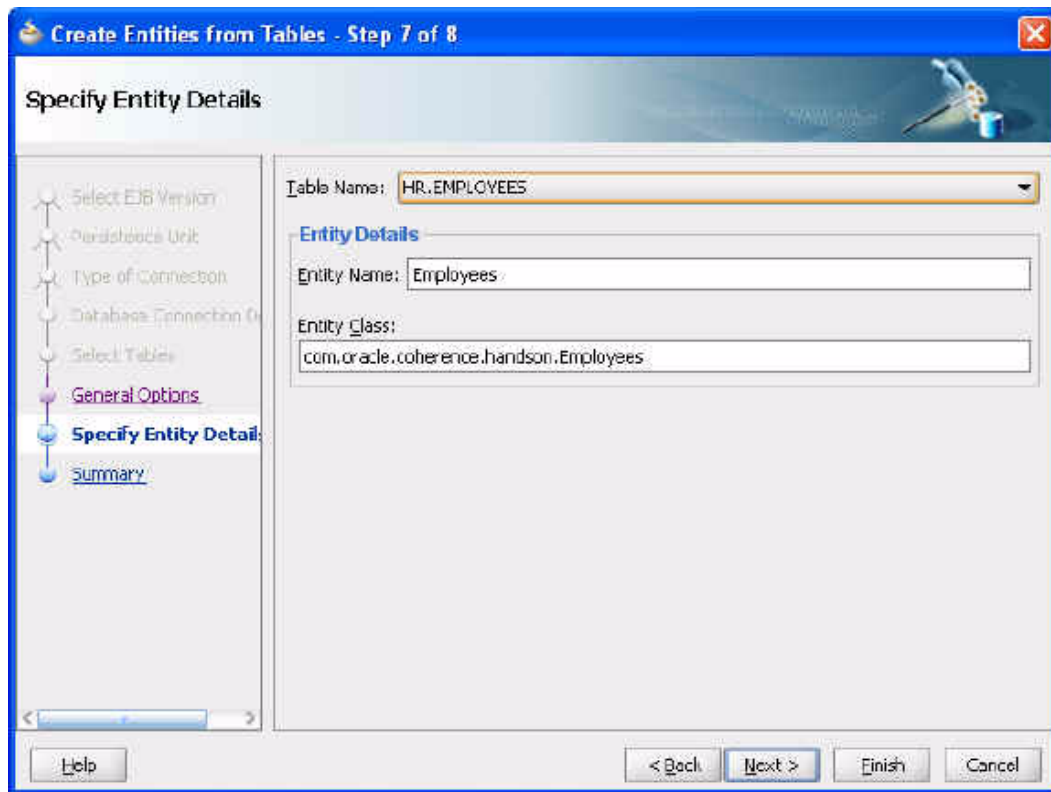


- i. Accept the default **General Options** and click **Next**.

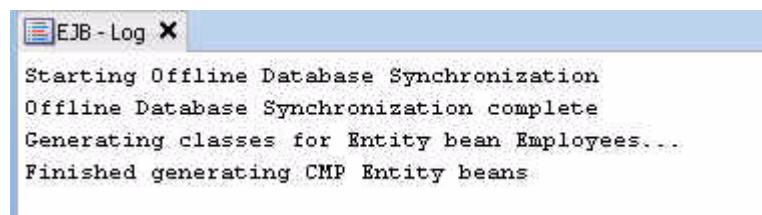
Figure 6–10 Choosing General Options for the Entity



- j. Accept the default **Entity Class Details** and click **Next**.

Figure 6–11 Specifying the Entity Details

- k. A **Summary** page appears. Click **Finish** to create the Entity bean. You should see messages similar to [Figure 6–12](#) in the EJB Log window of the navigator.

Figure 6–12 Generating EJB Entity Beans—the EJB Log Window

- i. Replace the contents of `persistence.xml` with the code in [Example 6–1](#) and save the file. Ensure that the connection details match your database connection details.

Example 6–1 *persistence.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"/>
  </properties>
</persistence-unit>
</persistence>
```

```

        <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 6-2](#). Save the file in the `home\oracle\labs\` directory.

Example 6-2 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>
        </backing-map-scheme>
      </read-write-backing-map-scheme>
    </distributed-scheme>
  </caching-schemes>

```

```

        </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`. Declare the cache configuration file in `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\JPA\classes;
```

```
C:\home\oracle\labs\JPA\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\eclipselink.jar;
```

...

[Example 6–3](#) illustrates a modified `jpa-cache-server.cmd` file:

Example 6–3 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\JPA\classes;C:\oracle\product\coherenc
e\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. Save the `jpa-cache-server.cmd` file and ensure that all other cache servers are stopped. Run `jpa-cache-server.cmd`.

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

6. Modify the JPA Project Properties.

- a. Edit the **JPA 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
```

- 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):

—TopLink predefined JDeveloper library. This will provide the required EclipseLink JARs and APIs

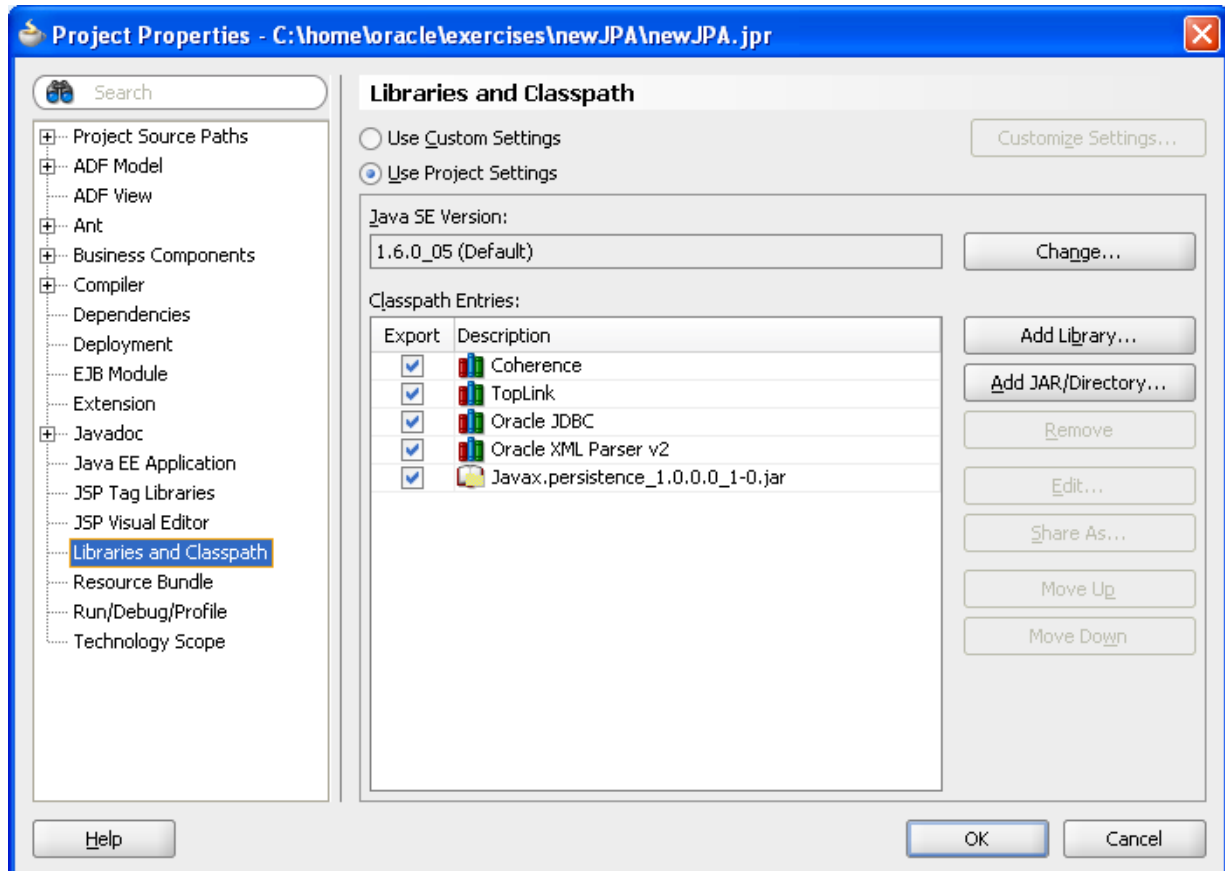
—Oracle JDBC predefined JDeveloper library for database connectivity

