# Contents

**Preface**............................................................................................................................................................ xiii
Audience................................................................................................................................................................ xiii
Documentation Accessibility ......................................................................................................................... xiii
Related Documents ........................................................................................................................................ xiv
Conventions ...................................................................................................................................................... xiv

## 1 Installing and Configuring Coherence

### Installing Coherence.......................................................................................................................... 1-1

### Testing a Coherence Installation......................................................................................................... 1-2

- To Test a Coherence Installation—Main Steps ....................................................................................... 1-2
- Configure and Run the Sample Cache Server Application................................................................. 1-3
- Configure and Run the Sample Cache Client Application ............................................................... 1-6
- Exercise the Sample Cache Client Application.................................................................................... 1-7
- Troubleshooting Cache Server Clustering ............................................................................................ 1-13
- Restricting Coherence to Your Own Host .............................................................................................. 1-13
- Advanced Steps to Restrict Coherence to Your Own Host ................................................................. 1-14

## 2 Installing and Configuring Eclipse and OEPE with Coherence

### Installing Eclipse and OEPE .............................................................................................................. 2-1

### Configuring Eclipse and OEPE for Coherence .................................................................................. 2-1

### Creating a New Project in the Eclipse IDE .......................................................................................... 2-3

### Launching a Cache Server in the Eclipse IDE ..................................................................................... 2-7

### Learning Eclipse IDE Basics ............................................................................................................. 2-10

- Creating a Java Class ........................................................................................................................... 2-11
- Creating a Runtime Configuration ....................................................................................................... 2-12
- Stopping Cache Servers ....................................................................................................................... 2-13

## 3 Accessing the Data Grid from Java

### Introduction .............................................................................................................................................. 3-1

### Creating Your First Coherence-Based Java Program ......................................................................... 3-2

- Create a Program to Put Values in the Cache ....................................................................................... 3-2
- Create a Program to Get Values from the Cache ................................................................................ 3-4

### Creating Your First Coherence-Based Java Application ..................................................................... 3-9

- Create the Console Application ........................................................................................................... 3-10
- Run the Console Application .............................................................................................................. 3-11
4 Working with Complex Objects

Introduction ......................................................................................................................... 4-1
Creating and Caching Complex Objects .............................................................................. 4-2
Create the Data Objects ....................................................................................................... 4-2
Create the Complex Object .................................................................................................. 4-13
Create the Driver Class ......................................................................................................... 4-17
Create the POF and Cache Configuration Files .................................................................. 4-18
Run the Sample Project ........................................................................................................ 4-20

5 Loading Data Into a Cache

Introduction ............................................................................................................................ 5-1
Populating a Cache with Domain Objects .............................................................................. 5-1
Create a Class with the Key for the Domain Objects ............................................................. 5-2
Edit the POF Configuration File .......................................................................................... 5-5
Create the Data Generator .................................................................................................... 5-6
Create a Console Application to Load the Cache ................................................................. 5-10
Run the Cache Loading Example .......................................................................................... 5-12
Querying and Aggregating Data in the Cache ...................................................................... 5-15
Create the Class to Query Cache Data .................................................................................. 5-17
Run the Query Example ........................................................................................................ 5-20
Edit the Query Example to Perform Aggregations ............................................................... 5-22
Run the Query and Aggregation Example ............................................................................ 5-25

6 Simplifying Cache Calls and Aggregations

Introduction ............................................................................................................................ 6-1
Simplifying the Query Example ............................................................................................. 6-2
Rerunning the Query Example .............................................................................................. 6-6

7 Listening for Changes and Modifying Data

Introduction ............................................................................................................................ 7-1
Creating a Cache Listener ...................................................................................................... 7-2
Create a Class to Listen for Changes in the Cache ............................................................... 7-2
Run the Cache Listener Example .......................................................................................... 7-4
Responding to Changes in the Cache .................................................................................... 7-5
Create a Class to Update Entries in the Cache ................................................................. 7-5
Edit the POF Configuration File .......................................................................................... 7-6
Run the Cache Update Example ............................................................................................ 7-9

8 Using JPA with Coherence

Introduction ............................................................................................................................ 8-1
Mapping Relational Data to Java Objects with JPA .............................................................. 8-1
Unlock the OracleXE Database ............................................................................................. 8-2
Download the JPA Libraries .................................................................................................. 8-3
Configure the Project for JPA .............................................................................................. 8-3
Create the JPA Persistence Unit and Entity Beans ............................................................... 8-9
9 Interacting with the Cache and the Database

Introduction .............................................................................................................. 9-1
Creating a Cache Application ..................................................................................... 9-2
  Create an Application that Constructs a Cache ......................................................... 9-2
  Create a Cache Configuration File ............................................................................. 9-4
  Configure the Project Properties ................................................................................ 9-5
  Create the Cache Server Start-Up Configuration ....................................................... 9-5
  Run the Cache Creation Application .......................................................................... 9-6

Creating a Database Cache ......................................................................................... 9-8
  Create an Oracle Database Cache .............................................................................. 9-8
  Create a Class to Define a Custom Cache Store ....................................................... 9-9
  Modify the Cache Configuration File .......................................................................... 9-12
  Create a Class to Construct the Database Cache ...................................................... 9-14
  Run the Database Cache Application ......................................................................... 9-17

10 Working with Security

Introduction .................................................................................................................. 10-1
Enabling Token-Based Security ................................................................................... 10-1
  Use a Security Helper File ......................................................................................... 10-2
  Create an Identity Transformer .................................................................................. 10-5
  Create an Identity Asserter ......................................................................................... 10-7
  Create the Password File .......................................................................................... 10-9
  Enable the Identity Transformer and Asserter .......................................................... 10-10
  Create a Cache Configuration File for the Extend Client .......................................... 10-11
  Create a Cache Configuration File for the Extend Proxy ......................................... 10-12
  Create a Start-Up Configuration for a Cache Server ................................................. 10-14
  Create a Start-Up Configuration for a Cache Server with a Proxy Service ............... 10-15
  Run the Password Example ....................................................................................... 10-16

Including Role-Based Access Control to the Cluster .................................................. 10-18
  Define Which User Roles Are Entitled to Access Cache Methods ............................. 10-19
  Apply the Entitlements to the Cache Service ............................................................ 10-28
  Create the Access Control Example Program ........................................................... 10-30
  Edit the Cluster-Side Cache Configuration File ......................................................... 10-32
  Run the Access Control Example .............................................................................. 10-33

Including Role-Based Access Control to an Invocable Object ..................................... 10-37
  Create an Invocable Object ....................................................................................... 10-37
  Create an Entitled Invocation Service ....................................................................... 10-39
  Create the Access Invocation Service Example Program ......................................... 10-41
  Edit the Cluster-Side Cache Configuration File ......................................................... 10-43
11 Caching Sessions with Coherence and WebLogic Server

Introduction............................................................................................................................................ 11-1

Caching Session Information for Web Application Instances .................................................... 11-2
  Ensure That You are Using Coherence 3.7 with WebLogic Server 10.3.4......................... 11-2
  Start a Cache Server ........................................................................................................................ 11-2
  Configure and Start the WebLogic Server ............................................................................. 11-3
  Create a Machine....................................................................................................................... 11-3
  Create the WebLogic Servers .................................................................................................... 11-5
  Create a Coherence Cluster ......................................................................................................... 11-7
  Deploy the Shared Library Files ................................................................................................. 11-10
  Create the Counter Web Application ....................................................................................... 11-14
  Deploy the Application ............................................................................................................... 11-16
  Start the Node Manager and the WebLogic Servers ............................................................... 11-17
  Verify the Example...................................................................................................................... 11-18

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  query.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 .................................................................................... 1-8
1–6  Exercising Coherence Commands ............................................................................................. 1-11
2–1  Cache Server Output in the Eclipse Console Window ................................................................. 2-9
3–1  Creating a NamedCache Object: Inserting and Verifying Values .............................................. 3-3
3–2  Output of MyFirstSample Program ............................................................................................ 3-3
3–3  Getting a Value from the Cache ................................................................................................ 3-4
3–4  Output of the MyFirstSampleReader Program ......................................................................... 3-5
3–5  Output of MyFirstSampleReader Program with a Running Cache Server ............................... 3-6
3–6  Output of the MyFirstSample Class with Cache Storage Disabled ........................................... 3-8
3–7  A Coherence-Based Java Application ......................................................................................... 3-10
3–8  Output for the QueryPlus Cache Client ...................................................................................... 3-11
3–9  Output of the YourFirstCoherenceApplication Class .............................................................. 3-12
3–10 Output of the YourFirstCoherenceApplication Class with a New Key Value ............................ 3-12
4–1  Implementation of an Address Class .......................................................................................... 4-7
4–2  Implementation of a PhoneNumber Class ................................................................................... 4-10
4–3  Sample Contact Class .............................................................................................................. 4-14
4–4  Sample ContactDriver Class .................................................................................................... 4-17
4–5  POF Configuration File ............................................................................................................. 4-18
4–6  Cache Configuration File ........................................................................................................... 4-19
4–7  Output from the Contacts Cache Server ..................................................................................... 4-21
4–8  Output of the Contacts Example in the Eclipse IDE ................................................................. 4-23
4–9  Contacts Cache Server Displaying the Arrival and Departure of the Contacts Client ................ 4-24
5–1  Simple Contact ID Class .............................................................................................................. 5-3
5–2  POF Configuration File with the ContactId Entry ................................................................... 5-5
5–3  Sample Data Generation Class .................................................................................................. 5-6
5–4  Contents of the contacts.csv File .............................................................................................. 5-10
5–5  Sample Cache Loading Program ............................................................................................... 5-10
5–6  Output from the Sample Cache Loading Program ................................................................... 5-14
5–7  Sample QueryExample Class .................................................................................................... 5-18
5–8  Results of the QueryExample Program ..................................................................................... 5-20
5–9  Methods to Aggregate Over Keys or by Specifying Filters ....................................................... 5-22
5–10 QueryExample with Aggregation ............................................................................................. 5-23
5–11 Output from the Aggregators .................................................................................................... 5-25
6–1  Edited QueryExample File .......................................................................................................... 6-3
6–2  Output of the MA Residents Filter ............................................................................................ 6-6
6–3  Output of the MA Residents, Work Elsewhere Filter .............................................................. 6-7
6–4  Output of the City Begins with S Filter ..................................................................................... 6-7
6–5  Output of the Age Greater than 42 Filter ................................................................................... 6-7
6–6  Output of the State and Age Aggregators ................................................................................... 6-8
7–1  Listener Methods on a NamedCache .......................................................................................... 7-1
7–2  Code Pattern for Registering an Event ...................................................................................... 7-2
7–3  Sample Listener Class .............................................................................................................. 7-2
7–4  Listener Program Waiting for Events ......................................................................................... 7-4
7–5  Sample Program to Update an Object in the Cache ................................................................. 7-7
7–6  Output from the ObserverExample and ProcessorExample Classes ....................................... 7-9
8–1  Generated persistence.xml File ................................................................................................ 8-16
8–2  Cache Configuration for JPA ..................................................................................................... 8-16
8–3  Sample Employee Class File .................................................................................................... 8-19
8–4  Output from the RunEmployeeExample Program ..................................................................... 8-20
8–5  Cache Server Response to Logging In to the Database ............................................................ 8-21
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9–1 Implementation of a Coherence Cache</td>
<td>9-3</td>
</tr>
<tr>
<td>9–2 Cache Configuration File</td>
<td>9-4</td>
</tr>
<tr>
<td>9–3 Output of the Coherence Cache Application</td>
<td>9-7</td>
</tr>
<tr>
<td>9–4 SQL Script for Creating a Database Table</td>
<td>9-8</td>
</tr>
<tr>
<td>9–5 Running the SQL Script for Creating a Database Table</td>
<td>9-8</td>
</tr>
<tr>
<td>9–6 Database Cache Store Implementation</td>
<td>9-9</td>
</tr>
<tr>
<td>9–7 Database Cache Configuration File</td>
<td>9-12</td>
</tr>
<tr>
<td>9–8 Implementation for the Database Cache Class File</td>
<td>9-14</td>
</tr>
<tr>
<td>9–9 Output of the DatabaseCache Program</td>
<td>9-17</td>
</tr>
<tr>
<td>9–10 Cache Server Response to the DatabaseCache Program</td>
<td>9-19</td>
</tr>
<tr>
<td>9–11 Output from the SELECT Statement</td>
<td>9-19</td>
</tr>
<tr>
<td>10–1 A Security Helper File</td>
<td>10-3</td>
</tr>
<tr>
<td>10–2 Sample Identity Transformer Implementation</td>
<td>10-6</td>
</tr>
<tr>
<td>10–3 Sample Identity Asserter Implementation</td>
<td>10-7</td>
</tr>
<tr>
<td>10–4 Sample Implementation to Run the Password Example</td>
<td>10-9</td>
</tr>
<tr>
<td>10–5 Specifying an Identity Transformer and an Asserter</td>
<td>10-10</td>
</tr>
<tr>
<td>10–6 Sample Extend Client Cache Configuration File</td>
<td>10-11</td>
</tr>
<tr>
<td>10–7 Sample Cache Configuration File for the Proxy Server</td>
<td>10-13</td>
</tr>
<tr>
<td>10–8 Password Example Output in the Eclipse Console</td>
<td>10-17</td>
</tr>
<tr>
<td>10–9 Response from the Cache Server Running the Proxy Service Shell</td>
<td>10-18</td>
</tr>
<tr>
<td>10–10 Entitled Named Cache</td>
<td>10-20</td>
</tr>
<tr>
<td>10–11 Entitled Cache Service</td>
<td>10-28</td>
</tr>
<tr>
<td>10–12 Sample Program to Run the Access Control Example</td>
<td>10-30</td>
</tr>
<tr>
<td>10–13 Cache Service Proxy Configuration for a Cluster-Side Cache Configuration</td>
<td>10-32</td>
</tr>
<tr>
<td>10–14 Access Control Example Output in the Eclipse Console</td>
<td>10-34</td>
</tr>
<tr>
<td>10–15 Output for the Cache Server Running the Proxy Service</td>
<td>10-35</td>
</tr>
<tr>
<td>10–16 A Sample Invocable Object</td>
<td>10-38</td>
</tr>
<tr>
<td>10–17 A Sample Entitled Invocation Service</td>
<td>10-40</td>
</tr>
<tr>
<td>10–18 Sample Program to Run the Access Invocation Service Example</td>
<td>10-41</td>
</tr>
<tr>
<td>10–19 Invocation Service Proxy Configuration for a Cluster-Side Cache</td>
<td>10-43</td>
</tr>
<tr>
<td>10–20 POF Configuration File with ExampleInvocable User Type</td>
<td>10-43</td>
</tr>
<tr>
<td>10–21 Client Program Response in the Eclipse Console</td>
<td>10-46</td>
</tr>
<tr>
<td>10–22 Proxy Service Response in the Eclipse Console</td>
<td>10-47</td>
</tr>
<tr>
<td>11–1 Script to Start the Cache Server</td>
<td>11-2</td>
</tr>
<tr>
<td>11–2 Sample weblogic.xml File</td>
<td>11-15</td>
</tr>
<tr>
<td>11–3 Sample manifest.mf File</td>
<td>11-16</td>
</tr>
</tbody>
</table>
List of Tables

1–1 Network Addresses and Ports Used by Coherence .............................................................. 1-2
3–1 Methods in the NamedCache Interface ............................................................................. 3-1
3–2 Methods in the CacheFactory Class ................................................................................ 3-2
6–1 ReflectionExtractors and Their Equivalent createExtractor Statements ....................... 6-2
6–2 *Filter Statements and Their Equivalent createFilter Statements in Queries ............... 6-2
6–3 Filter Statements and Their Equivalent createFilter Statements in Aggregations .......... 6-3
9–1 Descriptions of Cache Types ............................................................................................ 9-2
9–2 Types of Read/Write Caching Supported by Coherence .................................................. 9-13
## List of Figures