—Oracle XML Parser v2 predefined JDeveloper library for interpreting XML

—Java Persistence JAR file for the persistence API:
 C:\oracle\product\modules\javax.persistence_1.0.0.0_1-0.jar

The **Libraries and Classpath** screen should look similar to [Figure 6–13](#):

Figure 6–13 Adding JARs and Libraries to the Classpath



7. Create a new class in the JPA project to interact with the `Employee` object.
 - a. Create a new class with a main method called `RunEmployeeExample`. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.
 - b. Create the code to perform the following:
 - Get an employee using `EMPLOYEE_ID`. `EMPLOYEE_ID` should be a long data type.
 - Display the salary.
 - Give them a 10% pay raise.
 - Get the value again to confirm the pay raise.

[Example 6–4](#) illustrates a possible solution.

Example 6–4 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`.

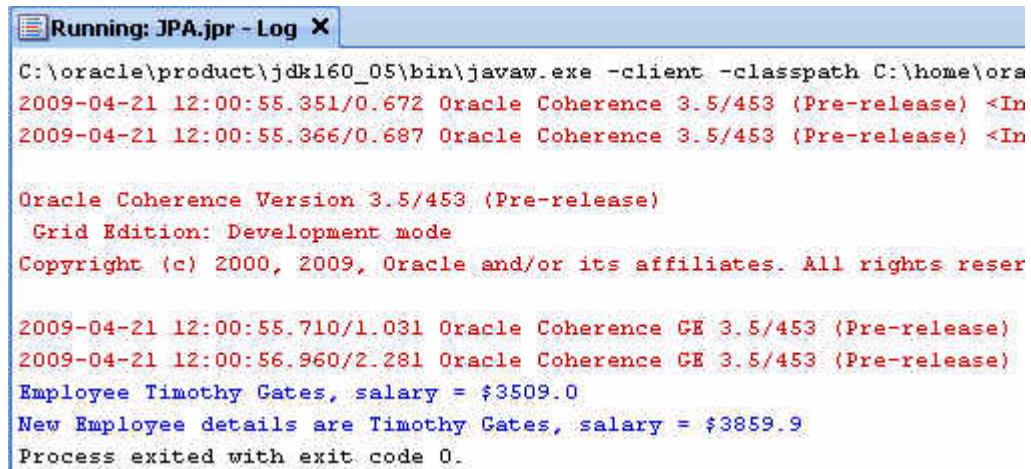
Now that the `Employees` class has been annotated to persist to the database using JPA, and you have included the `persistence.xml` file to tell JPA where your database is, Coherence employs a `CacheStore` implementation that uses JPA to load and store objects to the database. When you use the `get(Object 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.

The output should look similar to the text illustrated [Figure 6-14](#).

Figure 6–14 Results from the RunEmployeeExample Application

```
C:\oracle\product\jdk160_05\bin\javaw.exe -client -classpath C:\home\ora
2009-04-21 12:00:55.351/0.672 Oracle Coherence 3.5/453 (Pre-release) <In
2009-04-21 12:00:55.366/0.687 Oracle Coherence 3.5/453 (Pre-release) <In

Oracle Coherence Version 3.5/453 (Pre-release)
  Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reser

2009-04-21 12:00:55.710/1.031 Oracle Coherence CE 3.5/453 (Pre-release)
2009-04-21 12:00:56.960/2.281 Oracle Coherence CE 3.5/453 (Pre-release)
Employee Timothy Gates, salary = $3509.0
New Employee details are Timothy Gates, salary = $3859.9
Process exited with exit code 0.
```

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. This chapter has the following sections:

- [Introduction](#)
- [Creating a Cache Application](#)
- [Creating a Database Cache](#)

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. The drawback of replicated caches is that they require a large memory footprint as every node has a copy of every object.
- **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 7-1](#).

Table 7-1 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 <code>java.util.Map</code> interface

Creating a Cache Application

This exercise consolidates 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:

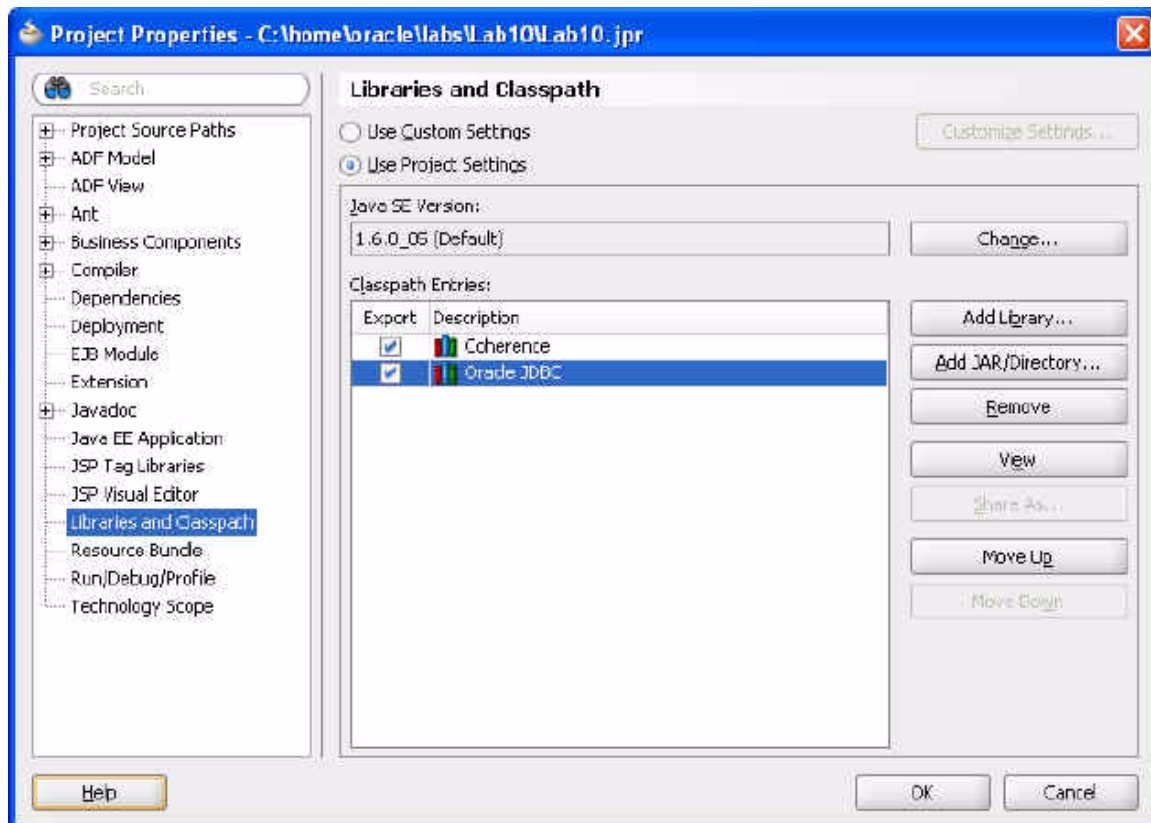
1. Create a project and an application in Oracle JDeveloper.
 - a. Create a project called `Interact` in Oracle JDeveloper. See "[Creating a New Project in an Existing Application](#)" on page 2-11 if you need detailed information.
 - b. Create a Java class, `CoherenceCache`, in the project. See "[Creating a Java Class](#)" on page 2-13 if you need detailed information.
The Java class will be used to create a Coherence cache.
 - c. Create an XML document, `cache-config.xml`, as the cache configuration deployment descriptor.

To add the XML document, right-click the project and choose **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.

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 7-1 Adding the Oracle JDBC Libraries to the Classpath

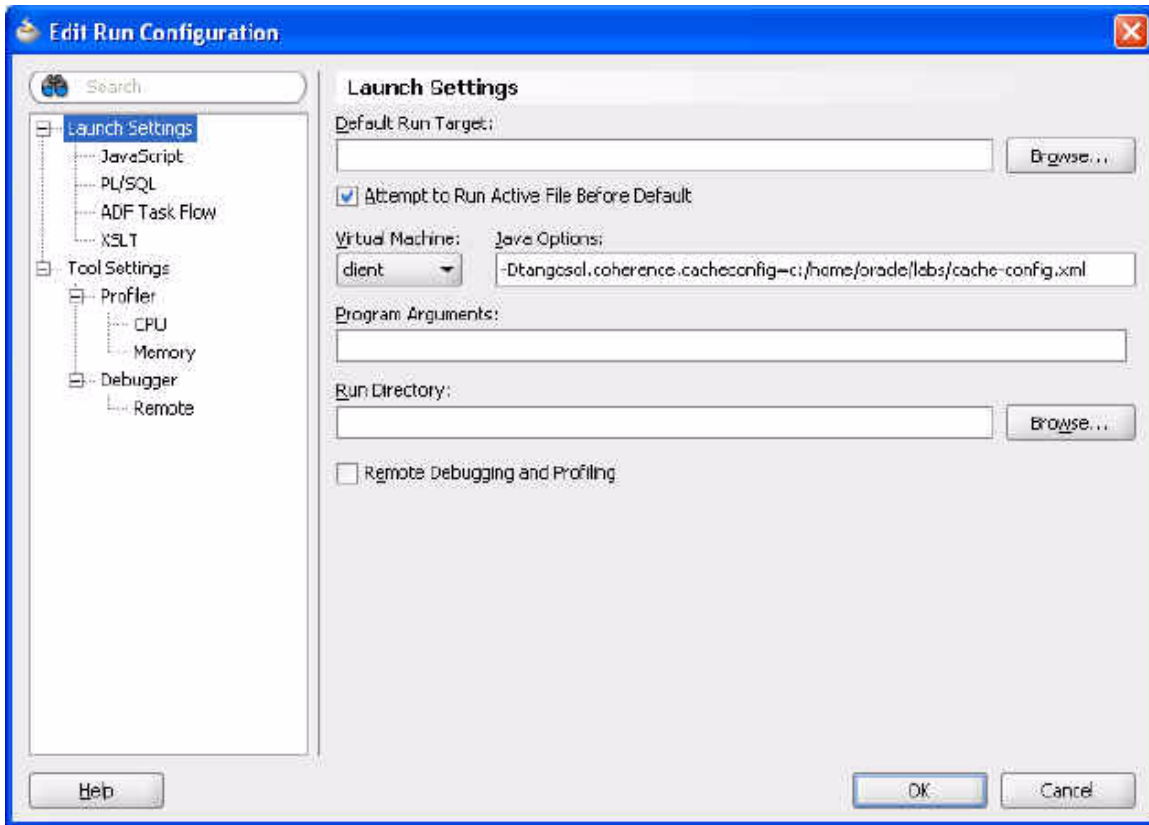


3. Modify the Run configuration for the application to add the cache configuration file as a run-time Java option.
 - a. Select the project node and select **Tools > Project Properties**. In the **Project Properties** window, select **Run/Debug/Profile**. The **Default** configuration is selected by 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/cache-config.xml

Figure 7–2 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 7–1](#). Copy the contents of this example to the cache-config.xml file in Oracle JDeveloper.

Example 7–1 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>
</cache-config>
```

```

        </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 7-2 illustrates a possible solution. You can copy the code to the `CoherenceCache` application in Oracle JDeveloper.

Example 7-2 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 6, "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.