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Workspace Launcher Dialog Box</td>
<td>2-2</td>
</tr>
<tr>
<td>2-2</td>
<td>The Java EE Perspective in the Menu Bar</td>
<td>2-2</td>
</tr>
<tr>
<td>2-3</td>
<td>New Application Client Project Dialog Box</td>
<td>2-3</td>
</tr>
<tr>
<td>2-4</td>
<td>Selecting Oracle Coherence in the Project Facets Dialog Box</td>
<td>2-4</td>
</tr>
<tr>
<td>2-5</td>
<td>Specifying a Configuration Name in the Save Preset Dialog Box</td>
<td>2-4</td>
</tr>
<tr>
<td>2-6</td>
<td>Creating a Coherence User Library</td>
<td>2-5</td>
</tr>
<tr>
<td>2-7</td>
<td>Naming the Coherence Library</td>
<td>2-5</td>
</tr>
<tr>
<td>2-8</td>
<td>Adding coherence.jar File to the Coherence User Library</td>
<td>2-6</td>
</tr>
<tr>
<td>2-9</td>
<td>The coherence.jar File Defined in the Coherence User Library</td>
<td>2-7</td>
</tr>
<tr>
<td>2-10</td>
<td>Main Tab in the Run Configurations Dialog Box</td>
<td>2-7</td>
</tr>
<tr>
<td>2-11</td>
<td>Coherence Tab of the Run Configurations Dialog Box</td>
<td>2-8</td>
</tr>
<tr>
<td>2-12</td>
<td>Common Tab of the Run Configurations Dialog Box</td>
<td>2-9</td>
</tr>
<tr>
<td>2-13</td>
<td>Defining a New Java Class</td>
<td>2-11</td>
</tr>
<tr>
<td>2-14</td>
<td>Defining a Launch Configuration for a Runnable File</td>
<td>2-12</td>
</tr>
<tr>
<td>2-15</td>
<td>Setting Runtime Arguments for the Launch Configuration</td>
<td>2-13</td>
</tr>
<tr>
<td>2-16</td>
<td>Java Process in the Windows Task Manager</td>
<td>2-14</td>
</tr>
<tr>
<td>3-1</td>
<td>Main Tab for the Query Client Configuration</td>
<td>3-11</td>
</tr>
<tr>
<td>4-1</td>
<td>Generate Getters and Setters Dialog Box</td>
<td>4-4</td>
</tr>
<tr>
<td>4-2</td>
<td>Generate Constructors using Fields Dialog Box</td>
<td>4-5</td>
</tr>
<tr>
<td>4-3</td>
<td>Coherence Tab for the Contacts Cache Server Executable</td>
<td>4-21</td>
</tr>
<tr>
<td>5-1</td>
<td>Adding Folders to the Project Build Path</td>
<td>5-2</td>
</tr>
<tr>
<td>5-2</td>
<td>Contents of the Order and Export Tab for the Loading Project</td>
<td>5-3</td>
</tr>
<tr>
<td>5-3</td>
<td>Classpath for the DataGenerator Program</td>
<td>5-13</td>
</tr>
<tr>
<td>5-4</td>
<td>Classpath for the LoaderExample Program</td>
<td>5-14</td>
</tr>
<tr>
<td>8-1</td>
<td>Connecting to the Database</td>
<td>8-2</td>
</tr>
<tr>
<td>8-2</td>
<td>Unlocking the Database Account</td>
<td>8-2</td>
</tr>
<tr>
<td>8-3</td>
<td>EclipseLink SDK in the Eclipse Marketplace</td>
<td>8-3</td>
</tr>
<tr>
<td>8-4</td>
<td>Selecting the JPA Perspective in the Open Perspective Dialog Box</td>
<td>8-4</td>
</tr>
<tr>
<td>8-5</td>
<td>Project Facets for a JPA Project</td>
<td>8-4</td>
</tr>
<tr>
<td>8-6</td>
<td>Contents of the New JPA Project Dialog Box</td>
<td>8-5</td>
</tr>
<tr>
<td>8-7</td>
<td>Downloading the EclipseLink JPA Library</td>
<td>8-6</td>
</tr>
<tr>
<td>8-8</td>
<td>EclipseLink Libraries Added to the JPA Facet Page</td>
<td>8-7</td>
</tr>
<tr>
<td>8-9</td>
<td>Connection Profile Page</td>
<td>8-8</td>
</tr>
<tr>
<td>8-10</td>
<td>The New Connection Profile Dialog Box and the Success Message</td>
<td>8-9</td>
</tr>
<tr>
<td>8-11</td>
<td>Selecting the Database Tables</td>
<td>8-10</td>
</tr>
<tr>
<td>8-12</td>
<td>Associations for the EMPLOYEES Table</td>
<td>8-11</td>
</tr>
<tr>
<td>8-13</td>
<td>Customize Default Entity Generation Page</td>
<td>8-12</td>
</tr>
<tr>
<td>8-14</td>
<td>Customize Individual Entities Page</td>
<td>8-13</td>
</tr>
<tr>
<td>8-15</td>
<td>General Tab in the persistence.xml Editor</td>
<td>8-14</td>
</tr>
<tr>
<td>8-16</td>
<td>Connection Properties Tab in the persistence.xml Editor</td>
<td>8-15</td>
</tr>
<tr>
<td>8-17</td>
<td>Properties Tab in the persistence.xml Editor</td>
<td>8-16</td>
</tr>
<tr>
<td>8-18</td>
<td>Main Tab for the JPA Cache Server</td>
<td>8-18</td>
</tr>
<tr>
<td>8-19</td>
<td>Classpath Tab for the JPA Cache Server Executable</td>
<td>8-18</td>
</tr>
<tr>
<td>8-20</td>
<td>Classpath Tab for the RunEmployeeExample Program</td>
<td>8-20</td>
</tr>
<tr>
<td>9-1</td>
<td>Classpath for the CoherenceCacheProgram</td>
<td>9-5</td>
</tr>
<tr>
<td>9-2</td>
<td>Order and Export Tab for Libraries for the Java Build Path</td>
<td>9-6</td>
</tr>
<tr>
<td>9-3</td>
<td>Classpath for the Interact Project Cache Server</td>
<td>9-6</td>
</tr>
<tr>
<td>9-4</td>
<td>Classpath for the DatabaseCacheProgram</td>
<td>9-17</td>
</tr>
<tr>
<td>10-1</td>
<td>Class Path for the Security Cache Server</td>
<td>10-14</td>
</tr>
<tr>
<td>10-2</td>
<td>Arguments Tab for the Security Proxy Server</td>
<td>10-15</td>
</tr>
<tr>
<td>10-3</td>
<td>Class Path for the Proxy Server</td>
<td>10-16</td>
</tr>
<tr>
<td>10-4</td>
<td>Classpath Tab for the PasswordExample Program</td>
<td>10-17</td>
</tr>
<tr>
<td>10-5</td>
<td>Classpath Tab for the AccessControlExample Program</td>
<td>10-33</td>
</tr>
<tr>
<td>10-6</td>
<td>Class Path for the Cache Server and the Server Running the Proxy Service</td>
<td>10-44</td>
</tr>
</tbody>
</table>
Oracle Coherence (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, Coherence enables continuous data availability and transactional integrity, even in the event of a server failure.

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 JavaBeans event model to receive data change notifications.

Audience

This tutorial is intended for software developers, architects, and administrators. It describes how to develop applications for the Oracle Coherence data grid.

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.
Access to Oracle Support
Oracle customers have access to electronic support through My Oracle Support. For information, visit [http://www.oracle.com/support/contact.html](http://www.oracle.com/support/contact.html) or visit [http://www.oracle.com/accessibility/support.html](http://www.oracle.com/accessibility/support.html) if you are hearing impaired.

Related Documents

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

- Oracle Coherence Getting Started Guide
- Oracle Coherence Developer’s Guide
- Oracle Coherence Client Guide
- Oracle Coherence Tutorial for Oracle Coherence
- Oracle Coherence Integration Guide
- Oracle Coherence Management Guide
- Oracle Coherence Administrator’s Guide
- Oracle Coherence Security Guide

Conventions

The following text conventions are used in this tutorial:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
This chapter describes how to install and set up your environment for running Oracle Coherence Release 3.7 (Coherence).

Note: This tutorial uses the Eclipse Helios 3.6.2 release with Oracle Enterprise Pack for Eclipse (OEPE) 11gR1 (11.1.1.7). For information about configuring Eclipse and OEPE for Coherence-based projects, see Chapter 2, "Installing and Configuring Eclipse and OEPE with Coherence".

You must 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. You must have a working installation of Java SE (JDK) version 1.6 or higher. You must also be running one of the following Microsoft Windows operating systems: XP, Vista, 2000, 2003, or 2008.

This chapter contains the following sections:

- Installing Coherence
- Testing a Coherence Installation

Installing Coherence

To download and install Coherence:

1. Download Coherence to your desktop.
   
   Coherence (Java edition) ships as a single zip file, typically called coherence-\<version\>.zip. You can download Coherence from the following URL:
   

2. Extract the contents of the zip file to a folder named C:\oracle\product.
   
   The zip file contains the coherence folder with these subfolders:
   
   - bin, which contains command scripts
   - doc, which contains the product documentation
   - lib, which contains the required library files
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.

To complete this exercise, you must have Oracle Coherence (Java Edition) Release 3.7 installed. See "Installing Coherence" on page 1-1 for more information.

Coherence uses a variety of network addresses and ports to enable communication between clustered processes. If these addresses and 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

<table>
<thead>
<tr>
<th>Address / Port / Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>224._version / version / Multicast</td>
<td>Cluster member discovery and broadcast. The variable version represents the release version of Coherence. For example, the multicast address for the 3.7.0.0 release is 224.3.7.0. The port number also reflects the release version. For example, for the 3.7.0.0 release, the port number is 37000. Designing the address in this way ensures that different versions of Coherence do not cluster with each other by default.</td>
</tr>
<tr>
<td>localhost / 8088+ / Unicast</td>
<td>Interprocess communication between cluster members. (localhost is the local IP address and not the loopback address.)</td>
</tr>
</tbody>
</table>

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 cache client 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 either on a single host or on several hosts, you can determine whether Coherence is operating correctly locally or across a network.

When an application uses Coherence as shipped, objects placed into Coherence caches are typically stored and managed in-process within the application. However, to increase object availability, Coherence can manage objects in memory but out of the application process. This enables 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 the application state in a cluster outside the application process. It is similar to a database server, but without the requirement for storage.

To Test a Coherence Installation—Main Steps

Follow these steps to test a Coherence Installation. The steps are described in detail in the following sections.

1. Configure and Run the Sample Cache Server Application
2. Configure and Run the Sample Cache Client Application
3. Exercise the Sample Cache Client Application

4. Troubleshooting Cache Server Clustering

Configure and Run the Sample Cache Server Application

To set up and run the sample cache server application:

1. Open a terminal window and verify that the PATH environment variable is set to include the Java JDK (for example, \oracle\product\jdk160_14_R27.6.5-32\bin). If the PATH environment variable does not include the JDK \bin folder, then set it as follows:
   a. Set the JAVA_HOME environment variable to the base of the JDK installation.
      ```
      set JAVA_HOME=\oracle\product\jdk160_14_R27.6.5-32
      ```
   b. Include JAVA_HOME\bin in the PATH environment variable.
      ```
      set PATH=%JAVA_HOME%\bin;%PATH%
      ```

2. Navigate to the folder where Coherence is installed. Edit the cache-server.cmd file and set the COHERENCE_HOME variable to point to the Coherence installation folder.
   ```
   cd C:\oracle\product\coherence\bin
   ```
   In the cache-server.cmd file, set the COHERENCE_HOME environment variable:
   ```
   COHERENCE_HOME=C:\oracle\product\coherence
   ```
   Example 1–1 illustrates the cache-server.cmd file with the edited value of the COHERENCE_HOME environment variable.

   **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
   ```
Testing a Coherence Installation

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

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 the cache server determines that there are no clusters to join, it starts one. On start-up, 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.
- Information about how configuration files are loaded. The default is to load from the coherence.jar file:

  Loaded operational configuration from resource
  "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"

- The Coherence release number: Oracle Coherence 3.7.0.0...

- The Coherence edition: Grid Edition: Development mode

- The multicast address. This address changes with each Coherence version. Note the 3.7.0 in the address for Coherence Release 3.7:

  Group{Address=224.3.7.0, Port=37000, TTL=4}

- The Member ID indicates the number of members in your cluster. For the purpose 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_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Server VM (build 14.0-b16, mixed mode)

2011-03-14 16:16:30.998/1.531 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"

2011-03-14 16:16:31.342/1.875 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"

2011-03-14 16:16:31.357/1.890 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/tangosol-coherence-override.xml" is not specified

2011-03-14 16:16:31.373/1.906 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-14 16:16:32.638/3.171 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence-cache-config.xml"

2011-03-14 16:16:33.295/3.828 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider


2011-03-14 16:16:36.873/7.406 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB

Group(Address=224.3.7.0, Port=37000, TTL=4)

MasterMemberSet
{
  ThisMember=Member(Id=1, Timestamp=2011-03-14 16:16:33.295, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,process:1920, Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2011-03-14 16:16:33.295, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,process:1920, Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2)
    Member(Id=1, Timestamp=2011-03-14 16:16:33.295, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,process:1920, Role=CoherenceServer)
}

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2011-03-14 16:16:36.935/7.468 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=1): Service Management joined the cluster with senior service member 1

2011-03-14 16:16:37.217/7.750 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache, member=1): Service DistributedCache joined the cluster with senior service member 1

2011-03-14 16:16:37.295/7.828 Oracle Coherence GE 3.7.0.0 <D5> (thread=ReplicatedCache, member=1): Service ReplicatedCache joined the cluster with senior service member 1

2011-03-14 16:16:37.310/7.843 Oracle Coherence GE 3.7.0.0 <D5> (thread=OptimisticCache, member=1): Service OptimisticCache joined the cluster with senior service member 1

2011-03-14 16:16:37.310/7.843 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:InvocationService, member=1): Service InvocationService joined the cluster with senior service member 1

2011-03-14 16:16:37.326/7.859 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=1): Services
{
  ClusterService(Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.7, OldestMemberId=1)
  InvocationService(Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1)
  PartitionedCache(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}  

Started DefaultCacheServer...

Note: By default, Coherence is configured to use multicast to join a cluster and to distribute cluster events. Multicast can also be used to distribute a message efficiently to multiple nodes in the cluster. You can configure Coherence to disable multicast.

The output of the cache-server.cmd file 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 ThisMember has an Id greater than one, it means that your cache server has joined a pre-existing cluster in your network. For the purposes of these exercises, the cluster should contain only one member. Follow the steps in "Restricting Coherence to Your Own Host" on page 1-13 to restrict Coherence to your own host.

Configure and Run the Sample Cache Client Application

To set up and run the sample cache client application:

1. Open another terminal window to start the cache client.
   
   Verify that the PATH environment variable is set to include the %JAVA_HOME%\bin folder. If the PATH environment variable does not include the %JAVA_HOME%\bin folder, then set the variable as described in "Configure and Run the Sample Cache Server Application" on page 1-3.

2. Navigate to the \oracle\product\coherence\bin folder. Edit the query.cmd file to set the COHERENCE_HOME variable to point to the Coherence installation folder.
   
   (Optional) Set the JLINE_HOME variable to point to the JLine installation folder. JLine is a Java library that simplifies working with console commands. You can download the Jline library from this URL:

   http://jline.sourceforge.net/

   Example 1-3 illustrates the query.cmd file, with COHERENCE_HOME=\oracle\product\coherence. JLINE_HOME is set to c:\oracle\product\coherence\jline-0.9.94.

   **Example 1–3** query.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
3. Execute the `query.cmd` file to start the cache client.

   `query.cmd`

   The output of starting the cache client, illustrated in Example 1–4, displays the basic distributed cache functionality that is built into Coherence. At the end of the output, the `CohQL>` prompt is displayed.

   **Example 1–4  Output from Starting the Coherence Cache Client**
   
   ```
   C:\coherence360\coherence\bin>query.cmd
   ** Starting storage disabled console **
   java version '1.6.0_14'
   Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
   Java HotSpot(TM) Server VM (build 14.0-b16, mixed mode)
   ```
Exercise the Sample Cache Client Application

Exercise the cache client application by entering various commands and examining the output.

1. Execute the following Coherence commands in the cache client:
   - Enter `help` to see the list of commands that are available.
   - Enter `create cache "products"` to create a cache named `products`.

   The cache `products` implements the `com.tangosol.net.NamedCache` interface. A cluster can have many named caches.

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

Example 1–5  Output from Starting a Coherence Cache

```
Coql> create cache "products"
2011-03-14 16:49:09.217/109.469 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2011-03-14 16:49:09.482/109.734 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2011-03-14 16:49:09.482/109.734 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/tangosol-coherence-override.xml" is not specified
2011-03-14 16:49:09.482/109.734 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified
Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
```