Example 7-3 Output of the Coherence Cache Application

```

2009-04-21 14:32:59.272/0.703 Oracle Coherence 3.5/453b (Pre-release) <Info>
(thread=main, member=n/a): Loaded operational configuration from resource
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2009-04-21 14:32:59.272/0.703 Oracle Coherence 3.5/453b (Pre-release) <Info>
(thread=main, member=n/a): Loaded operational overrides from resource "jar:file:/
C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.
xml"
2009-04-21 14:32:59.272/0.703 Oracle Coherence 3.5/453b (Pre-release) <D5>
(thread=main, member=n/a): Optional configuration override "/"
tangosol-coherence-override.xml" is not specified
2009-04-21 14:32:59.288/0.719 Oracle Coherence 3.5/453b (Pre-release) <D5>
(thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is
not specified

Oracle Coherence Version 3.5/453b (Pre-release)
  Grid Edition: Development mode
  Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.

2009-04-21 14:32:59.600/1.031 Oracle Coherence GE 3.5/453b (Pre-release) <Info>
(thread=main, member=n/a): Loaded cache configuration from file
"C:\home\oracle\exercises\cache-config.xml"
2009-04-21 14:33:00.928/2.359 Oracle Coherence GE 3.5/453b (Pre-release) <D5>
(thread=Cluster, member=n/a): Service Cluster joined the cluster with senior
service member n/a

```



```

2009-04-21 14:33:01.131/2.562 Oracle Coherence GE 3.5/453b (Pre-release) <Info>
(thread=Cluster, member=n/a): This Member(Id=2, Timestamp=2009-04-21 14:33:00.944,
Address=130.35.99.50:8089, MachineId=49714, Location=site:us.oracle.
com,machine:tpfaeffl-pc,process:5124, Role=OracleCoherenceCoherenceCache,
Edition=Grid Edition, Mode=Development, CpuCount=1, SocketCount=1) joined cluster
"cluster:0x2EDB" with senior Member(Id=1, Timestamp=2009-04-21 14:31:32.585,
Address=130.35.99.50:8088, MachineId=49714, Location=site:us.oracle.
com,machine:tpfaeffl-pc,process:5148, Role=CoherenceServer, Edition=Grid Edition,
Mode=Development, CpuCount=1, SocketCount=1)
2009-04-21 14:33:01.194/2.625 Oracle Coherence GE 3.5/453b (Pre-release) <D5>
(thread=Cluster, member=n/a): Member 1 joined Service Management with senior
member 1
2009-04-21 14:33:01.194/2.625 Oracle Coherence GE 3.5/453b (Pre-release) <D5>
(thread=Cluster, member=n/a): Member 1 joined Service JpaDistributedCache with
senior member 1
2009-04-21 14:33:01.350/2.781 Oracle Coherence GE 3.5/453b (Pre-release) <D5>
(thread=Invocation:Management, member=2): Service Management joined the cluster
with senior service member 1
2009-04-21 14:33:01.835/3.266 Oracle Coherence GE 3.5/453b (Pre-release) <D5>
(thread=DistributedCache, member=2): Service DistributedCache joined the cluster
with senior service member 2
Hello Cache
2009-04-21 14:33:01.975/3.406 Oracle Coherence GE 3.5/453b (Pre-release) <D4>
(thread=ShutdownHook, member=2): ShutdownHook: stopping cluster node
Process exited with exit code 0.

```

Creating a Database Cache

In this section, you create a cache backed by the Oracle database. This is also referred to as an "Oracle database cache".

1. Create an Oracle database cache.

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 7-4 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.

[Example 7-5](#) illustrates the output from the script.

Example 7-5 Running the SQL Script for Creating a Database Table

```

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Apr 21 14:45:53 2009
Copyright (c) 1982, 2005, Oracle. All rights reserved.
SQL> connect hr/hr;

```

```
Connected.
SQL> @/home/oracle/labs/dbscript.sql
```

```
Table created
```

```
1 row created
```

```
1 row created
```

2. Create a custom CacheStore.

- a. Create a Java class `DBCacheStore` in Oracle JDeveloper. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.
- b. Create the code to connect to the database and get table data.

[Example 7–6](#) 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 7–6 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 7-7](#) and Replace the existing code in the `cache-config.xml` file in Oracle JDeveloper.

Example 7-7 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>
  </distributed-scheme>
</caching-schemes>
</cache-config>

```

```

<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 7-2](#):

Table 7-2 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.

Table 7–2 (Cont.) Types of Read-Write Caching Supported by Coherence

Types of Read-Write Caching	Action
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.

3. Create a Java class `DatabaseCache` for the database cache in Oracle JDeveloper.

The class must contain a main method. See ["Creating a Java Class"](#) on page 2-13 if you need detailed information.

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 7–8](#).

Example 7–8 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 must be created before querying, and 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);
```
4. Stop any running cache servers. Start the JPA cache server (`jpa-cache-server.cmd`) that you created in [Chapter 6, "Using JPA with Coherence"](#).
 5. Right-click the `DatabaseCache` application in Oracle JDeveloper and select **Run**. [Figure 7-3](#) illustrates the expected results.

Figure 7-3 Results from Running the DatabaseCache Application

```
Oracle Coherence Version 3.5/453 (Pre-release)
  Grid Edition: Development mode
Copyright (c) 2000, 2009, Oracle and/or its affiliates. All rights reserved.

2009-04-21 12:09:45.351/1.078 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.351/2.078 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.398/2.125 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.398/2.125 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.773/2.500 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.788/2.515 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.788/2.515 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:46.866/2.593 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:47.398/3.125 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:48.382/4.109 Oracle Coherence GE 3.5/453 (Pre-release)
Catalog ID: catalog3, Title: Tuning Grid Management
Catalog ID: catalog4, Title: Tuning Coherence
Catalog ID: catalog5, Title: Tuning Database
2009-04-21 12:09:50.101/5.828 Oracle Coherence GE 3.5/453 (Pre-release)
2009-04-21 12:09:50.101/5.828 Oracle Coherence GE 3.5/453 (Pre-release)
Process exited with exit code 0.
```

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.

6. 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;
```

[Example 7-9](#) illustrates the results.

Example 7-9 Output from the select Command

```
....  
SQL> select * from hr.catalog;
```

```
-----
```

```
VALUE
```

```
-----
```

```
catalog3  
Tuning Grid Management  
catalog4  
Tuning Coherence
```

```
catalog5  
Tuning Database
```

```
ID
```

```
-----
```

```
VALUE
```

```
-----
```

```
catalog1  
Tuning Undo Tablespace
```

```
catalog2  
Tuning Your View Objects
```

Caching HTTP Sessions with Coherence*Web

The following example demonstrates how to use Coherence*Web to cache session information for Web application instances that are deployed across WebLogic application servers. To do this, you will create a Web application and deploy it to two application server instances. The application is a simple counter that stores the current count as a session attribute. Coherence*Web automatically serializes and replicates the attribute across both server instances. Lastly, a browser is used to access each application instance to demonstrate that the same session attribute is used among the instances.