```
2011-03-14 16:49:10.498/110.750 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2011-03-14 16:49:11.185/111.437 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8090 using SystemSocketProvider
2011-03-14 16:49:11.732/111.984 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service Cluster with senior member 1
2011-03-14 16:49:11.732/111.984 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service Management with senior member 1
2011-03-14 16:49:11.732/111.984 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service DistributedCache with senior member 1
2011-03-14 16:49:11.732/111.984 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service ReplicatedCache with senior member 1
```
2. Execute the following commands at the `CohQL>` prompt in the cache client. For definitions of these commands, see "Using Coherence Query Language" in Developer’s Guide for Oracle Coherence.

- Create a cache named `products`:
  ```sql
  create cache "products"
  ```

- Insert an entry (key-value pair) into the cache:
  ```sql
  insert into "products" key "television" value "ID-5070"
  ```

- Change the value of the key:
  ```sql
  update "products" set value() = "ID-5080" where key() like 'television'
  ```

- Retrieve the values in the cache:
  ```sql
  select * from "products"
  ```

- Retrieve the value of a key that does not exist. An empty result set will be returned:
  ```sql
  select key(), value() from "products" where key() is 'radio'
  ```
- Delete an existing key in the cache. An empty result set will be returned:
  
  `delete from "products" where key() = "television"`

- Delete the contents of the `products` cache. An empty result set will be returned:
  
  `delete from "products"`

- Destroy the `products` cache:
  
  `drop cache "products"`

- Re-create the `products` cache:
  
  `create cache "products"`

- Insert more entries into the cache:
  
  `insert into "products" key "television" value "ID-5080"
  insert into "products" key "radio" value "ID-5090"
  insert into "products" key "MP3 Player" value "ID-5100"
  insert into "products" key "laptop" value "ID-5110"

- Retrieve the keys and values in the `products` cache:
  
  `select key(), value() from "products"`

- Save a serialized representation of the cache in a file:
  
  `backup cache "products" to "products.bkup"`

- Delete a key from the cache:
  
  `delete from "products" where key() = "television"`

- Retrieve the cache contents again, notice that the deleted key and value will not be present:
  
  `select key(), value() from "products"`

- Delete the contents of the cache:
  
  `delete from "products"`

- Retrieve the contents of the cache. An empty result set will be returned:
  
  `select * from "products"`

- Restore the cache contents from the backup file:
  
  `restore cache "products" from file "products.bkup"`

- Retrieve the cache contents. Note that all of the entries will be restored and returned:
  
  `select key(), value() from "products"`

- Destroy the `products` cache:
  
  `drop cache "products"`

- Exit the command-line tool:
Testing a Coherence Installation

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

Example 1–6  Exercising Coherence Commands

CohQL> create cache "products"

CohQL> insert into "products" key "television" value "ID-5070"

CohQL> update "products" set value() = "ID-5080" where key() like "television"
Results
television: true

CohQL> select * from "products"
Results
ID-5080

CohQL> select key(), value() from "products" where key() is "radio"
Results

CohQL> delete from "products" where key() = "television"
Results

CohQL> delete from "products"
Results

CohQL> drop cache "products"

CohQL> create cache "products"

CohQL> insert into "products" key "television" value "ID-5080"

CohQL> insert into "products" key "radio" value "ID-5090"

CohQL> insert into "products" key "MP3 Player" value "ID-5100"

CohQL> insert into "products" key "laptop" value "ID-5110"

CohQL> select key(), value() from "products"
Results
'television', 'ID-5080'
'radio', 'ID-5090'
'MP3 Player', 'ID-5100'
'laptop', 'ID-5110'

CohQL> backup cache "products" to "products.bkup"
WARNING: The backup command should not be used on active data set, as it makes no provisions that ensure data consistency during the backup. Please see the documentation for more detailed information.

CohQL> delete from "products" where key() = "television"
Results

CohQL> select key(), value() from "products"
Results
'radio', 'ID-5090'
'MP3 Player', 'ID-5100'
'laptop', 'ID-5110'
3. Open another terminal window and set the PATH environment variable to include %JAVA_HOME% and %JAVA_HOME%\bin. Enter the query.cmd command in the new terminal window to start an instance of the cache client.

4. Restart the first client with the query.cmd command. Enter the create cache "products" command. The terminal window displays a message similar to the following that describes where the first client is running. member 1 is the cache server and member 3 is the restarted first client.

5. Enter the create cache "products" command in the new terminal window to connect to the products cache. Try to get and put values in different sessions. Notice that each client can observe changes made by the other client.

6. Terminate one of the query.cmd shells (bye). Note that the other shell displays a message indicating that the member has left the cluster, for example:
Testing a Coherence Installation

Installing and Configuring Coherence

1-13

7. If you terminate each of the cache clients (byye), and then restart them, note that data from the previous session is still available. This is because the data is held in the cache server.

8. If you start another Coherence cache server, and then terminate the initial one that was started (using Ctr1+C or by closing the command window), note that the data is still available.

9. Terminate both the query.cmd shells and cache-server.cmd shell. Restart query.cmd. Create a cache called test using the command create cache test. Try to put a value into the cache, as before. Because the cache client is configured to start in storage-disabled mode, you will receive a response similar to the following:

    com.tangosol.net.RequestPolicyException: No storage-enabled nodes exist for service DistributedCache

10. Start a cache server by running the cache-server.cmd file. Try to insert a value again. This time, the value will be accepted.

11. Terminate all 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 can 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 that while you perform this tutorial, you restrict Coherence to your own host.

Note: To restrict Coherence to a single host, the cache server and cache client executable files used in this tutorial use the tangosol.coherence.clusterport command line property set to 3155.

Restricting Coherence to Your Own Host

The value of the Member ID in the output of the cache-server.cmd cache server command indicates whether you have one or more members in your cluster. For the purpose of this tutorial, the value of Member ID must be equal to 1:

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

If the value of Member ID is greater than 1, it means that multiple clusters are being formed in your subnet.
There are several ways to restrict Coherence to your own host. The easiest way is to use the `tangosol.coherence.clusterport` system property to declare a unique cluster port value in your cache server startup file. For example, add the following Java option to your `cache-server.cmd` file. The value assigned to the system property can be any unique value, such as the last four digits of your telephone number.

```
-Dtangosol.coherence.clusterport=3155
```

**Advanced Steps to Restrict Coherence to Your Own Host**

If you follow the steps in "Restricting Coherence to Your Own Host" on page 1-13 and the `cache-server.cmd` command still fails to return a Member Id value of 1, then there might be additional problems to 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 after each attempt:

- If connected to a Virtual Private Network (VPN), then disconnect from it. By default, most VPN are not configured to permit multicast and some unicast traffic. In this environment, Coherence might not work, if left in the configuration in which it was shipped. Coherence can be configured to run across a VPN, but this requires some advanced settings.

- If you run a firewall, configure it to enable the specified addresses and ports.

- If you still experience problems, disconnect from all networks. This includes wireless and wired networks.

- If all the preceding options fail, set up Coherence to run on a single host.
This chapter describes how to set up the Helios release of the Eclipse IDE for Java EE Developers (Eclipse) and Oracle Enterprise Pack for Eclipse (OEPE) 11gR1 (11.1.1.7) to build and run Coherence-based Java applications.

This chapter contains the following sections:

- Installing Eclipse and OEPE
- Configuring Eclipse and OEPE for Coherence
- Creating a New Project in the Eclipse IDE
- Launching a Cache Server in the Eclipse IDE
- Learning Eclipse IDE Basics

**Installing Eclipse and OEPE**

You can download an "all-in-one" version of OEPE 11gR1 (11.1.1.7) that bundles a preconfigured version of Eclipse Helios and the OEPE plug-ins. Oracle recommends installing the "all-in-one" version.

As an alternative, you can download the OEPE distribution by itself and install it into an existing Eclipse Helios installation. The OEPE installer includes Oracle WebLogic Server, Oracle Coherence, and the Oracle ADF Runtime.

To download either version, select the Downloads tab on the Oracle Enterprise Pack for Eclipse page:


Follow the instructions in the installer. You can also find installation instructions at this URL:


This tutorial assumes that you will be installing Eclipse into an eclipse folder that you have created at the file system root, for example:

C:\eclipse\*

**Configuring Eclipse and OEPE for Coherence**

To start and configure Eclipse for use with Coherence:
1. Open a terminal window and verify that the `PATH` environment variable is set to include the Java JDK, for example: `\oracle\product\jdk160_14_R27.6.5-32\bin`. Note that you must have a working installation of Java SE (JDK) version 1.6 or higher.

If the `PATH` environment variable does not include the Java JDK `\bin` folder, then set the variable as follows:

   a. Set the `JAVA_HOME` environment variable to the base of the JDK installation, for example:
      ```
      set JAVA_HOME=\oracle\product\jdk160_14_R27.6.5-32
      ```

   b. Include `%JAVA_HOME%\bin` in the `PATH` environment variable, for example:
      ```
      set PATH=%JAVA_HOME%\bin;%PATH%
      ```

2. Start Eclipse.

   The `eclipse` executable is `eclipse.exe`. If you extracted Eclipse into a directory called `eclipse`, you will find `eclipse.exe` here:
   ```
   C:\eclipse\eclipse.exe
   ```

3. If Eclipse prompts you to set or select a workspace in the **Workspace Launcher** dialog box, enter `C:\home\oracle\workspace`.

   Figure 2–1 illustrates the **Workspace Launcher** dialog box with the path `C:\home\oracle\workspace` selected.

![Figure 2–1 Workspace Launcher Dialog Box](image)

4. After Eclipse starts, select the Java EE perspective in the menu bar.

   Figure 2–2 illustrates the location of the Java EE perspective button.
Creating a New Project in the Eclipse IDE

To create a new project in the Eclipse IDE:

1. Select File then New then Application Client Project.
2. In the New Application Client Project dialog box, enter a value, Coherence for example, as the Project name.

3. In the Configuration section, click Modify. In the Project Facets dialog box, select the Oracle Coherence check box. Select version 3.7 from the drop-down list if it is not already displayed.

Adding the Oracle Coherence facet automatically adds the coherence.jar to your project class path. It also makes these commonly used configuration files available in the Project Explorer:

- coherence-cache-config.xml, the default cache configuration file.
- pof-config.xml, the configuration file for Portable Object Format serialization.
- tangosol-coherence-override.xml, the override file for operational and runtime settings used by Oracle Coherence.
4. Click **Save As** to enter a name for the configuration in the **Save Preset** dialog box. For example, enter `CoherenceConfig` in the **Name** field and **Includes Coherence Facet** in the **Description** field.

Figure 2–5 illustrates the **Save Preset** dialog box.

5. Click **Next** in the **Application Client Module** page of the **New Application Client Project** dialog box.

6. Click **Next** in the **Java** page of the **New Application Client Project** dialog box to accept the defaults.

7. Deselect the **Create a default Main class** in the **Application Client Module** page of the **New Application Client Project** dialog box. Click **Next**.
8. In the Coherence page, illustrated in Figure 2–6, add the Coherence 3.7 library as a User Library to the project.

*Figure 2–6  Creating a Coherence User Library*

- Click the Manage Library icon, then click New in the Preferences dialog box.
- Enter Coherence37 in the New User Library dialog box, as illustrated in Figure 2–7. Select the System library (added to the boot class path) check box.

*Figure 2–7  Naming the Coherence Library*

- Click OK to close the New User Library dialog box.
- In the Preferences dialog box, click Add JARs to add the coherence.jar file to the library. Navigate to the location of the coherence.jar file in the
Coherence distribution that you downloaded to your file system in "Installing Coherence" on page 1-1.

Figure 2–8 illustrates the JAR Selection dialog box with the coherence.jar file selected.

Figure 2–8  Adding coherence.jar File to the Coherence User Library

e. the Preferences dialog box should look similar to Figure 2–9. Click OK to close the Preferences dialog box.
9. Select the Coherence library in the Coherence page and click Finish. The new project and its associated files appear in the Project Explorer window in the Eclipse IDE.

Launching a Cache Server in the Eclipse IDE

1. Right click the project in the Eclipse IDE. Select Run As then Run Configurations. In the Run Configurations dialog box, select Oracle Coherence then the New launch configuration icon. Enter DefaultCacheServer as the name for the cache server configuration.

2. Under Project, click Browse and select the name of the project from the Project Selection dialog box.

3. Under Main class, select the Include system libraries when searching for a main class checkbox. Click the Search button and enter DefaultCacheServer in the Select Main Type dialog box. Select com.tangosol.net.DefaultCacheServer and click OK. Click Apply. The Main tab should look similar to Figure 2–10.

4. In the Coherence tab, select the General tab. Click the Browse icon to navigate to the cache configuration file. Select local storage to be enabled (cache server). Enter
a unique value, such as 3155 for the Cluster port. Click Apply. The Coherence tab should look similar to Figure 2–11.

**Figure 2–11  Coherence Tab of the Run Configurations Dialog Box**

5. Open the Arguments tab. Enter -showversion in the VM Arguments field. Click Apply.

6. Open the Common tab of the dialog box. Click the Shared file radio button and click Browse to navigate to the project. Click Apply. The Common tab should look similar to Figure 2–12.
7. Click Run to start the cache server. The cache server should start and display output similar to Example 2–1.

Example 2–1 Cache Server Output in the Eclipse Console Window