To complete this example, the following software must be installed:

- Oracle Coherence 3.5
- WebLogic 10.X (This example uses 10.3. WebLogic 8.X and 9.X are also supported.)

Install Coherence*Web on WebLogic 10.X

The Coherence*Web module includes a plug-in installer that supports WebLogic 9.2 MP1 and 10.3. For more information on Coherence*Web, including installation and configuration instructions, see the *User's Guide for Oracle Coherence*Web*.

Start a Cache Server

Start a Coherence Cache Server. [Example 8–1](#) illustrates a sample script to start the server.

Example 8–1 Script to Start the Cache Server

```
setlocal

if (%COHERENCE_HOME%)==() (
    set COHERENCE_HOME=c:\oracle\product\coherence
)

set COH_OPTS=%COH_OPTS% -server -cp
c:\oracle\product\coherence\lib\coherence.jar;c:\oracle\product\coherence\lib\coherence-web-spi.war;
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.management.remote=true
-Dtangosol.coherence.cacheconfig=/WEB-INF/classes/session-cache-config.xml
-Dtangosol.coherence.session.localstorage=true

java %COH_OPTS% -Xms512m -Xmx512m com.tangosol.net.DefaultCacheServer
```

```
:exit
```

Configure the WebLogic Server

This example requires two application server instances:

1. Run the Oracle WebLogic Configuration Wizard (**Start>All Programs> Oracle Fusion Middleware>WebLogic Server 10.3>Tools>Configuration Wizard**) to create a WebLogic domain called `test_domain`.

Before exiting the wizard, select the **Start Admin Server** check box, and click **Done**. The configuration wizard automatically starts the administration server.

2. Start the Server Administration Console.

From the browser, log in to the Oracle WebLogic Server Administration Console using the following URL: `http://hostname:7001/console`. The console starts, and the domain home page displays.

3. Create a machine on which to run the servers.

From the Domain Structures window, click **Environment>Machines**. Click **New**. The **Create a New Machine** page displays. Enter a name for the machine (in this case, **Test**) and click **OK**.

Figure 8–1 Creating a New Machine

Create a New Machine

OK Cancel

Machine Properties

The following properties will be used to identify your new Machine.

* Indicates required fields

What would you like to name your new Machine?

* **Name:**

Specify the type of machine operating system.

Machine OS:

OK Cancel

The **Summary of Machines** page should look similar to [Figure 8–2](#).

Figure 8–2 Summary of Machines

Summary of Machines

A machine is the logical representation of the computer that hosts one or more WebLogic Server instances (servers). WebLogic Server uses configured machine names to determine the optimum server in a cluster to which certain tasks, such as HTTP session replication, are delegated. The Administration Server uses the machine definition in conjunction with Node Manager to start remote servers.

This page displays key information about each machine that has been configured in the current WebLogic Server domain.

[Customize this table](#)

Machines

Showing 1 to 1 of 1 Previous | Next

<input type="checkbox"/> Name <small>⬆</small>	Type
<input type="checkbox"/> Test	Machine

Showing 1 to 1 of 1 Previous | Next

4. Create servers associated with the machine.

Click the name of the machine in the **Summary of Machines** page to open the **Settings for <machine>** page. Click the **Servers** tab then **Add** to create a server.

5. Select **Create a new server and associate it with this machine** in the **Add a Server to Machine** page, and click **Next**.
6. Provide details about the server in the **Add a Server to Machine** page.

Enter **ServerA** as the **Server Name** and **8080** as the **Server Listen Port**. Enter the appropriate value for the **Server Listen Address**. Click **Finish**.

Figure 8–3 Adding a Server to a Machine

Add a Server to Machine

Back Next Finish Cancel

Server Properties

The following properties will be used to identify your new server:

* Indicates required fields

What would you like to name your new server?

* **Server Name:**

Where will this server listen for incoming connections?

Server Listen Address:

* **Server Listen Port:**

Back Next Finish Cancel

7. When you are returned to the **Settings for machine** page, repeat the previous three steps to create a second server.
Enter `ServerB` as the **Server Name** and `8081` as the **Server Listen Port**. Enter the appropriate value for the **Server Listen Address**. Click **Finish**.
8. Expand **Environment** in the **Domain Structure** menu and click **Servers**.
The **Summary of Servers** page displays and should be similar to [Figure 8–4](#):

Figure 8–4 Summary of Servers Page

Summary of Servers

Configuration **Control**

A server is an instance of WebLogic Server that runs in its own Java Virtual Machine (JVM) and has its own configuration.

This page summarizes each server that has been configured in the current WebLogic Server domain.

[Customize this table](#)

Servers (Filtered - More Columns Exist)

New Clone Delete Showing 1 to 3 of 3 Previous Next

<input type="checkbox"/>	Name	Cluster	Machine	State	Health	Listen Port
<input type="checkbox"/>	examplesServer(admin)			RUNNING	OK	7001
<input type="checkbox"/>	ServerA			SHUTDOWN		8080
<input type="checkbox"/>	ServerB			SHUTDOWN		9081

New Clone Delete Showing 1 to 3 of 3 Previous Next

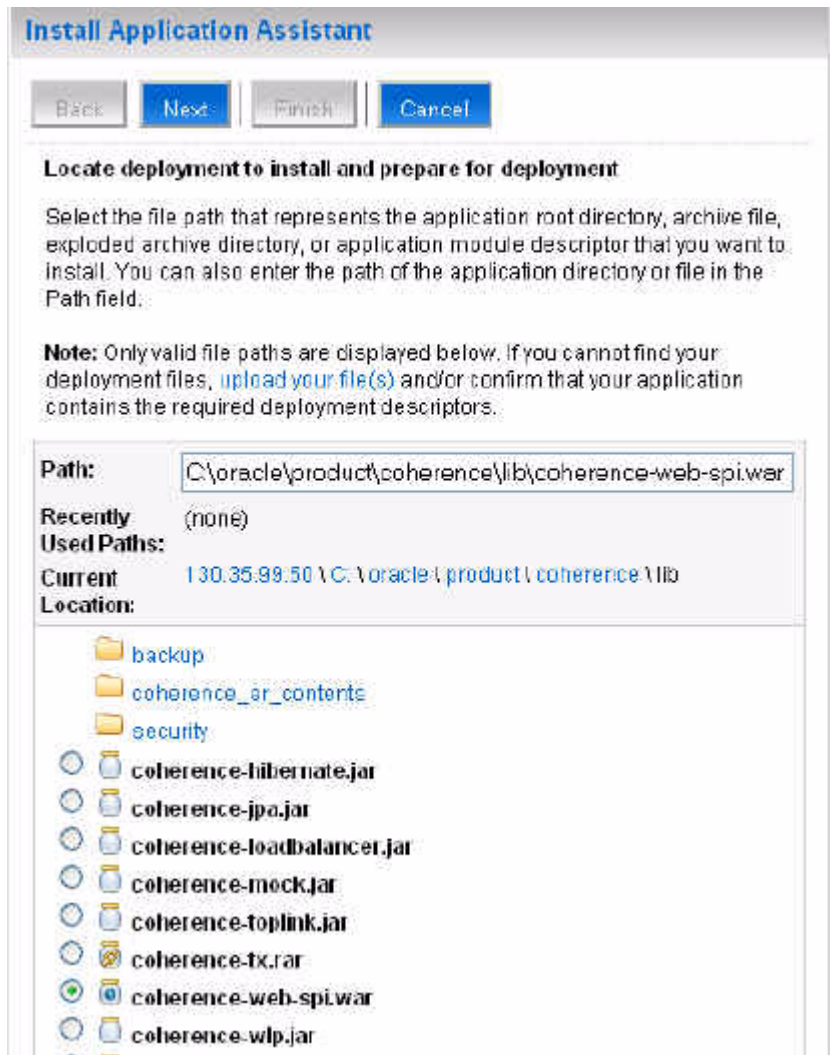
9. Start the Node Manager from **Start>All Programs>Oracle Fusion Middleware>WebLogic Server10.3>Tools>Node Manager**.
10. From the **Summary of Servers** screen, click the **Control** tab and start both server instances.