```
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode)
2011-03-14 17:08:24.053/0.281 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml'
2011-03-14 17:08:24.116/0.344 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml'
2011-03-14 17:08:24.147/0.375 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'file:/C:/home/oracle/workspace/InsertValue/build/classes/tangosol-coherence-override.xml'
2011-03-14 17:08:24.147/0.375 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override '/custom-mbeans.xml' is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-14 17:08:24.397/0.625 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cache-config.xml'
2011-03-14 17:08:24.600/0.828 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider
2011-03-14 17:08:28.116/4.344 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB
```
Group(Address=224.3.7.0, Port=3155, TTL=4)

MasterMemberSet
{
  ThisMember=Member(Id=1, Timestamp=2011-03-14 17:08:24.6, Address=130.35.99.213:8088,
  MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:2412,
  Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2011-03-14 17:08:24.6, Address=130.35.99.213:8088,
  MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:2412,
  Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2011-03-14 17:08:24.6, Address=130.35.99.213:8088, MachineId=49877,
    Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:2412, Role=CoherenceServer)
  )
RecycleMillis=1200000
RecycleSet=MemberSet(Size=0, BitSetCount=0
)
}
TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2011-03-14 17:08:28.163/4.391 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=1): Service Management joined the cluster with senior service member 1
2011-03-14 17:08:28.319/4.547 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache, member=1): Service DistributedCache joined the cluster with senior service member 1
2011-03-14 17:08:28.350/4.578 Oracle Coherence GE 3.7.0.0 <D5> (thread=ReplicatedCache, member=1): Service ReplicatedCache joined the cluster with senior service member 1
2011-03-14 17:08:28.413/4.641 Oracle Coherence GE 3.7.0.0 <D5> (thread=OptimisticCache, member=1): Service OptimisticCache joined the cluster with senior service member 1
2011-03-14 17:08:28.413/4.641 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:InvocationService, member=1): Service InvocationService joined the cluster with senior service member 1
2011-03-14 17:08:28.413/4.641 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=1): Services
  {
    ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.7, OldestMemberId=1}
    InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
    PartitionedCache{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}
  }
Started DefaultCacheServer...

Learning Eclipse IDE Basics

This section describes common tasks that are a part of building projects in the Eclipse IDE.

- Creating a Java Class
- Creating a Runtime Configuration
- Stopping Cache Servers
Creating a Java Class

To create a new Java class in the Eclipse IDE:

1. Right-click the project entry in the Project Explorer window. Select New then Class.
2. In the New Java Class dialog box, enter a package name. In this tutorial, the value will typically be com.oracle.handson.
3. Enter the name of the class in the Name field.
4. Under Which method stubs would you like to create?:
   - Select the public static void main(String[] args) if you want the file to be runnable.
   - Select Constructors from superclass check box if you want stubs of the constructors from the new class's superclass to be added.
   - Inherited abstract methods should be selected by default. This option adds stubs of any abstract methods from superclasses or methods of interfaces that need to be implemented.
5. Click OK to create the Java class.

Figure 2–13 illustrates the New Java Class dialog box.

Figure 2–13 Defining a New Java Class
Creating a Runtime Configuration

To create a runtime configuration for a runnable file:

1. Right click the name of the runnable file in the Project Explorer window. Select Run As then Run Configurations.

2. Click Oracle Coherence, then the New Launch Configuration icon. Select the Java Application node. Click the New Launch Configuration button.

3. Enter a name for the new configuration. Under Project in the Main tab, click the Browse button to navigate to the name of the current project. Click the Search button to navigate to the name of the runnable file for which you are creating the runtime configuration. Click Apply.

This figure illustrates the Main tab of the Run Configurations dialog box.

Figure 2–14  Defining a Launch Configuration for a Runnable File

4. Click the Coherence tab. Under the General tab, you can set many of the more commonly used VM runtime arguments for the configuration. Examples of runtime arguments include the path to the cache configuration descriptor, and the local storage, log level, and cluster port values.

Figure 2–15 illustrates the General tab of the Coherence tab in the Run Configurations dialog box.
Stopping Cache Servers

In the Eclipse IDE, you can stop any running cache server or client by clicking the Terminate and Remove all terminated launches icons in the Eclipse Console.

However, if you look in the Windows Task Manager window, you might notice that the Java process (java.exe) associated with the server or client is still running. To prevent any errors being thrown when the server or client is restarted, you must delete its associated Java process. Select the Java process in the Windows Task Manager window and click End Process.

Figure 2–16 illustrates the Windows Task Manager window with the Java process selected.
Figure 2–16  Java Process in the Windows Task Manager
In this exercise, you develop a simple, Java console-based application to access, update, and remove simple types of information from a Coherence clustered cache. You also are introduced to the Coherence Java APIs. Using the Eclipse IDE, you will perform the following tasks:

- Create a new project
- Create a new named cache (NamedCache object)
- Put information into the cache and then retrieve it
- Retrieve information about the cache

This chapter contains the following sections:

- Introduction
- Creating Your First Coherence-Based Java Program
- Creating Your First Coherence-Based Java Application

### Introduction

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 the `java.util.Map` interface and holds data and resources that are shared among the cluster members. Each `NamedCache` object 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. Table 3–1 describes some of the commonly used methods within the `NamedCache` interface.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void clear()</code></td>
<td>Removes all entries from the <code>NamedCache</code> object.</td>
</tr>
<tr>
<td><code>boolean containsKey(Object key)</code></td>
<td>Returns <code>true</code> if the <code>NamedCache</code> object contains an entry for the key.</td>
</tr>
<tr>
<td><code>boolean containsValue(Object value)</code></td>
<td>Returns <code>true</code> if there is at least one entry with this value in the <code>NamedCache</code> object.</td>
</tr>
<tr>
<td><code>Object get(Object key)</code></td>
<td>Gets the entry from the <code>NamedCache</code> object for that key.</td>
</tr>
<tr>
<td><code>Object put(Object key, Object value)</code></td>
<td>Puts an object in the cache and returns the previous value (if any).</td>
</tr>
</tbody>
</table>
The `com.tangosol.net.CacheFactory` class is typically used to get an instance of a `NamedCache` object. Table 3–2 describes some of the more commonly used methods in the `CacheFactory` class.

### Table 3–2 Methods in the CacheFactory Class

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

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

### Creating Your First Coherence-Based Java Program

This section describes how to create a Java program that enables you to access, update, and remove simple types of information from a Coherence clustered cache.

1. Create a Program to Put Values in the Cache
2. Create a Program to Get Values from the Cache

#### Create a Program to Put Values in the Cache

To create a Coherence Java-based application:

1. Create a new Application Client Project in Eclipse. Name the project `InsertValue`. Ensure that the folder is `C:\home\oracle\workspace\InsertValue`.

   In the Configuration section of the New Application Client Project dialog box, click Modify. In the Project Facets dialog box, select `CoherenceConfig` from the Configuration drop down list.

   See "Creating a New Project in the Eclipse IDE" on page 2-3 for detailed instructions.

2. Create your first Coherence Java program. Name the class `MyFirstSample` and select the `public static void main(String[] args)` check box in the New Java Class dialog box.

   See "Creating a Java Class" on page 2-11 for detailed information.
3. In the Eclipse editor, write the code to create a NamedCache object, enter a value in the cache, and then verify the value that you entered. Example 3–1 illustrates a sample program.

Example 3–1  Creating a NamedCache Object: Inserting and Verifying Values

```java
package com.oracle.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","Gene Smith");

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


5. Run the program in the Eclipse IDE: right-click the MyFirstSample.java class in the editor and select Run As then Run Configuration. Double-click Oracle Coherence to create a Coherence configuration. Enter MyFirstSample in the Name field of the Run Configuration dialog box.

   In the Main tab, enter InsertValue in the Project field and com.oracle.handson.MyFirstSample in the Main class field.

   In the Coherence tab, enter a unique value, such as 3155, in the Cluster port field to ensure that Coherence is limited to your own host. Select Enabled (cache server) under Local storage.

   Note that in the Classpath tab, the coherence.jar file should be present in the Bootstrap Entries list. Click Apply, then Run.

   Messages similar to Example 3–2 are displayed:

Example 3–2  Output of MyFirstSample Program

```
2011-03-14 17:01:25.443/0.297 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from 'jar:/tangosol-coherence.xml'
2011-03-14 17:01:25.490/0.344 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'jar:/tangosol-coherence-override-dev.xml'
2011-03-14 17:01:25.537/0.391 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'file:/C:/home/oracle/workspace/InsertValue/build/classes/tangosol-coherence-override.xml'
2011-03-14 17:01:25.537/0.391 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified
```
Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-14 17:01:25.771/0.625 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar/coherence-cache-config.xml"
2011-03-14 17:01:25.975/0.829 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider
2011-03-14 17:01:29.490/4.344 Oracle Coherence GE 3.7.0.0 <Info> (thread=Cluster, member=n/a): Created a new cluster "cluster:0x96AB" with Member(Id=1, Timestamp=2011-03-14 17:01:25.975, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5000, Role=OracleHandsonMyFirstSample, Edition=Grid Edition, Mode=Development, CpuCount=2, SocketCount=1) UID=0x822363D50000012E552F4A57C2D51F98
2011-03-14 17:01:29.490/4.344 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB
Group(Address=224.3.7.0, Port=3155, TTL=4)

MasterMemberSet
{
    ThisMember=Member(Id=1, Timestamp=2011-03-14 17:01:25.975, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5000, Role=OracleHandsonMyFirstSample)
    OldestMember=Member(Id=1, Timestamp=2011-03-14 17:01:25.975, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5000, Role=OracleHandsonMyFirstSample)
    ActualMemberSet=MemberSet(Size=1, BitSetCount=2
        Member(Id=1, Timestamp=2011-03-14 17:01:25.975, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5000, Role=OracleHandsonMyFirstSample)
    )
RecycleMillis=1200000
RecycleSet=MemberSet(Size=0, BitSetCount=0
)
}

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2011-03-14 17:01:29.521/4.375 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=1): Service Management joined the cluster with senior service member 1
2011-03-14 17:01:29.600/4.454 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache, member=1): Service DistributedCache joined the cluster with senior service member 1

Value in cache is Gene Smith

Create a Program to Get Values from the Cache

To create a Java class that gets the value from your cache, instead of using a put and then a get method:

1. Create another Java class with a main method named MyFirstSampleReader. See “Creating a Java Class” on page 2-11 for detailed information. Example 3–3 illustrates a sample program.

Example 3–3 Getting a Value from the Cache

```java
package com.oracle.handson;
import com.tangosol.net.CacheFactory;
```
import com.tangosol.net.NamedCache;

public class MyFirstSampleReader {
    public MyFirstSampleReader() {
    }

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

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

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

2. Run the MyFirstSampleReader class in the Eclipse IDE: right-click the MyFirstSampleReader.java class in the editor and select Run As then Run Configuration.

Enter MyFirstSampleReader in the Name field of the Run Configuration dialog box. Ensure that InsertValue appears in the Project field in the Main tab and that local storage is enabled in the Coherence tab. Click Apply, then Run.

Example 3–4 illustrates the output from the program. Note that a null value is returned. Although the MyFirstSample program successfully created and populated the NamedCache cache, it only existed within the MyFirstSample process memory. When the MyFirstSample program terminated so did the cache.

Example 3–4 Output of the MyFirstSampleReader Program

2011-03-14 17:06:17.115/0.281 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2011-03-14 17:06:17.162/0.328 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2011-03-14 17:06:17.193/0.359 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "file:/C:/home/oracle/workspace/InsertValue/build/classes/tangosol-coherence-override.xml"
2011-03-14 17:06:17.193/0.359 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-14 17:06:17.568/0.734 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider
2011-03-14 17:06:21.115/4.281 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB
Group(Address=224.3.7.0, Port=3155, TTL=4)

MasterMemberSet
(  
ThisMember=Member(Id=1, Timestamp=2011-03-14 17:06:17.568, Address=130.35.99.213:8088,  
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:216,  
Role=OracleHandsonMyFirstSampleReader)  
OldestMember=Member(Id=1, Timestamp=2011-03-14 17:06:17.568, Address=130.35.99.213:8088,  
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:216,  
Role=OracleHandsonMyFirstSampleReader)  
ActualMemberSet=MemberSet(Size=1, BitSetCount=2  
Member(Id=1, Timestamp=2011-03-14 17:06:17.568, Address=130.35.99.213:8088, MachineId=49877,  
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:216,  
Role=OracleHandsonMyFirstSampleReader)  
)  
RecycleMillis=1200000  
RecycleSet=MemberSet(Size=0, BitSetCount=0  
)  
)

TcpRing{Connections=[], IpMonitor{AddressListSize=0}

2011-03-14 17:06:21.162/4.328 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management,  
member=1): Service Management joined the cluster with senior service member 1  
2011-03-14 17:06:21.350/4.516 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=1): Loaded  
cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.  
jar/coherence-cache-config.xml"  
2011-03-14 17:06:21.412/4.578 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache, member=1):  
Service DistributedCache joined the cluster with senior service member 1

**Value in cache is null**

ShutdownHook: stopping cluster node

Cluster left the cluster

3. **Start the DefaultCacheServer that you created in "Launching a Cache Server in  
the Eclipse IDE" on page 2-7. Ensure that InsertValue appears in the Project  
field in the Main tab and that local storage is enabled in the Coherence tab. Click  
Apply, then Run.**

4. **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 Example 3–5. The Gene Smith value stored by MyFirstSample is  
returned by MyFirstSampleReader.**

**Example 3–5  Output of MyFirstSampleReader Program with a Running Cache Server**

2011-03-14 17:18:01.678/0.282 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded  
operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.  
jar/tangosol-coherence.xml"  
2011-03-14 17:18:01.725/0.329 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded  
operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.  
jar/tangosol-coherence-override-dev.xml"  
2011-03-14 17:18:01.756/0.360 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded  
operational overrides from  
"file:/C:/home/oracle/workspace/InsertValue/build/classes/tangosol-coherence-override.xml"  
2011-03-14 17:18:01.771/0.375 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional  
configuration override "/custom-mbeans.xml" is not specified

3-6 Oracle Coherence Tutorial for Oracle Coherence
Creating Your First Coherence-Based Java Program

Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-14 17:18:02.131/0.735 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8090 using SystemSocketProvider
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service Cluster with senior member 1
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service Management with senior member 1
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service DistributedCache with senior member 1
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service ReplicatedCache with senior member 1
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service OptimisticCache with senior member 1
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=n/a): Member 1 joined Service InvocationService with senior member 1
2011-03-14 17:18:02.631/1.235 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB

Group{Address=224.3.7.0, Port=3155, TTL=4}

MasterMemberSet
{
  ThisMember=Member [Id=3, Timestamp=2011-03-14 17:18:02.459, Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.com, machine:tpfaeffl-lap7, process:424, Role=OracleHandsonMyFirstSampleReader]
  OldestMember=Member [Id=1, Timestamp=2011-03-14 17:16:43.021, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com, machine:tpfaeffl-lap7, process:2692, Role=CoherenceServer]
  ActualMemberSet=MemberSet [Size=2, BitSetCount=2]
    Member [Id=3, Timestamp=2011-03-14 17:18:02.459, Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.com, machine:tpfaeffl-lap7, process:424, Role=OracleHandsonMyFirstSampleReader]
    Member [Id=1, Timestamp=2011-03-14 17:16:43.021, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com, machine:tpfaeffl-lap7, process:2692, Role=CoherenceServer]
}

RecycleMillis=1200000
RecycleSet=MemberSet [Size=0, BitSetCount=0]

TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}

2011-03-14 17:18:02.662/1.266 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=3): Service Management joined the cluster with senior service member 1
2011-03-14 17:18:02.756/1.360 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=3): Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar/coherence-cache-config.xml"
Creating Your First Coherence-Based Java Program

This should not be the case for a process that joins the cluster only to perform an operation, such as entering a value and then terminating, like MyFirstSample. By default, all processes start as storage-enabled. The process can store data as part of the cluster. Modify the process so that it is not storage-enabled.

a. Right click the MyFirstSample class, select Run As then Run Configurations. See "Creating a Runtime Configuration" on page 2-12 for detailed information.

b. In the Coherence tab, select Disabled (cache client) under Local storage. This similar to setting the Java parameter

\[-Dtangosol.coherence.distributed.localstorage=false\]

5. Shut down any running cache servers and rerun your MyFirstSample class. You will receive a message similar to the one in Example 3–6 indicating that storage is not enabled on the cluster, because you have set this member to be storage-disabled.

**Example 3–6  Output of the MyFirstSample Class with Cache Storage Disabled**

```
2011-03-14 17:23:46.896/0.296 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2011-03-14 17:23:46.959/0.359 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2011-03-14 17:23:46.990/0.390 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "file:/C:/home/oracle/workspace/InsertValue/build/classes/tangosol-coherence-override.xml"
2011-03-14 17:23:46.990/0.390 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified
```

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

```
2011-03-14 17:23:47.256/0.656 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2011-03-14 17:23:47.443/0.843 Oracle Coherence 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider
2011-03-14 17:23:50.990/4.390 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB
Group(Address=224.3.7.0, Port=3155, TTL=4)
MasterMemberSet
```
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.

To perform this exercise, you must complete "Testing a Coherence Installation" on page 1-2.

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 can use the services of the cluster. Coherence-based applications typically do not connect to a cluster of applications; they become part of the cluster.

1. Create the Console Application

6. Restart the DefaultCacheServer cache server and run MyFirstSample and MyFirstSampleReader again. You should now see that the data is persisted between running the two Java examples.
2. **Run the Console Application**

**Create the Console Application**

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 `C:\oracle\product\coherence\doc` folder.

2. Write 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 2-11 for detailed information on creating a Java class.
   
a. Examine the methods that are available in the `NamedCache` interface using the Javadoc.

b. 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).

c. 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).

d. Write the value to standard output using the `System.out.println(....)` method.

Example 3–7 illustrates a sample Coherence-based Java application:

**Example 3–7  A Coherence-Based Java Application**

```java
package com.oracle.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();
    }
}
```
Run the Console Application

To run the Coherence application.


2. Create a run configuration for a cache client that uses Coherence Query Language and the QueryHelper API. Right click the project and select Run As then Run Configurations.

   **Note:** For more information about Coherence Query Language, see "Using Coherence Query Language" in Developer’s Guide for Oracle Coherence.

   a. In the Run Configurations dialog box click Oracle Coherence then the New launch configuration icon. Enter QueryPlus as the Name of the configuration.

   b. In the Main tab, under Project click the Browse button and select the InsertValue project. Under Main class, select Include system libraries when searching for a main class and click the Search button. In the Select Main Type dialog box, enter QueryPlus and select QueryPlus - com.tangosol.coherence.dslquery. Click OK. The Main tab should look similar to Figure 3–1.

   ![Figure 3–1 Main Tab for the Query Client Configuration](image)

   c. In the Coherence tab, select Disabled (cache client) under Local storage. Enter a unique value for the Cluster port (the value must be the same as the value defined for the cache server you defined in the previous section).

   d. In the Arguments tab, enter -showversion in the VM arguments field.

   e. In the Common tab, select Shared file and click the Browse button to navigate to the \InsertValue project name. Ensure that the Allocate console checkbox is selected. Click Apply.

3. Click Run to start the QueryPlus client. You should see output similar to Example 3–8 in the Eclipse Console.

   **Example 3–8 Output for the QueryPlus Cache Client**

   java version '1.6.0_14'
   Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Creating Your First Coherence-Based Java Application

Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode)
Coherence Command Line Tool
jline library cannot be loaded, so you cannot use the arrow keys for line editing and history.

CohQL>

4. At the CohQL> prompt, enter the following command to create a cache named mycache and to connect to it.

CohQL> create cache "mycache"

5. Create a run configuration for YourFirstCoherenceApplication. In the Run Configurations dialog box, enter YourFirstCoherenceApplication in the Name field. In the Main tab, enter InsertValue in the Project field and com.oracle.handson.YourFirstCoherenceApplication as the Main class.

In the Coherence tab, select Disabled (cache client) under Local storage, and enter a unique value for the Cluster port (this must be the same value that you used for the cache server and cache client).

6. Execute YourFirstCoherenceApplication from the Eclipse IDE and view the result.

Example 3–9 illustrates the output from YourFirstCoherenceApplication. The output indicates that there is no data in the cache for the message key.

```
Example 3–9  Output of the YourFirstCoherenceApplication Class
...
TcpRing{Connections=[2]}
IpMonitor{AddressListSize=0}
2011-03-14 18:01:15.271/1.312 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=4): Service Management joined the cluster with senior service member 1
2011-03-14 18:01:15.381/1.422 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=4): Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2011-03-14 18:01:15.443/1.484 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache, member=4): Service DistributedCache joined the cluster with senior service member 1
null
2011-03-14 18:01:15.475/1.516 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=4): Service Management left the cluster
2011-03-14 18:01:15.475/1.516 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache, member=4): Service DistributedCache left the cluster
2011-03-14 18:01:15.506/1.547 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=4): Service Cluster left the cluster
```

7. Using the running QueryPlus cache client in the Eclipse Console, change the key message. For example, enter the following at the CohQL> prompt:

CohQL> insert into "mycache" key "message" value 'hello'

Rerun YourFirstCoherenceApplication from the Eclipse IDE to see the changed values. Example 3–10 illustrates that the cache now holds the value hello for the key message.

```
Example 3–10  Output of the YourFirstCoherenceApplication Class with a New Key Value
...
TcpRing{Connections=[2]}
```
8. In the run configuration for YourFirstCoherenceApplication, change the value of the Local storage from Enabled to Disabled. Notice that the output is the same as the previous run.

9. Shut down your cache server and cache client instances. Restart the cache server and then rerun YourFirstCoherenceApplication (with the new value for Local storage). Note the output is now null.

10. If you change the value of the message key in your application (using the put method), verify that the new value available through the cache client.
   a. For example, comment out the get method and add the put method.
      ```java
      //String message = (String)myCache.get("message");
      String message = (String)myCache.put("message", "bye");
      ```
   b. Run YourFirstCoherenceApplication.
   c. Run the get command in the QueryPlus cache client.
      ```
      select key(), value() from "mycache" where key() is 'message'
      ```
      The output bye is displayed.
In this chapter, you work with complex objects located in the cache. Using Eclipse, you create a new Contact class, and then store and retrieve Contact objects in the cache using Portable Object Format (POF) serialization.

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 cache. 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:

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

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

Because Coherence might 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) the bytes into an active object at a future time. For example, objects that implement the java.io.Serializable interface are serializable.

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

The PortableObject interface provides two simple methods, readExternal and writeExternal, that permit you to explicitly read and write serialized object attributes from the provided PofReader and PofWriter streams respectively. By taking control over the serialization format, Coherence provides a way to improve the performance of the process. Using POF reduces the size of the resulting binary file. The size of the binary file is often 5 to 10 times smaller, and the conversion to or from the binary file 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. You also use POF serialization to put the objects in the cache and retrieve them by implementing the PortableObject interface.

1. Create the Data Objects
2. Create the Complex Object
3. Create the Driver Class
4. Create the POF and Cache Configuration Files
5. Run the Sample Project

Create the Data Objects

This section describes how to create two data objects that will later be incorporated into another data object. An Address object will provide employee address information and a PhoneNumber object will provide telephone contact information.

1. Create an Address object to store address information for an employee.
   a. Create a new Application Client Project in Eclipse called Contacts. Ensure that the CoherenceConfig is selected in the Configuration field on the opening page and the Create a default main is not selected on the Application Client module page.
      See "Creating a New Project in the Eclipse IDE" on page 2-3 for detailed information.
   b. Create a new Java class called Address. Ensure that the Default Package is com.oracle.handson. Do not select the Main Method check box. See "Creating a Java Class" on page 2-11 for detailed information.
   c. Write the class to use the PortableObject interface for data serialization. In the Eclipse code editor, change your generated Address class to implement com.tangosol.io.pof.PortableObject. Add an import statement for the PortableObject interface.
   e. Add the default public constructor for Address that is required by the PortableObject interface.
   f. 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.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()
    {
    }
}

g. Eclipse can generate the default get and set methods for your attributes. From the Source menu, select Generate Getters and Setters. Click Select All to select all of the attributes in the class. All of the attributes are now automatically selected. Click OK to continue.

    Figure 4–1 illustrates the Generate Getters and Setters dialog box with the generated accessors for the Address class.
Figure 4–1 Generate Getters and Setters Dialog Box

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

Figure 4–2 illustrates the Generate Constructor using Fields dialog box with the Street1, Street2, City, State, Zip, and Country fields selected.
Add "this" to the members of the generated constructor. The generated constructor should then look similar to the following:

```java
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;
}
```

i. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface. For example, the following implementation of the `readExternal` method enables the values for the street, city, state, and country to be read as POF objects.

```java
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));
}
```

j. Implement the `equals`, `hashCode`, and `toString` object methods.
Creating and Caching Complex Objects

**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 nontransient 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 the equals() method returns true must serialize identically; most built-in Java types meet this requirement.

To support these methods, import the com.tangosol.util.Base and com.tangosol.util.HashHelper classes. The Base class provides support for the equals method. The HashHelper class contains helper functions for calculating hash code values for any group of Java intrinsics.

The following code illustrates a sample implementation of the equals() method:

```java
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());
}
```

The following code illustrates a sample implementation of the hashCode() method:

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

The following code illustrates a sample implementation of the toString() method:

```java
public String toString()
{
    return getStreet1() + "\n" +
           getStreet2() + "\n" +
```
Creating and Caching Complex Objects

Working with Complex Objects

k. The resulting class should look similar to Example 4–1.

Example 4–1  Implementation of an Address Class

```java
package com.oracle.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 PhoneNumber class to store telephone contact data.
   
   a. Create a new Java class called PhoneNumber. Do not include a main method. See "Creating a Java Class" on page 2-11 for detailed information.
   
   b. Use the PortableObject interface for data serialization. In the Eclipse code editor, change your generated PhoneNumber class to implement PortableObject. Add an import statement for the com.tangosol.io.pof.PortableObject interface.

d. Add the default public constructor for the PhoneNumber class that is required by the PortableObject interface.

e. Enter the following private attributes for your PhoneNumber class. You can add others.

- short AccessCode
- short CountryCode
- short AreaCode
- int LocalNumber

f. Eclipse can generate the default get and set methods for your attributes. From the Source menu, select Generate Getters and Setters. Click Select All to select all of the attributes in the class. All of the attributes are now automatically selected. Click OK to continue.

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

Add "this" to the members of the generated constructor. The generated constructor then looks similar to the following:

```java
public PhoneNumber(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 looks similar to Example 4–2.

**Example 4–2 Implementation of a PhoneNumber Class**

```java
package com.oracle.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 PhoneNumber implements PortableObject {

    private short AccessCode;
    private short CountryCode;
    private short AreaCode;
```

---

4-10 Oracle Coherence Tutorial for Oracle Coherence
private int LocalNumber;

//------------ constructors --------------------------------
/**
 * Default constructor (necessary for PortableObject implementation).
 */
public PhoneNumber() {
}

public PhoneNumber(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;
    }
    PhoneNumber that = (PhoneNumber) 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();
}
Create the Complex Object

The Contact object provides the name, address, and telephone information of employees by incorporating the Address and PhoneNumber data objects.

1. Create a new Java class called Contact. Do not include a main method. See "Creating a Java Class" on page 2-11 for more information.

2. Because the class uses the PortableObject interface for data serialization, change your generated Contact class to implement PortableObject in the Eclipse code editor. Add an import statement for the com.tangosol.io.pof.PortableObject interface.


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

6. Eclipse can generate the default get and set methods for your attributes. From the Source menu, select Generate Getters and Setters. Click Select All to select all of the attributes in the class. Click OK to continue.

7. Create an accessor, getAge, to calculate the age of an employee:

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

8. Add a definition for MILLIS_IN_YEAR.

   ```java
   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 using Fields, click Select All, and then click OK.

   The generated constructor looks similar to the following:

   ```java
   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 looks similar to Example 4–3.

**Example 4–3  Sample Contact Class**

```java
package com.oracle.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() {
        super();
        this.FirstName = FirstName;
        this.LastName = LastName;
        this.HomeAddress = HomeAddress;
        this.WorkAddress = WorkAddress;
        this.TelephoneNumbers = TelephoneNumbers;
        this.BirthDate = BirthDate;
    }

    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;
    }

    @Override
    public String toString() {
        return super.toString() + "FirstName: " + FirstName + ", LastName: " + LastName + ", HomeAddress: " + HomeAddress + ", WorkAddress: " + WorkAddress + ", TelephoneNumbers: " + TelephoneNumbers + ", BirthDate: " + BirthDate;
    }

    @Override
    public int hashCode() {
        int result = super.hashCode();
        result = 31 * result + (FirstName != null ? FirstName.hashCode() : 0);
        result = 31 * result + (LastName != null ? LastName.hashCode() : 0);
        result = 31 * result + (HomeAddress != null ? HomeAddress.hashCode() : 0);
        result = 31 * result + (WorkAddress != null ? WorkAddress.hashCode() : 0);
        result = 31 * result + (TelephoneNumbers != null ? TelephoneNumbers.hashCode() : 0);
        result = 31 * result + (BirthDate != null ? BirthDate.hashCode() : 0);
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Contact other = (Contact) obj;
        return 
```
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.
 */
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());
    sb.append(' ').
        append(getLastName())
        .append('
Addresses')
        .append('
Home: ').append(getHomeAddress())
        .append('
Work: ').append(getWorkAddress())
        .append('
Telephone Numbers');
    for (Iterator iter = TelephoneNumbers.entrySet().iterator();
        iter.hasNext(); )
    {
        Map.Entry entry = (Map.Entry) iter.next();
        sb.append('
');

d
Create 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. Ensure that it includes a main method.

   See "Creating a Java Class" on page 2-11 for detailed information.

2. In the ContactDriver class, 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 4-4 illustrates a sample implementation of this class.

Example 4-4  Sample ContactDriver Class

```java
package com.oracle.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);
        PhoneNumber phonenumber = new PhoneNumber((short)11, (short)650, (short)506, 7000);
        Map map = new HashMap();
        map.put("home", phonenumber);

        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!!");
}
}
}

**Create the POF and Cache Configuration Files**

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. You must also specify POF serialization and the name of the POF configuration file in the cache configuration file.

1. Create a POF configuration file for the **Contact**, **Address**, and **PhoneNumber** objects.
   a. To create a POF configuration file for your data types, use the `pof-config.xml` file that was created with your project.
   b. Define `<user-type>` elements for the **Contact**, **Address**, and **PhoneNumber** objects, assign type IDs 1001, 1002, and 1003 to them and provide their full class names. The file must include the `coherence-pof-config.xml` file which reserves the first 1000 IDs for Coherence data types.
   c. Rename the file in the **Project Explorer** and save it as `contacts-pof-config.xml` (it will be saved to the `C:\home\oracle\workspace\Contacts\appClientModule` folder). Example 4-5 illustrates a sample `contacts-pof-config.xml` file.

**Example 4-5  POF Configuration File**

```xml
<?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.handson.Contact</class-name>
        </user-type>
        <user-type>
            <type-id>1002</type-id>
            <class-name>com.oracle.handson.Address</class-name>
        </user-type>
        <user-type>
            <type-id>1003</type-id>
            <class-name>com.oracle.handson.PhoneNumber</class-name>
        </user-type>

    </user-type-list>
    <allow-interfaces>true</allow-interfaces>
</pof-config>
```
2. Create a cache configuration file. You can use the coherence-cache-config.xml file in the Project Explorer, which is based on the cache-config.dtd file. You can also find a copy of coherence-cache-config.xml file in the coherence.jar file.

a. Use ExamplesPartitionedPofScheme as the scheme-name and PartitionedPofCache as the service-name.

b. The serializer section is responsible for mapping a POF serialized object to an appropriate serialization routine which is either a PofSerializer or by calling through the PortableObject interface. In this case, use the com.tangosol.io.pof.ConfigurablePofContext class.

c. Use the <init-param> section to point to the name of the POF configuration file, in this case contacts-pof-config.xml.

d. Rename the file in the Project Explorer and save it as contacts-cache-config.xml (it will be saved to the C:\home\oracle\workspace\Contacts\appClientModule folder). Example 4-6 illustrates a sample contacts-cache-config.xml file.

**Example 4-6  Cache Configuration File**

```xml
<?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>
      <service-name>PartitionedPofCache</service-name>
      <serializer>
        <instance>
          <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>
        </instance>
      </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>
    </distributed-scheme>
    </caching-schemes>
  </cache-config>
</cache-config>
```

Run the Sample Project

The contacts project requires a cache server to be running. The cache server and the application must reference the same POF and cache configuration files. This section describes how to create run configurations for the cache server and the application.

1. Stop any cache servers that are running. See "Stopping Cache Servers" on page 2-13 for detailed information.

2. Create an executable file to start the cache server.
   a. Right click the project and select Run As then Run Configurations. In the Run Configurations dialog box, enter ContactsCacheServer in the Name field.
   b. In the Main tab, ensure that Contacts appears in the Project field. Select the Include system libraries when searching for a main class checkbox and click the Search button. In the Select Main Type dialog box, enter DefaultCacheServer and select the DefaultCacheServer - com.tangosol.net class. Click OK in the Select Main Type dialog box, then Apply in the Run Configurations dialog box.
   c. In the General tab of the Coherence tab, select Absolute File Path in the Topology field and browse to the contacts-cache-config.xml file in the C:\home\oracle\workspace\Contacts\appClientModule directory. Ensure that Enabled (cache server) is selected. Enter a unique value for the Cluster port. Click Apply. The General tab should look similar to Figure 4–3.
d. In the Other tab of the Coherence tab, scroll down to the tangosol.pof.config item. Enter the path to the POF configuration file: C:\home\oracle\workspace\Contacts\appClientModule\contacts-pof-config.xml.

e. In the Arguments tab, enter -showversion under VM Arguments.

f. In the Classpath tab, ensure that the Contacts project appears under User Entries.

g. In the Common tab, select Shared file and browse for the \Contacts project.

h. Click Run to start the Contacts cache server. The output should be similar to Example 4–7.

Example 4–7  Output from the Contacts Cache Server