Deploy the coherence-web-spi JAR File

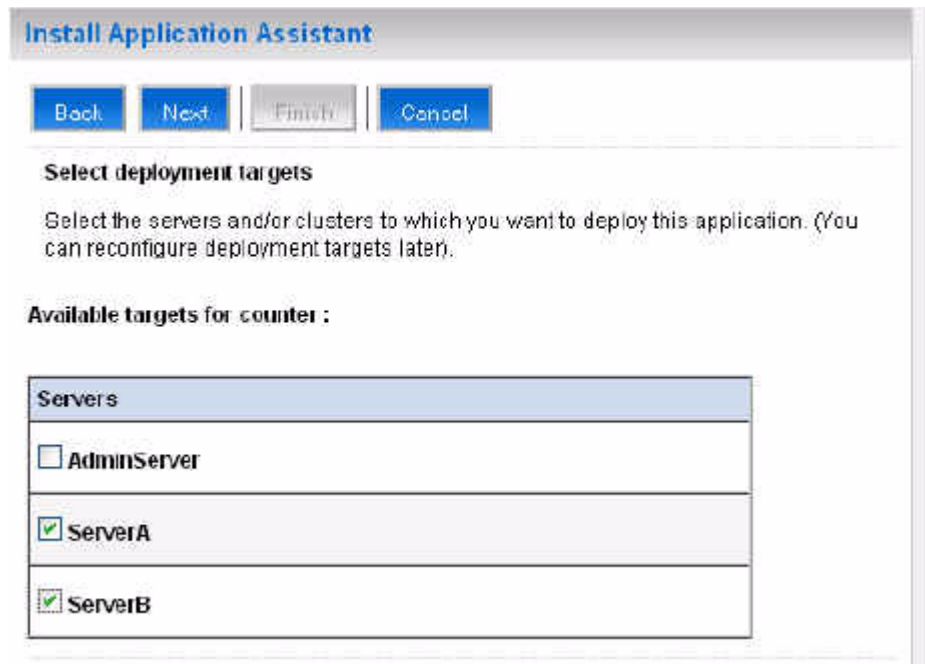
The current release of Coherence provides a deployable shared library, `coherence-web-spi.jar`, that contains a native plug-in to WebLogic Server's HTTP Session Management interface.

To deploy the `coherence-web-spi.jar` file:

1. From the **Domain Structure** menu, click **Deployments**. The **Summary of Deployments** page displays.
2. Click **Install**. The **Install Application Assistant** screen displays.
3. Use the **Install Application Assistant** to deploy `coherence-web-spi.jar` to both `ServerA` and `ServerB`.
 - a. Locate the `coherence-web-spi.jar` file.

Figure 8-5 Selecting the coherence-web-spi.jar File for Deployment

- b. Select ServerA and ServerB as the deployment targets.

Figure 8-6 Choosing the Deployment Servers

4. You can click **Finish** to skip the rest of the steps in the **Install Application Assistant**. The **Summary of Deployments** page displays after the application is deployed.

Create the Counter Web Application

The Counter Web application is a simple counter implemented as a JSP. The counter is stored as an HTTP session attribute and increments each time the page is accessed.

To create the Counter Web application:

1. Create a standard Web application directory as follows:

```
/
 /WEB-INF
```

2. Copy the following code to a text file and save it as `web.xml` to the `/WEB-INF` directory.:

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  xmlns="http://java.sun.com/xml/ns/j2ee" version="2.5">
  <description>Empty web.xml file for Web Application</description>
</web-app>
```

3. Copy the following code for the counter JSP to a text file and save the file as `counter.jsp` in the root of the Web application directory.

```
<h3>
Counter :
<%
Integer counter = new Integer(1);
HttpSession httpsession = request.getSession(true);
```

```

        if (httpSession.isNew()) {
            HttpSession.setAttribute("count", counter);
            out.println(counter);
        } else {
            int count = ((Integer)
HttpSession.getAttribute("count")).intValue();
            HttpSession.setAttribute("count", new Integer(++count));
            out.println(count);
        }
    }
}
%>
</h3>

```

4. Add a library reference for the coherence-web-spi jar file to the weblogic.xml file. Save the file in the root of the Web application directory.

```

<weblogic-web-app>
...
  <library-ref>
    <library-name>coherence-web-spi</library-name>
    <specification-version>1.0.0.0</specification-version>
    <implementation-version>1.0.0.0</implementation-version>
    <exact-match>>false</exact-match>
  </library-ref>
...
</weblogic-web-app>

```

5. The Web application directory should appear as follows:

```

/
/counter.jsp
/weblogic.xml
/WEB-INF/web.xml

```

6. ZIP or JAR the Web application directory and save the file as counter.war.

Deploy the Application

To deploy the counter.war application:

1. Open the **Summary of Deployments** page by clicking **Deployments** in the **Domain Structure** menu in the Oracle WebLogic Server Administration Console.
2. Click **Install**. The **Install Application Assistant** screen displays.
3. Use the **Install Application Assistant** to deploy counter.war to both ServerA and ServerB. The **Summary of Deployments** page displays after the application is deployed. [Figure 8-7](#) illustrates the deployments table with the counter Web application.

Figure 8–7 Deployments Window Showing the Deployed Application

Deployments

Install Update Delete Start Stop Showing 1 to 16 of 16 Previous | Next

<input type="checkbox"/>	Name	State	Health	Type	Deployment Order
<input type="checkbox"/>	apache_xbean.jar	Active		Library	100
<input type="checkbox"/>	asyncServletEar	Active	OK	Enterprise Application	100
<input type="checkbox"/>	coherence-web-spi(1.0.0.0,1.0.0.0)	Active		Library	100
<input type="checkbox"/>	counter	Active	OK	Web Application	100

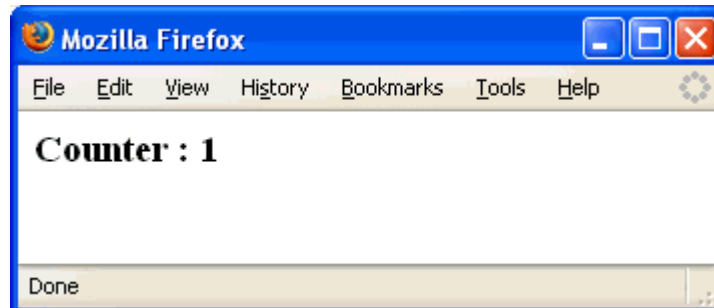
Verify the Example

To verify the example:

1. Open a browser and access the ServerA Counter instance using the following URL:

```
http://host:8080/counter/counter.jsp
```

The counter page displays and the counter is set to 1 as follows:

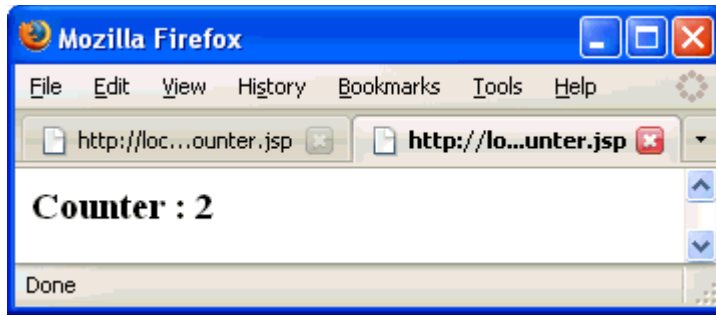
Figure 8–8 Counter Page with Counter Set to 1

2. From the same browser (or in a new browser tab), access the ServerB Counter instance using the following URL:

```
http://host:8081/counter/counter.jsp
```

The counter page displays and the counter is incremented to 2 based on the session data: If you refresh the page, the counter is incremented to 3. Refresh the instance on ServerA and the counter is at 4.

Figure 8–9 Counter Page with Counter Set to 4



Coherence Client Application Commands

This appendix describes the commands that can be issued to the Coherence command line application `coherence.cmd` (or `coherence.sh` on UNIX).

bye

Gracefully shuts down the cluster member and closes the Coherence command-line application. To test "ungraceful" shutdown, use `Ctrl-C` (Windows or UNIX) or `kill -9 <pid>` on UNIX.

bulkput <# of iterations> <block size> <start key>

Populates the current `NamedCache` with the specified number of objects. This is best described with an example:

```
bulkput 1000 10000 1
```

This will put 1000 objects of size 10000 bytes (10 MB total) starting with key 1 into the cache. The type of the key will be `java.lang.Integer` and the type of the value is `byte[]`. (The one exception is for the size zero, the value is of type `Coherence$DebugCacheItem`, which is used to test custom serialization and `ClassLoader` issues.)

clear

Clears the current `NamedCache` by calling `clear()`.

destroy

Calls `destroy()` on the current `NamedCache`. This will remove the `NamedCache` completely from the cluster, causing other cluster members to receive an `IllegalStateException` if they are holding on to a reference to the same cache and try to do any operation on that cache.

get <key>

Returns the value associated with this `<key>` from the cache by calling `get(key)`.

hash

Returns a hash value of the `NamedCache`. This is similar to a checksum and is mainly used for verifying the equivalency of the replicated caches.

help

Prints the list of commands available to you.

inc <key> [<count>]

Increments the specified key the specified number of times, using locking to ensure that no "missing updates" occur. For example:

```
inc counter 10000
```

This will increment the `counter` key 10000 times. If you run this command simultaneously on 10 machines, the value of the `counter` key will be 100,000 (assuming it started at 0 or did not exist previously).

kill

Shuts down the current `com.tangosol.net.CacheService` in an orderly (graceful) manner. This does not affect other service members.

list [<map name>]

Lists the contents of the specified `NamedCache`, or the current one if none is specified.

Note:

- If there are more than 50 items in the `NamedCache` only the first 50 items will be displayed through the `list` command.
- Java implements the `toString` method for `byte[]` values as `[` (that is, `array`) + `B` (that is, `byte`) + `hashCode`. For example:

```
[B@701a1e
```

listen [('start' | 'stop')] [('cluster' | 'local')]

Registers (`start`) or unregisters (`stop`) a debug implementation of `com.tangosol.util.MapListener` with the current `NamedCache` (`cluster`) or with the "backing-store" `com.tangosol.util.ObservableMap` (`local`). (The backing store is the `java.util.Map` object obtained by the cache service from the `LocalCacheFactory` interface.)

For example, with the distributed cache service, this command will show all cache changes that this member manages:

```
listen start local
```

This command will show all cache changes, even if they are managed by a different member, and even if this member has `local-storage-enabled` set to `false`.

```
listen start cluster
```


lock <key> [<timeout>|]

Locks the specified key. If the lock cannot be obtained immediately, it returns `false`, unless a time-out has been specified. For example, this command attempts to lock the counter key for up to 10 seconds:

```
lock counter 10001
```

log (<size> | <message>) [<iterations> [<level>]]

Prints a message of specified <size> or specified <message> a specified number of times (<iterations>) at a given <level> where `level=1` is `ERROR`, `level=2` is `WARNING` and `level=3` is `INFO`. This command is used mainly for batch testing, or to verify the format of messages when changes are made to the message format options in the `tangosol-coherence.xml` file.

map [(['Optimistic' | 'Replicated' | 'Distributed') ':'] <map name> [COH33UG:<limit>]]

Creates (if it does not already exist) the specified `NamedCache` and makes it the "current" cache. The <limit> parameter is only used in conjunction with the Optimistic Cache and specifies the maximum number of items in a Local Cache (size-limited `com.tangosol.util.Cache`). For example, the following command creates a Replicated Cache of the name `Test` and makes it the current (that is, active) `NamedCache` for subsequent commands:

```
map Replicated:Test
```

maps

Lists all the `NamedCache(s)` for the current service, `com.tangosol.net.CacheService`, regardless of whether they were created by this member.

memory

Lists the "total" used and "free" memory values for this JVM.

put <key> <value>

Puts the specified key/value pair into the current `NamedCache` using the `put(key, value)` method.

release

Releases the reference to the current `NamedCache` on this cluster member by calling the `release()` method. This does not affect other cluster nodes. If subsequent commands are issued on this cluster node against the current `NamedCache`, an `IllegalStateException` will occur.

remove <key>

Removes the specified key from the current `NamedCache` using the `remove(key)` method.

scan <start key> <stop key>

This method is used to check for missing keys in a sequence. For example, the following command checks the `java.lang.Integer` keys from 1 to 1000 inclusive, and prints out any keys in that range that are not present in the current `NamedCache`:

```
scan 1 1000
```

services

Lists all the services (`com.tangosol.net.Service`) known to this cluster. Some of these services may not be currently running on this cluster node.

size

Shows the count of objects in the current `NamedCache`.

unlock <key>

Unlocks the specified key in the current `NamedCache` using the `unlock(key)` method.

waitkey <start key> <stop key>

This method is used in conjunction with multiple Coherence command line processes and the `bulkput` command to time cache replication. Before one machine starts its `bulkput` command, a second machine is told to wait for the range of keys that will be placed into the cache.

For example, if you run these commands on the first JVM:

```
clear  
waitkey 1 1000
```

Then run this command on the second JVM:

```
bulkput 1000 100 1
```

The first JVM will display the total replication time for 1000 entries with values of 100 bytes.

who

Lists information about each `Member` (`com.tangosol.net.Member`) in the cluster.

whoami

Lists the current `Service` (`com.tangosol.net.Service`) information.

worker <command>

Performs the specified command on a new temporary thread.

#<repeat count> <command>

Repeats the following <command> the specified number of times. Multiple commands can be delimited with a semicolon. For example, in a loop of ten iterations, this command increments the `Counter` key, then lists the current `NamedCache` values:

```
#10 inc Counter; list
```

@<command>

Silent mode. Suppresses output from the command. This can be used in conjunction with '#' (repeat mode). For example, in a loop of ten iterations, this command increments the `Counter` key, then lists the current `NamedCache` values. The output from the `inc` command is suppressed:

```
#10 @inc Counter; list
```

&<functionName> [paramValue]*

Uses reflection to call the specified function on the current `NamedCache` instance. This is intended primarily for internal testing use.

&<functionName> [paramValue]*

A

AbstractProcessor interface, 5-6
aggregating cache data, 4-12
aggregating data, 4-12

B

bulkput command, A-1
bye command, A-1

C

cache
 loading data, 4-1
 populating, 4-1
cache server, setting up, 1-2
cache server, starting, 8-1
cache types, 7-1
cache types, described, 7-2
cache, creating with JDeveloper, 7-2
cache-config.xml file, 6-1, 7-2, 7-16
CacheFactory class, 2-19, 2-23
CacheLoader interface, 7-11
cache-mapping element, 7-2, 7-4
cache-name element, 7-16
CacheStore interface, 7-11
cachestore-scheme element, 7-11
caching complex objects, 3-2
caching-scheme-mapping element, 7-2, 7-4
caching-schemes element, 7-2
ChainedExtractor interface, 4-14
clear command, A-1
client application command
 bulkput, A-1
 bye, A-1
 clear, A-1
 destroy, A-1
 get, A-1
 hash, A-2
 help, A-2
 inc, A-2
 kill, A-2
 list, A-2
 listen, A-2
 lock, A-3

log, A-3
map, A-3
maps, A-3
memory, A-3
put, A-3
release, A-3
remove, A-3
scan, A-4
services, A-4
size, A-4
unlock, A-4
waitkey, A-4
who, A-4
whoami, A-4
worker, A-4
clustering, troubleshooting, 1-12
Coherence
 configuring, 1-1
 installing, 1-1
 restricting to your own host, 1-12
 testing the installation, 1-2
Coherence JPA libraries, 6-13
coherence shell, setting up, 1-2
Coherence*Web, 8-1
coherence-cache-config.xml file, 1-9
coherence-pof-config.xml file, 3-15
coherence-web-spi.jar file, 8-5
complex objects
 caching, 3-2
 creating, 3-2
configuring Coherence, 1-1
configuring JDeveloper, 2-1

D

data grid, accessing from Java, 2-18
destroy command, A-1
distributed-scheme element, 7-4

E

EclipseLink, 6-1
EclipseLink Libraries, 6-13
EntryProcessor interface, 5-1, 5-5

F

Filter interface, 4-13
filter package, 4-13, 7-15

G

get command, A-1

H

hash command, A-2
help command, A-2

I

inc command, A-2
init-param element, 3-16
installing JDeveloper, 1-1
InvocableMap interface, 5-5
InvocableMap.EntryAggregator interface, 4-17

J

Java Persistence API, 6-1
JAVA_HOME environment variable, 2-1
javax.persistence.* libraries, 6-13
JDBC libraries, 6-13
JDeveloper
 configuring, 2-1
 installing, 1-1
JPA, 6-1
jpa-cache-config.xml file, 6-12

K

kill command, A-2

L

list command, A-2
listen command, A-2
listeners, 5-2
loading data into the cache, 4-1
lock command, A-3
log command, A-3

M

map command, A-3
MapEvent class, 5-1
MapListener interface, 5-1
maps command, A-3
memory command, A-3

N

NamedCache interface, 1-9, 2-18, 3-1
network addresses, 1-2

O

Object Relational Mapping (ORM), 6-1
object-relational mapping, 6-1
ObservableMap interface, 5-1
Oracle Database 10g Express Edition (OracleXE), 6-1

P

persistence, 6-1
persistence unit, 6-5
persistence.xml file, 6-11, 6-16
PofReader interface, 3-1
PofWriter interface, 3-1
populating the cache, 4-1
PortableObject interface, 3-1, 4-1, 5-6
put command, A-3

Q

querying cache data, 4-12
querying data, 4-12
QueryMap interface, 4-12, 4-15

R

readExternal method, 3-1
read-write-backing-map scheme, 7-12
ReflectionExtractor interface, 4-14
release command, A-3
remove command, A-3

S

scan command, A-4
services command, A-4
size command, A-4

T

tangosol-coherence.xml file, 7-6
troubleshooting clustering, 1-12
types of read and write caching, 7-12

U

unlock command, A-4
user-type element, 3-15

V

ValueExtractor interface, 4-14

W

waitkey command, A-4
WebLogic Server, 8-1
Weblogic Server, configuration, 8-2
who command, A-4
whoami command, A-4
worker command, A-4
writeExternal method, 3-1

X

XML document, adding to a project, 7-3