```
java version '1.6.0_14'
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode)
2011-03-15 15:30:10.095/0.125 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml'
2011-03-15 15:30:10.095/0.125 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml'
2011-03-15 15:30:10.095/0.125 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'file:/C:/home/oracle/workspace/Contacts/build/classes/tangosol-coherence-override.xml'
```
Creating and Caching Complex Objects

2011-03-15 15:30:10.095/0.125 Oracle Coherence 3.7.0.0 <D5> {thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-15 15:30:10.267/0.297 Oracle Coherence GE 3.7.0.0 <Info> {thread=main, member=n/a): Loaded cache configuration from "file:/C:/home/oracle/workspace/Contacts/appClientModule/contacts-cache-config.xml"

2011-03-15 15:30:10.455/0.485 Oracle Coherence GE 3.7.0.0 <D4> {thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider


2011-03-15 15:30:13.970/4.000 Oracle Coherence GE 3.7.0.0 <Info> {thread=main, member=n/a): Started cluster Name=cluster:0x96AB

Group(Address=224.3.7.0, Port=3155, TTL=4)

MasterMemberSet
{
    ThisMember=Member(Id=1, Timestamp=2011-03-15 15:30:10.47, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4780, Role=CoherenceServer)
    OldestMember=Member(Id=1, Timestamp=2011-03-15 15:30:10.47, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4780, Role=CoherenceServer)
    ActualMemberSet=MemberSet(Size=1, BitSetCount=2
        Member(Id=1, Timestamp=2011-03-15 15:30:10.47, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4780, Role=CoherenceServer)
    )
    RecycleMillis=1200000
    RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
}

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2011-03-15 15:30:14.017/4.047 Oracle Coherence GE 3.7.0.0 <D5> {thread=Invocation:Management, member=1): Service Management joined the cluster with senior service member 1
2011-03-15 15:30:14.220/4.250 Oracle Coherence GE 3.7.0.0 <D5> {thread=DistributedCache:PartitionedPofCache, member=1): Service PartitionedPofCache joined the cluster with senior service member 1
2011-03-15 15:30:14.252/4.282 Oracle Coherence GE 3.7.0.0 <Info> {thread=main, member=1): Services
{
    ClusterService(Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.7pre, OldestMemberId=1)
    InvocationService(Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1)
Creating and Caching Complex Objects

PartitionedCache(Name=PartitionedPofCache, State=(SERVICE_STARTED), LocalStorage=enabled, PartitionCount=257, BackupCount=1, AssignedPartitions=257, BackupPartitions=0)

Started DefaultCacheServer...

3. Create a run configuration for the ContactDriver executable.
   - Right click the ContactDriver.java file in the Project Explorer and select Run As then Run Configurations.
   - In the Run Configurations dialog box, double-click the Oracle Coherence node. Enter ContactsDriver in the Name field. In the Main tab, browse for Contacts in the Project field, and com.oracle.handson. ContactDriver in the Main class field. Click Apply.
   - In the General tab of the Coherence tab, browse for the absolute path to the contacts-cache-config.xml file in the Cache configuration descriptor field. Select the Disable (cache client) radio button. In the Cluster port field, enter 3155.
   - In the Other tab of the Coherence tab, scroll down to the tangosol.pof.config field and replace the value with the absolute path to the contacts-pof-config.xml POF configuration file.
   - In the Classpath tab, ensure that Coherence37 appears under Bootstrap Entries. Ensure that Contacts (default classpath) appears under User Entries.

4. Click Run to run the ContactDriver configuration. The output of the Contacts example should look similar to Example 4–8.

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

Example 4–8 Output of the Contacts Example in the Eclipse IDE

2011-03-15 15:31:50.970/0.125 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml'
2011-03-15 15:31:50.970/0.125 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml'
2011-03-15 15:31:50.970/0.125 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'file:/C:/home/oracle/workspace/Contacts/build/classes/tangosol-coherence-override.xml'
2011-03-15 15:31:50.986/0.141 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override '/custom-mbeans.xml' is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-15 15:31:51.142/0.297 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from 'file:/C:/home/oracle/workspace/Contacts/appClientModule/contacts-cache-config.xml'
2011-03-15 15:31:51.345/0.500 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8090 using SystemSocketProvider
2011-03-15 15:31:51.799/0.954 Oracle Coherence GE 3.7.0.0 <Info> (thread=Cluster, member=n/a): This Member(Id=2, Timestamp=2011-03-15 15:31:51.619, Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5720, Role=OracleHandsonContactDriver,
Creating and Caching Complex Objects

5. If you go back and look at the cache server output, you can see messages reporting Contacts cache client joining the cluster, completing its work, then leaving the cluster. See Example 4–9.

Example 4–9 Contacts Cache Server Displaying the Arrival and Departure of the Contacts Client

... Started DefaultCacheServer...
2011-03-15 15:31:51.845/101.875 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 joined Service Management with senior member 1
2011-03-15 15:31:51.986/102.016 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 joined Service PartitionedPofCache with senior member 1
2011-03-15 15:31:52.033/102.063 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): TcpRing disconnected from Member(Id=2, Timestamp=2011-03-15 15:31:51.619, Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5720, Role=OracleHandsonContactDriver) due to a peer departure; removing the member.
2011-03-15 15:31:52.033/102.063 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 left service Management with senior member 1
2011-03-15 15:31:52.033/102.063 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 left service PartitionedPofCache with senior member 1
2011-03-15 15:31:52.033/102.063 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member(Id=2, Timestamp=2011-03-15 15:31:52.033, Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5720, Role=OracleHandsonContactDriver) left Cluster with senior member 1
In this chapter, you 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 put objects into the cache and retrieved them individually. Each call to the `put` method can result in increased network traffic, especially for partitioned and replicated caches. Additionally, each call to a `put` method returns the object it just replaced in the cache, which adds more unnecessary overhead. By using the `addAll` method, loading the cache can be made much more efficient.

To perform the tasks in this chapter, you must first complete the project described in Chapter 4, "Working with Complex Objects." You must also be 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 shows you how to create a console application that populates a Coherence cache with domain objects. The application will use the Coherence `com.tangosol.io.pof.PortableObject` implementation to serialize the objects into Portable Object format (POF).

In the exercise, you create a key that helps to get the `Contact` object, a generator to provide data for the cache, and a loader to load the cache.

1. **Create a Class with the Key for the Domain Objects**
2. **Edit the POF Configuration File**
3. **Create the Data Generator**
4. **Create a Console Application to Load the Cache**
5. **Run the Cache Loading Example**
Create a Class with the Key for the Domain Objects

To create a class that contains the key for a domain object:

1. Create a new Application Client Project called Loading. Select CoherenceConfig from the Configuration drop-down list. In the Application Client Module page of the New Application Client Project wizard, deselect the Create a default Main class checkbox.

See "Creating and Caching Complex Objects" on page 4-2 for information on creating a new project.

2. Add the classes and files related to the Address, PhoneNumber, and Contact classes that you created in an earlier exercise (Contacts). These files can be found in c:\home\oracle\workspace\Contacts\appClientModule and c:\home\oracle\workspace\Contacts\build\classes directories.

Right click the Loading project in the Project Explorer and select Properties. In the Properties for Loading dialog box, select Java Build Path. In the Projects tab, click Add. Select the Contacts project from the Required Project Selection dialog box, as illustrated in Figure 5–1.

In the Order and Export tab, use the Up and Down buttons to move Contacts to the top of the list. The contents of the tab should look similar to Figure 5–2.
3. Create a `ContactId` class that provides a key to the employee for whom information is tracked. See “Creating a Java Class” on page 2-11 for 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 get the `Contact` object.

Because this class uses POF serialization, it must implement the `PortableObject` interface, 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 the `hashCode` and `equals` object methods 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 the `equals` method returns true must serialize identically; most built-in Java types meet this requirement.

Example 5–1 illustrates a possible implementation of the `ContactId` class.

**Example 5–1 Simple Contact ID Class**

```java
package com.oracle.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;

Edit the POF Configuration File

Edit the POF configuration file. Add a <user-type> entry for the ContactId class
to the contacts-pof-config.xml file. The file looks similar to Example 5–2.

Example 5–2 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.handson.Contact</class-name>
Create the Data Generator

Create a Java class named DataGenerator to generate random employee contact names and addresses. See "Creating a Java Class" on page 2-11 for detailed information.

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

Example 5–3 illustrates a possible implementation of the data generator. This implementation creates a text file, contacts.cvs, that contains the employee contact information.

Example 5–3  Sample Data Generation Class

```java
package com.oracle.handson;

import com.oracle.handson.Address;
import com.oracle.handson.Contact;
import com.oracle.handson.PhoneNumber;
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 ------------------------------------------
```
```java
/**
 * Generate contacts.
 */
public static void main(String[] asArg)
    throws IOException
{
    String sFile = asArg.length > 0 ? asArg[0] : 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 PhoneNumber getRandomPhone()
{
    int[] anPhone = getRandomPhoneDigits();
    return new PhoneNumber((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", "DF", "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",
    "VA", "WA", "WV", "WI", "WY"
};

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

The DataGenerator class generates, at random, information about an employee. The information includes the employee’s name, home address, work address, home telephone number, work telephone number, and the employee’s age. The information is stored in a CSV-formatted file named contacts.csv. This file follows the standard rules for a CSV-formatted file, where there is one record per line, and individual fields within the record are separated by commas.
Example 5–4 illustrates the first few entries of the contacts.csv file.

**Example 5–4  Contents of the contacts.csv File**

<table>
<thead>
<tr>
<th>Name</th>
<th>First Last</th>
<th>Address 1</th>
<th>City, State, ZIP</th>
<th>Country</th>
<th>Phone1</th>
<th>Phone2</th>
<th>Phone3</th>
<th>Home1</th>
<th>Home2</th>
<th>Home3</th>
<th>Home4</th>
<th>Work1</th>
<th>Work2</th>
<th>Work3</th>
<th>Work4</th>
</tr>
</thead>
</table>

... 

Create a Console Application to Load the Cache

Create a Java class called `LoaderExample`. See "Creating a Java Class" on page 2-11 for detailed information.

Implement the class to load the cache with employee data generated by the program described in "Create the Data Generator" on page 5-6. Use input streams and buffered readers to load the employee information in the contacts.csv file into a single Coherence cache.

Add code to parse the employee information in the data file. After you have this information, create 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 5–5 illustrates a possible implementation of the `LoaderExample` class.

**Example 5–5  Sample Cache Loading Program**

```java
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.oracle.handson.ContactId;
import com.oracle.handson.Address;
import com.oracle.handson.PhoneNumber;
import com.oracle.handson.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 file.
 */
public class LoaderExample {
```
/**
 * Load contacts.
 */
public static void main (String[] asArg)
        throws IOException
{
    String sFile  = asArg.length > 0 ? asArg[0] : DataGenerator.FILENAME;
    String sCache = asArg.length > 1 ? asArg[1] : CACHENAME;
    System.out.println("input file: " + sFile);
    System.out.println("cache name: " + sCache);
    CacheFactory.getCache(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 PhoneNumber(
               /*access code*/ Short.parseShort(asPart[ofPart++]),
               /*country code*/ Short.parseShort(asPart[ofPart++]));
Populating a Cache with Domain Objects

```java
/*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
Contact con1 = new Contact(sFirstName, sLastName, addrHome,
    addrWork, mapTelNum, dtBirth);
System.out.println(con1);
mapBatch.put(id, con1);
++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";
```

Run the Cache Loading Example

To run the cache loading example:


2. Start a cache server with the ContactsCacheServer file.
   a. Right click the project and select Run As then Run Configurations to edit the ContactsCacheServer configuration. In the Main tab, click Browse and select the Loading project.
   b. In the Classpath tab, select User Entries and click Add Projects. In the Project Selection dialog box, select the Loading project. Click OK.
   c. In the Common tab, click Shared file and browse for the Loading project.
   d. Click Apply, then Run.
3. Create a run configuration for DataGenerator. Right click DataGenerator in the Project Explorer and select Run As. In the Run Configurations dialog box select Oracle Coherence and click the New Configuration icon.

   a. In the Name field, enter DataGenerator.

   b. In the Project field in the Main tab, enter Loading. In the Main class field, enter com.oracle.handson.DataGenerator.

   c. In the General tab of the Coherence tab, browse to the c:\home\oracle\workspace\Contacts\appClientModule\coherence-cache-config.xml file in the Cache configuration descriptor field. Select the Disabled (cache client) button. Enter 3155 in the Cluster port field. Click Apply.

   In the Other tab, scroll down to the tangosol.pof.config field. Enter the absolute path to the POF configuration file contacts-pof-config.xml. Click Apply.

   d. In the Common tab, select Shared file and browse for the Loading directory.

   e. Examine the contents of Loading in the Classpath tab. Contacts should appear as one of the entries under Loading, as in Figure 5–3.

   **Figure 5–3 Classpath for the DataGenerator Program**

   ![Classpath for the DataGenerator Program](image)

4. Create a run configuration for LoaderExample.

   a. In the Name field, enter LoaderGenerator.

   b. In the Project field in the Main tab, enter Loading. In the Main class field, enter com.oracle.handson.LoaderExample.

   c. In the General tab of the Coherence tab, browse to the c:\home\oracle\workspace\Contacts\appClientModule\coherence-cache-config.xml file in the Cache configuration descriptor field. Select the Disabled (cache client) button. Enter 3155 in the Cluster port field. Click Apply.

   In the Other tab, scroll down to the tangosol.pof.config field. Enter the absolute path to the POF configuration file contacts-pof-config.xml. Click Apply.

   d. Examine the contents of the Classpath tab. Contacts should appear as one of the entries under Loading, as in Figure 5–4.
5. Run the DataGenerator configuration from the Run Configurations dialog box. Then run the LoaderExample configuration.

The DataGenerator run configuration generates a data file; you should not see any output or response in the Eclipse console window. The LoaderExample run configuration sends a stream of employee contact information, similar to Example 5–6, to the Eclipse console window.

Example 5–6 Output from the Sample Cache Loading Program

...  
2011-03-15 17:07:51.413/2.266 Oracle Coherence GE 3.7.0.0 <DS> (thread=Invocation:Management, member=2): Service Management joined the cluster with senior service member 1  
2011-03-15 17:07:51.413/2.266 Oracle Coherence GE 3.7.0.0 <Info> (thread=Cluster, member=2): Loaded POF configuration from "file:/C:/home/oracle/workspace/Contacts/appClientModule/contacts-pof-config.xml"  
2011-03-15 17:07:51.413/2.266 Oracle Coherence GE 3.7.0.0 <Info> (thread=Cluster, member=2): Loaded included POF configuration from "jar:file:/C:/coherence360/coherence/lib/coherence.jar!/coherence-pof-config.xml"  
2011-03-15 17:07:51.413/2.266 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache:PartitionedPofCache, member=2): Service PartitionedPofCache joined the cluster with senior service member 1  
2011-03-15 17:07:51.413/2.266 Oracle Coherence GE 3.7.0.0 <D4> (thread=DistributedCache:PartitionedPofCache, member=2): Asking member 1 for 128 primary partitions  
John Hwolru  
Addresses  
Home: 742 Beacon St.  
Icymz, OH 74135  
US  
Work: Yoyodyne Propulsion Systems  
330 Lectroid Rd.  
Grover's Mill, WY 20222  
US  
Telephone Numbers  
work: +1 50 928 2637858  
home: +1 72 403 7946780  
Birth Date: 1981-12-28  
...  
John Unglg
Querying and Aggregating Data in the Cache

This exercise introduces querying and aggregating data in a cache. This exercise shows you how to:

- Query the cache for specific data
- Aggregate information within the cache

After putting complex objects in the named caches, you can query and aggregate 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 it, you also have the overhead of deserializing when querying information. When indexes are added, the values kept in the index itself are deserialized and, therefore, are quicker to access. The objects that are indexed are always kept serialized.

Some of the often-used methods in the QueryMap interface are:

- Set entrySet(Filter filter), which returns a set of entries that are contained in the map that satisfy the filter
addIndex(ValueExtractor extractor, boolean fOrdered, Comparator comparator), which adds an index

Set.keySet(Filter filter), which is 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 performed 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 NamedCache objects implement the com.tangosol.util.QueryMap interface. This enables NamedCache objects to support searching for keys or entries in a cache that satisfy a specified 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 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 does not have a get method, then the test fails.

Here are additional examples of using the Filter classes:

- 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 the age of 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 lets you define indexes to improve query performance by using the addIndex method. 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. Because there is no schema, you define indexes differently from a traditional database.

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

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

```
new ReflectionExtractor("getCity")
```

You can use value extractors 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, from left to right. The result of a previous extractor becomes the target object for a next extractor, for example:

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

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

1. Create the Class to Query Cache Data
2. Run the Query Example
3. Edit the Query Example to Perform Aggregations
4. Run the Query and Aggregation Example

## Create the Class to Query Cache Data

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 2-11 for detailed information.

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

- All employees who live in Massachusetts:
  ```java
cache.entrySet(new EqualsFilter("getHomeAddress.getState", "MA"));
```

- All employees who live in Massachusetts and work elsewhere:
  ```java
cache.entrySet(new AndFilter(
      new EqualsFilter("getHomeAddress.getState", "MA"),
      new NotEqualsFilter("getWorkAddress.getState", "MA")));
```

- All employees whose city name begins with an S:
  ```java
cache.entrySet(new LikeFilter("getHomeAddress.getCity", "S%"));
```

- All employees whose last name begins with an S and live in Massachusetts. Use both the key and value in the query. Note that the cache entries use ContactId objects as the keys. You can use the KeyExtractor to get these values. KeyExtractor is a specialized value extractor that indicates that a query be run against the key objects, rather than the values:
  ```java
cache.entrySet(new AndFilter(
      new EqualsFilter("getHomeAddress.getCity", "S"),
      new NotEqualsFilter("getWorkAddress.getState", "MA"),
      new KeyExtractor())
```

```java
```
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;
  Set results = cache.entrySet(new GreaterFilter("getAge", nAge));

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

Example 5–7 illustrates a possible implementation of the QueryExample class.

Example 5–7  Sample QueryExample Class

```java
package com.oracle.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 indexes 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);
Run the Query Example

To run the query example:


2. Create a run configuration for the QueryExample class.
   a. Right click QueryExample in the Project Navigator and select Run As then Run Configurations. Select Oracle Coherence and click the New launch configuration icon.
   b. In the Name field, enter QueryExample.
   c. In the Project field in the Main tab, enter Loading. In the Main class field, enter com.oracle.handson.QueryExample.
   d. In the General tab of the Coherence tab, browse to the c:\home\oracle\workspace\appClientModule\coherence-cache-config.xml file in the Cache configuration descriptor field. Select the Disabled (cache client) button. Enter 3155 in the Cluster port field. Click Apply.
   In the Other tab, scroll down to the tangosol.pof.config field. Enter the absolute path to the POF configuration file contacts-pof-config.xml. Click Apply.
   e. Examine the contents of the Classpath tab. The Contacts project should appear under the Loading project in User Entries.

3. Restart the ContactsCacheServer.

4. Run the DataGenerator class, the LoaderExample class, then the QueryExample class.

After printing all of the contact information in the cache, the QueryExample file displays the results of the queries. The results should look similar to Example 5–8.

Example 5–8  Results of the QueryExample Program

... 
MasterMemberSet
{
    ThisMember=Member(Id=3, Timestamp=2011-03-15 17:36:07.335, Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.com,role=OracleHandsonQueryExample)
    OldestMember=Member(Id=1, Timestamp=2011-03-15 17:35:09.585, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,role=CoherenceServer)
    ActualMemberSet=MemberSet(Size=2, BitSetCount=2
        Member(Id=1, Timestamp=2011-03-15 17:35:09.585, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,role=CoherenceServer)
        Member(Id=3, Timestamp=2011-03-15 17:35:09.585, Address=130.35.99.213:8089, MachineId=49877, Location=site:us.oracle.com,role=OracleHandsonQueryExample)
    )
    RecycleMillis=1200000
    RecycleSet=MemberSet(Size=0, BitSetCount=0
        Member(Id=1, Timestamp=2011-03-15 17:35:09.585, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,role=CoherenceServer)
    )
} 
TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}
2011-03-15 17:36:07.553/1.296 Oracle Coherence GE 3.7.0.0 <D5>
(thread=Invocation:Management, member=3): Service Management joined the cluster with senior service member 1
2011-03-15 17:36:07.647/1.390 Oracle Coherence GE 3.7.0.0 <Info>
(thread=DistributedCache:PartitionedPofCache, member=3): Loaded POF configuration from "file:/C:/home/oracle/workspace/Contacts/appClientModule/contacts-pof-config.xml"
2011-03-15 17:36:07.663/1.406 Oracle Coherence GE 3.7.0.0 <Info>
(thread=DistributedCache:PartitionedPofCache, member=3): Loaded included POF configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-pof-config.xml"
2011-03-15 17:36:07.710/1.453 Oracle Coherence GE 3.7.0.0 <D5>
(thread=DistributedCache:PartitionedPofCache, member=3): Service PartitionedPofCache joined the cluster with senior service member 1

**MA Residents**

ConverterEntry(Key="John Ueccc", Value="John Ueccc
Addresses
Home: 285 Beacon St.

Oxtqwisgti, MA 41063
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, VA 28107
US
Telephone Numbers
work: +11 48 835 1876678
home: +11 78 482 1247744
Birth Date: 1972-12-30"

...

**MA Residents, Work Elsewhere**

ConverterEntry(Key="John Ueccc", Value="John Ueccc
Addresses
Home: 285 Beacon St.

Oxtqwisgti, MA 41063
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, VA 28107
US
Telephone Numbers
work: +11 48 835 1876678
home: +11 78 482 1247744
Birth Date: 1972-12-30"

...

**City Begins with S**

ConverterEntry(Key="John Frepojf", Value="John Frepojf
Addresses
Home: 851 Beacon St.

Swmsfng, PR 00734
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, AR 97794
US
Telephone Numbers
work: +11 10 474 781020
Querying and Aggregating Data in the Cache

Edit the Query Example to Perform Aggregations

Add code to the QueryExample.java file to perform aggregations on the cache data. An entry aggregator (com.tangosol.util.InvocableMap.EntryAggregator) enables you to perform operations on all or a specific set of objects and return an aggregation. EntryAggregator instances are 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 5–9 illustrates the EntryAggregator methods that perform these tasks.

**Example 5–9 Methods to Aggregate Over Keys or by Specifying Filters**

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

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

To add aggregations to return data for filtering:
1. Using aggregations, write code in the `QueryExample` class to calculate the following:
   - The number of employees that are older than a specified age. Use the `GreaterFilter` and the `Count` class:
     ```java
     cache.aggregate(new GreaterFilter("getAge", nAge), new Count())
     ```
   - Lowest age in the set of employees. Use the `AlwaysFilter` and the `LongMin` class:
     ```java
     cache.aggregate(AlwaysFilter.INSTANCE, new LongMin("getAge"))
     ```
   - Highest age in the set of employees. Use the `AlwaysFilter` and the `LongMax` class:
     ```java
     cache.aggregate(AlwaysFilter.INSTANCE, new LongMax("getAge"))
     ```
   - Average age of employees. Use the `AlwaysFilter` and the `DoubleAverage` class:
     ```java
     cache.aggregate(AlwaysFilter.INSTANCE, new DoubleAverage("getAge")
     ```

2. Import the `Count`, `DoubleAverage`, `LongMax`, and `LongMin` aggregator classes.
   ```java
   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 looks similar to Example 5–10.

**Example 5–10 QueryExample with Aggregation**

```java
package com.oracle.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 indexes 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
```java
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());
    }
}
```

### Run the Query and Aggregation Example

To query the cache and aggregate the results:

2. Restart ContactsCacheServer.
3. Run the DataGenerator, LoaderExample, and QueryExample applications.

The output should look similar to Example 5–11 in the Eclipse Console.

**Example 5–11  Output from the Aggregators**

```plaintext
Last Name Begins with S and State Is MA
ConverterEntry(KeY="John Smeyas", Value="John Smeyas
Addresses
Home: 594 Beacon St.
Jxaxt, MA 10081
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover’s Mill, NJ 95236
US
Telephone Numbers
work: +11 70 837 4723938
home: +11 2 227 6816592
Birth Date: 1977-12-29"

count > 42: 482
min age: 22
avg age: 41.627
max age: 61
```
2011-03-15 17:47:26.663/2.078 Oracle Coherence GE 3.7.0.0 <D4>
(thread=ShutdownHook, member=3): ShutdownHook: stopping cluster node
2011-03-15 17:47:26.663/2.078 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=3): Service Cluster left the cluster
2011-03-15 17:47:26.663/2.078 Oracle Coherence GE 3.7.0.0 <D5>
(thread=Invocation:Management, member=3): Service Management left the cluster
In Chapter 5, "Loading Data Into a Cache," you created a file, QueryExample.java, that used a variety of filters, such as AlwaysFilter, LikeFilter, and EqualsFilter, to pull and aggregate information from the cache. The file also created indexes on the data by using a variety of specialized ValueExtractors: ChainedExtractors, KeyExtractors, and ReflectionExtractors. This created some verbose Java statements. These statements can be simplified by using the QueryHelper API.

This chapter contains the following sections:

■ Introduction
■ Simplifying the Query Example
■ Rerunning the Query Example

**Introduction**

To simplify filter and extractor statements, and the way in which you interact with the Coherence caches, Oracle Coherence provides the QueryHelper API. QueryHelper (com.tangosol.util.QueryHelper) is a utility class that provides a set of createFilter and createExtractor factory methods that can build instances of Filter and ValueExtractor. The methods in the class accept a String data type that specifies the creation of rich Filters in a format that is familiar to anyone who understands SQL WHERE clauses.

For example, the following statement uses createFilter(String s) to construct a filter for employees who live in Massachusetts but work in another state.

```java
...
QueryHelper.createFilter("homeAddress.state = 'MA' and workAddress.state != 'MA'")
...
```

This statement is more simple and easier to read than the equivalent filter and extractor statement using the Coherence API:

```java
new AndFilter(new EqualsFilter("getHomeAddress.getState", "MA"),
            new NotEqualsFilter("getWorkAddress.getState", "MA"))
```

For more information, see the Javadoc for the QueryHelper API. For information on the syntax of the WHERE clause within the Coherence Query Language, see "Using Coherence Query Language" in Developer’s Guide for Oracle Coherence.
Simplifying the Query Example

This section describes how you can simplify the indexes, cache calls, and aggregations in the QueryExample.java file that you created in the previous chapter.

1. Import the QueryHelper API into the QueryExample.java file as a static class.
   ```java
   import static com.tangosol.util.QueryHelper.*;
   ```

2. Comment out the imports for the ChainedExtractor, KeyExtractor, and ReflectionExtractor classes.

3. Comment out the imports for the AlwaysFilter, AndFilter, EqualsFilter, GreaterFilter, LikeFilter, and NotEqualsFilter classes.

4. In the cache.addIndex statements, replace instances of ReflectionExtractor with createExtractor from the QueryHelper API.
   Table 6–1 lists the ReflectionExtractor instances and their createExtractor equivalents.

<table>
<thead>
<tr>
<th>Replace This ReflectionExtractor Statement ...</th>
<th>With the Equivalent createExtractor Statement ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache.addIndex(new ReflectionExtractor(&quot;getAge&quot;), true, null);</td>
<td>cache.addIndex(createExtractor(&quot;age&quot;), true, null);</td>
</tr>
<tr>
<td>cache.addIndex(new ChainedExtractor(reflectAddrHome, new ReflectionExtractor(&quot;getState&quot;)), true, null);</td>
<td>cache.addIndex(createExtractor(&quot;homeAddress.state&quot;), false, null);</td>
</tr>
<tr>
<td>cache.addIndex(new ChainedExtractor(new ReflectionExtractor(&quot;getWorkAddress&quot;), new ReflectionExtractor(&quot;getState&quot;)), true, null);</td>
<td>cache.addIndex(createExtractor(&quot;workAddress.state&quot;), false, null);</td>
</tr>
<tr>
<td>cache.addIndex(new ChainedExtractor(reflectAddrHome, new ReflectionExtractor(&quot;getCity&quot;)), true, null);</td>
<td>cache.addIndex(createExtractor(&quot;homeAddress.city&quot;), true, null);</td>
</tr>
</tbody>
</table>

5. Replace the calls to the *Filter methods in the setResults statements with calls to createFilter with the appropriate Coherence Query Language.
   Table 6–2 lists the *Filter instances and their createFilter equivalents.

<table>
<thead>
<tr>
<th>Replace This *Filter Statement ...</th>
<th>With the Equivalent createFilter Statement ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set setResults = cache.entrySet(new EqualsFilter(&quot;getHomeAddress.getState&quot;, &quot;MA&quot;));</td>
<td>Set setResults = cache.entrySet(createFilter(&quot;homeAddress.state = 'MA'&quot;));</td>
</tr>
<tr>
<td>Set setResults = cache.entrySet(new AndFilter(new EqualsFilter(&quot;getHomeAddress.getState&quot;, &quot;MA&quot;), new NotEqualsFilter(&quot;getWorkAddress.getState&quot;, &quot;MA&quot;)));</td>
<td>setResults = cache.entrySet(createFilter(&quot;homeAddress.state is 'MA' and workAddress is not 'MA'&quot;));</td>
</tr>
</tbody>
</table>
Simplifying the Query Example

6. Replace the calls to the *Filter methods in the aggregate statements with calls to createFilter with the appropriate Coherence Query Language.

Table 6–3 lists the *Filter instances and their createFilter equivalents.

When you are finished with the code replacements, QueryExample.java looks similar to Example 6–1.

Example 6–1 Edited QueryExample File

```java
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.Filter;
import static com.tangosol.util.QueryHelper.*;

Set setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity", "S"));
Set setResults = cache.entrySet(createFilter("homeAddress.city like 'S%'"));

Set setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
Set setResults = cache.entrySet(createFilter("age > ?1", aEnv));

Set setResults = cache.entrySet(new AndFilter(new LikeFilter(new KeyExtractor("getLastName"), "S", (char) 0, false), new EqualsFilter("getHomeAddress.getState", "MA")));
Set setResults = cache.entrySet(createFilter("key(lastName) like 'S%' and homeAddress.state = 'MA'"));

System.out.println("count > " + nAge + ": " + cache.aggregate(new GreaterFilter("getAge", nAge), new Count()));
System.out.println("count > " + nAge + ": " + cache.aggregate(createFilter("age > ?1", aEnv), new Count()));

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

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

System.out.println("max age: " + cache.aggregate(AlwaysFilter.INSTANCE, new LongMax("getAge")));
System.out.println("max age: " + cache.aggregate(always, new LongMax("getAge")));
```
import com.tangosol.util.aggregator.Count;
// import com.tangosol.util.extractor.ChainedExtractor;
// import com.tangosol.util.extractor.KeyExtractor;
// import com.tangosol.util.extractor.ReflectionExtractor;

// 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.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);
  }

  public static void query(NamedCache cache)
  {
    // Add indexes to make queries more efficient
    // ReflectionExtractor reflectAddrHome =
    //     new ReflectionExtractor("getHomeAddress");
    // Add an index for the age
    // cache.addIndex(reflectAddrHome, true, null);
    // cache.addIndex(createExtractor("age"), 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);
  }

  // ----- sample methods ---------------------------------------------

  public static void query(NamedCache cache)
  {
    // 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);
cache.addIndex(createExtractor("homeAddress.city"), true, null);

// Find all contacts who live in Massachusetts
// Set setResults = cache.entrySet(new EqualsFilter(  
//        "getHomeAddress.getState", "MA"));
Set setResults = cache.entrySet(createFilter("homeAddress.state = '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")));
setResults = cache.entrySet(createFilter("homeAddress.state is 'MA' and  
workAddress is not 'MA'"));
printResults("MA Residents, Work Elsewhere", setResults);

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

final int nAge = 42;
Object[] aEnv  =  new Object[] {new Integer(nAge)};
// Find all contacts who are older than nAge
// setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
setResults = cache.entrySet(createFilter("age > ?1",aEnv));
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")));
setResults = cache.entrySet(createFilter("key(lastName) like 'S%' and  
homeAddress.state = 'MA'"));
setResults = cache.entrySet(createFilter("key().lastName like 'S%' and  
homeAddress.state = '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()));
System.out.println("count > " + nAge + ": " + cache.aggregate(  
        createFilter("age > ?1", aEnv), new Count()));

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

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

To rerun the query example:

2. Restart the ContactsCacheServer.
3. Run the DataGenerator, LoaderExample, and QueryExample files.
4. After printing all of the contact information in the cache, it displays the results of the queries. The results should look similar to the following examples.

Example 6–2 illustrates the output of the MA Residents filter.

Example 6–2  Output of the MA Residents Filter

...  

MA Residents
ConverterEntry(Key='John Hwdrrls', Value='John Hwdrrls
Addresses
Home: 369 Beacon St.

Ftggv, MA 24372
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, NE 84499
US
Telephone Numbers
work: +11 88 331 2307913
Example 6–3 illustrates the output of the MA Residents, Work Elsewhere filter.

**Example 6–3  Output of the MA Residents, Work Elsewhere Filter**

... MA Residents, Work Elsewhere ConverterEntry(Key="John Hwdrrls", Value="John Hwdrrls Addresses Home: 369 Beacon St.


Example 6–4 illustrates the output of the City Begins with S filter.

**Example 6–4  Output of the City Begins with S Filter**

... City Begins with S ConverterEntry(Key="John Pzek", Value="John Pzek Addresses Home: 309 Beacon St.


Example 6–5 illustrates the output of the age greater than 42 filter.

**Example 6–5  Output of the Age Greater than 42 Filter**

... Age > 42 ConverterEntry(Key="John Gddurqqziy", Value="John Gddurqqziy Addresses Home: 613 Beacon St.

Cxyskdo, DE 28968 US Work: Yoyodyne Propulsion Systems
Example 6–6 illustrates the output of the Last Name Begins with S and State Is MA filter and the output of the aggregators.

**Example 6–6  Output of the State and Age Aggregators**

```
Last Name Begins with S and State Is MA
ConverterEntry{Key='John Syaqlojl', Value='John Syaqlojl
Addresses
Home: 810 Beacon St.
Rgtaljwph, MA 07471
US
Work: 200 Newbury St.
Yoyodyne, Ltd.
Boston, MA 02116
US
Telephone Numbers
work: +11 37 18 1767648
home: +11 98 155 1073866
Birth Date: 1974-12-30'}
... count > 42: 446
  min age: 22
  avg age: 41.126
  max age: 61
```
Listening for Changes and Modifying Data

In this chapter, you set up listeners to observe data changes within a Coherence cache. You also learn how entry processors can be used to modify and process 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 act on the changes made to cache entries. It extends `java.util.EventListener` and uses the standard JavaBeans event model. All types of `NamedCache` instances implement this interface. To listen for an event, you register a `MapListener` (`com.tangosol.util.MapListener`) instance on the cache. `MapListener` instances are called on the client; that is, the listener code is executed in your client process.

There are multiple 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

These listener tasks can be performed on a `NamedCache` by the `addMapListener` methods listed in Example 7–1.

**Example 7–1  Listener Methods on a NamedCache**

```java
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 might not be present. Example 7–2 describes a pattern for registering these methods against a `NamedCache`. This has been done as an anonymous class. You can use the `getOldValue` or `getNewValue` methods in the `MapEvent` class to get the entry for which the event gets fired.

```java
Example 7–2  Pattern for Registering Methods against a NamedCache

new MapListener()
```
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. Create a Class to Listen for Changes in the Cache
2. Run the Cache Listener Example

Create a Class to Listen for Changes in the Cache

In the Loading project, create the class that will listen 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 2-11 for detailed information.

Within this class, add a listener to display a message whenever a new Contact is updated to the cache. For example, use the following code to keep the Java process running until you read from the console. Otherwise, your program will immediately exit.

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

Within the class, create an inner class to extend AbstractMapListener. Implement the methods to insert, update, and delete the cache values. In this case, most of the work should be done in the entryUpdated method, based on the old and new values contained in a MapEvent.

Example 7–3 illustrates a possible implementation of a listener class.

Example 7–3 Sample Listener Class

```java
package com.oracle.handson;

import com.tangosol.net.NamedCache;

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

import com.oracle.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 ");
      }
      System.out.println(sb.toString());
    }
  }
}
```

```
Responding to Changes in the Cache

In this section, you will 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. Entry processors (com.tangosol.util.InvocableMap.EntryProcessor) are agents that perform processing against entries. The entries are processed directly where the data is being held. The processing you perform can change the data: it can 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 I/O expense because data is not pulled to the client for processing.

Entry processors that work on 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), which invokes the passed EntryProcessor against an individual object and returns the result of the invocation.

- Map invokeAll(Collection keys, InvocableMap.EntryProcessor processor), which invokes the EntryProcessor against the collection of keys and returns the result for each invocation.
• Map `invokeAll(Filter filter, InvocableMap.EntryProcessor processor)`, which 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 class path 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` instance to change the work address of employees in the `Contacts` data set:

```java
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 can use the `invokeAll` method with the name of the `OfficeUpdater` class as one of its arguments.

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

In this exercise, you create a Java class with `EntryProcessor` instances that update entries in the cache. The `ObserverExample` class created in "Create a Class to Listen for Changes in the Cache" on page 7-2 will detect these changes and display the changed records.

1. **Create a Class to Update Entries in the Cache**
2. **Edit the POF Configuration File**
3. **Run the Cache Update Example**

**Create a Class to Update Entries in the Cache**

To create a file to update entries in the cache:

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 2-11 for detailed information.

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

   Include an inner class that implements the `PortableObject` interface (for serializing and deserializing data from the cache) and contains an `EntryProcessor` instance to set the work addresses. Use methods from the `Filter` class to isolate the `Contacts` members whose home addresses are in Massachusetts.
Example 7–5 illustrates a possible implementation of the ProcessorExample class.

Example 7–5  Sample Program to Update an Object in the Cache

```java
package com.oracle.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.handson.Address;
import com.oracle.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 -------------------------------------------
    }
```

Listening for Changes and Modifying Data 7-7
Responding to Changes in the Cache

/**
 * 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);
    System.out.println("Work address was updated for " + contact.
    getFirstName() + "'s address")
    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;
}

Edit the POF Configuration File

Edit the contacts-pof-config.xml file to add a user type ID for the OfficeUpdater entries. In this case, add the type ID 106 for the ProcessorExample$OfficeUpdater class.

...<user-type>
    <type-id>1006</type-id>
    <class-name>com.oracle.handson.
    ProcessorExample$OfficeUpdater</class-name>
</user-type>
...
Run the Cache Update Example

To run the cache update example.

1. Create a run configuration for ProcessorExample. Right click ObserverExample in the Project Explorer and select Run As. In the Run Configurations dialog box select Oracle Coherence and click the New Configuration icon
   - In the Name field, enter ProcessorExample.
   - In the Project field in the Main tab, enter Loading. In the Main class field, enter com.oracle.handson.ProcessorExample.
   - In the General tab of the Coherence tab, browse to the c:\home\oracle\workspace\Contacts\appClientModule\coherence-cache-config.xml file in the Cache configuration descriptor field. Select the Disabled (cache client) button. Enter 3155 in the Cluster port field. Click Apply.
     In the Other tab, scroll down to the tangosol.pof.config field. Enter the absolute path to the POF configuration file contacts-pof-config.xml. Click Apply.
   - In the Common tab, select Shared file and browse for the Loading directory.

2. Perform the following steps to test the ObserverExample and ProcessorExample classes.
   b. Restart the ContactsCacheServer.
   c. Run the LoaderExample class to load the cache.
   d. Run the ObserverExample class.
   e. Run the ProcessorExample to update records in the cache.
      You should see messages in the cache server console window, that are similar to Example 7–6, indicating that the work addresses for the specified employees were updated.

Example 7–6  Output from the ObserverExample and ProcessorExample Classes

...  
Group{Address=224.3.7.0, Port=3155, TTL=4}

MasterMemberSet
   {  
      ThisMember=Member(Id=3, Timestamp=2011-03-15 12:20:17.538, Address=130.35.99.213:8090,  
      MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4396,  
      Role=OracleHandsonObserverExample)  
      OldestMember=Member(Id=1, Timestamp=2011-03-15 12:17:47.491, Address=130.35.99.213:8088,  
      MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:1980,  
      Role=CoherenceServer)  
      ActualMemberSet=MemberSet(Size=2, BitSetCount=2  
         Member(Id=1, Timestamp=2011-03-15 12:17:47.491, Address=130.35.99.213:8088, MachineId=49877,  
         Member(Id=3, Timestamp=2011-03-15 12:20:17.538, Address=130.35.99.213:8090, MachineId=49877,  
         Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4396, Role=OracleHandsonObserverExample)  
      )  
      RecycleMillis=1200000  
      RecycleSet=MemberSet(Size=0, BitSetCount=0}
TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}

2011-03-15 12:20:17.772/1.313 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=3): Service Management joined the cluster with senior service member 1
2011-03-15 12:20:17.928/1.469 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache:PartitionedPofCache, member=3): Service PartitionedPofCache joined the cluster with senior service member 1
2011-03-15 12:20:33.913/17.454 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=3): Member 4 joined Service Management with senior member 1
2011-03-15 12:20:34.084/17.625 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=3): Member 4 joined Service PartitionedPofCache with senior member 1
Work address was updated for John Yuoo
Work address was updated for John Dikkx
Work address was updated for John Kwtkvp
Work address was updated for John Tdpynycf
Work address was updated for John Bojhtnxxig
Work address was updated for John Sgyichnpcj
Work address was updated for John Vjfdqs
Work address was updated for John Cudpahnguc
Work address was updated for John Wovzeja
Work address was updated for John Woyy
Work address was updated for John Peladt
Work address was updated for John Nuetsjrd
Work address was updated for John Oueywut
Work address was updated for John Unwypjxz
Work address was updated for John Xfazifsx
Work address was updated for John Qnmd
Work address was updated for John Sgiephelfq
Work address was updated for John Gajpavab
2011-03-15 12:20:34.272/17.813 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=3): Member 4 left service Management with senior member 1
2011-03-15 12:20:34.272/17.813 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=3): Member 4 left service PartitionedPofCache with senior member 1
2011-03-15 12:20:34.272/17.813 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=3): Member 4 left Cluster with senior member 1
In this chapter, 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

You must have a working version of the Oracle Database Express Edition (also known as the OracleXE database) installed on your system. If you do not have the database, you can download it from this URL:


Introduction

A major enhancement in Enterprise JavaBeans (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 earlier EJB technologies.

JPA defines how 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.

The JPA implementation used in this exercise provides Object Relational Mapping (ORM) from the Java world to the database world. It also allows you to use standard Coherence get and put methods, and have 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, you use Eclipse to perform the following tasks:

- Create a connection to the HR schema in the OracleXE database.
- Automatically generate JPA objects for the EMPLOYEES table.
- Modify the cache-server.cmd file to point to the sample JPA cache-config.xml file.
1. Unlock the OracleXE Database
2. Download the JPA Libraries
3. Configure the Project for JPA
4. Create the JPA Persistence Unit and Entity Beans
5. Edit the persistence.xml File
6. Create the Cache Configuration File for JPA
7. Create a Cache Server Start-Up Configuration
8. Create a Class to Interact with the Data Object
9. Create a Run Configuration for RunEmployeeExample
10. Run the JPA Example

Unlock the OracleXE Database

Unlock the HR account in your pre-installed OracleXE database.

You must have the OracleXE database installed on your system and be able to access the HR schema. To unlock the HR account, follow these steps:

1. Navigate to Start then All Programs then Oracle Database Express Edition then Run SQL Command Line.

2. Enter connect system as sysdba, and then enter welcome1 when prompted for the password. (Note: your user name is system and your password is welcome1.)

3. Enter the command to unlock the account:
   
   ```
   alter user hr identified by hr account unlock;
   ```

---

Figure 8–1  Connecting to the Database

---

Figure 8–2  Unlocking the Database Account

---
Download the JPA Libraries

This exercise requires the org.eclipse.persistence.* and the javax.persistence files to be present in your system. You can obtain these files through the Eclipse IDE.

1. Select Help then Eclipse Marketplace.
2. In the Eclipse Marketplace dialog box, enter EclipseLink in the Find field and click Go.
3. Select the EclipseLink SDK, as illustrated in Figure 8–3, and follow installation instructions. The required persistence files will be installed in the eclipse\plugins directory.

Figure 8–3 EclipseLink SDK in the Eclipse Marketplace

Configure the Project for JPA

Create a new project in Eclipse IDE called JPA. See "Creating a New Project in the Eclipse IDE" on page 2-3 for detailed information.

1. In the Eclipse IDE, click Window then Open Perspective then Other to open the Open Perspective dialog box. Select the JPA perspective, as illustrated in Figure 8–4.
2. Select File then New then JPA Project. Enter JPA as the project name. Ensure that the default location points to home/oracle/workspace/JPA. Click Modify in the Configuration field to open the Project Facets dialog box.

3. Select the Oracle Coherence facet. The JPA facet should already be selected.

4. Click Save As in the Configuration field. Enter JPAConfiguration in the Save Preset dialog box and click OK. The contents of the Project Facets dialog box should look similar to Figure 8–5.

Figure 8–5  Project Facets for a JPA Project
The contents of the **New JPA Project** dialog box should look similar to Figure 8–6. Click **Next** to go to the **Java** page.

**Figure 8–6 Contents of the New JPA Project Dialog Box**

5. Click **Next** in the **Java** page to accept the default output folder location.

6. The **Coherence** user library should already be selected in the **Coherence** page. Click **Next**.

7. In the **JPA Facet** page, select **EclipseLink 2.1.x** in the **Platform** drop-down list. Click the Download library icon to download the EclipseLink files. In the Download Library dialog box, select **EclipseLink 2.1.2 - Helios**, as illustrated by Figure 8–7.
8. Click **Next** to accept the license agreement, then click **Finish**. After downloading the library, the JPA Facet page should look similar to
9. Create a connection to the Oracle XE Database. Click **Add connection**.
   a. In **Connection Profile** dialog box, select **Oracle Database Connection** and enter **XE_HR** in the **Name** field, as illustrated in **Figure 8–9**. Click **Next**.
b. Select Oracle Database 10g Driver Default in the Drivers field. Edit the General tab. Enter hr in the User name field and hr in the Password field. Select the Save Password check box. Enter localhost in the Host field. Select the Connect when the wizard completes check box. Click Test Connection button. The test should return an alert box with a success message, as illustrated in Figure 8–10. Click OK, then Next.
c. In the JPA Facet page, click Add driver library to build path, then click Finish.

Create the JPA Persistence Unit and Entity Beans

A persistence unit provides an easy way to identify the set of metadata files, classes, and JARs that contain all classes that need to be persisted as a group. The name of the persistence unit is used to identify it. Entity beans are Enterprise Java beans that contain persistent data, and that can be saved in persistent data stores. The JPA entity bean can belong to a persistence unit. The persistence unit is described by the persistence.xml file.

To create the JPA persistence unit and the JPA entity beans:

1. Right-click the JPA project and select New then select Entities from Tables. The Select Tables dialog box opens.
2. Select the EMPLOYEES table, as illustrated in Figure 8–11 and click Next.
3. In the **Table Associations** page, illustrated in Figure 8–12, click **Next** to accept the defaults.
4. In the Customize Default Entity Generation page, select **Collection properties** type `java.util.List`, Enter `com.oracle.handson` in the **Package** field.

When you are finished, the **Customize Default Entity Generation** page should look similar to Figure 8–13. Click **Next**.
5. In the **Customize Individual Entities** page, accept the defaults as illustrated in and click **Finish** to generate the JPA Entities.
**Edit the persistence.xml File**

The *persistence.xml* file defines one or more persistence units. Use the *persistence.xml Editor* tool in the Eclipse IDE to edit the *persistence.xml* file.

1. Open the *persistence.xml* file in the Eclipse editor.
2. Click the **General** tab. Enter the name of the persistence provider. This will be `org.eclipse.persistence.jpa.PersistenceProvider`. 

---

**Figure 8–14  Customize Individual Entities Page**
3. In the Connection tab, select Resource Local from the Transaction type drop down list.

In JDBC connection properties section, click Populate from connection link. The values in this field will be provided from the database connection you defined in “Configure the Project for JPA” on page 8-3. The Connection Properties tab should look similar to Figure 8–16.
Figure 8–16  Connection Properties Tab in the persistence.xml Editor

4. Inspect the Properties tab. The contents should look similar to Figure 8–17.
Figure 8–17  Properties Tab in the persistence.xml Editor

5. Open the Source tab. The persistence.xml file should look similar to Example 8–1.

Example 8–1  Generated persistence.xml File

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence_2_0.xsd">
  <persistence-unit name="JPA" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>com.oracle.handson.Employees</class>
    <properties>
      <property name="eclipselink.target-server" value="None"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:oracle:thin:@localhost:1521:xe"/>
      <property name="javax.persistence.jdbc.user" value="hr"/>
      <property name="javax.persistence.jdbc.password" value="hr"/>
      <property name="javax.persistence.jdbc.driver" value="oracle.jdbc.OracleDriver"/>
    </properties>
  </persistence-unit>
</persistence>
```

Create the Cache Configuration File for JPA

Create a cache configuration file for the JPA project.

1. Right click coherence-cache-config.xml file and select Open.

2. Enter the code illustrated in Example 8–2. Use Save As to save the file as jpa-cache-config.xml. The file will be saved to the home\oracle\workspace\JPA\src folder.

Example 8–2  Cache Configuration for JPA

```xml
<?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>
</cache-config>
```
3. Delete the original coherence-cache-config.xml file from the Project Explorer.

### Create a Cache Server Start-Up Configuration

Create a configuration to start the cache server for the JPA project.

1. Right click the project and select Run As then Run Configurations. In the Run Configurations dialog box, Enter JPACacheServer in the Name field.

2. In the Main tab, click Browse in the Project field and select the JPA project in the Project Selection dialog box. Select the Include system libraries when searching for a main class checkbox and click Search. Enter DefaultCacheServer in the
Select Type field and select `com.tangosol.net.DefaultCacheServer`. Click OK. The Main tab should look similar to Figure 8–18.

**Figure 8–18 Main Tab for the JPA Cache Server**

![Main Tab for the JPA Cache Server](image)

3. In the General tab of the Coherence tab, identify the path to the cache configuration file under Topology. Click the Browse button to navigate to the Absolute file path of the JPA cache configuration file `C:\home\oracle\workspace\JPA\src\jpa-cache-config.xml`. Select Enabled (cache server) under Local storage. Enter a unique value, such as 3155, for the Cluster port.

   In the Other tab, ensure that the `tangosol.pof.config` item is set to the default `pof-config.xml`.

4. In the Common tab, select Shared file and browse to the `\JPA` project.

5. In the Classpath tab, click User Entries then Add External JARs to add the `coherence-jpa.jar` file to the list. Click Apply.

   The Classpath tab should look similar to Figure 8–19.

**Figure 8–19 Classpath Tab for the JPA Cache Server Executable**

![Classpath Tab for the JPA Cache Server Executable](image)

Create a Class to Interact with the Data Object

Create a new class in the JPA project to interact with the Employees object. The objective of the class will be to change the value of an employee’s salary.

1. Create a new class with a main method called `RunEmployeeExample`. See "Creating a Java Class" on page 2-11 for detailed information.

2. Create the code to perform the following:
a. Get an employee using the EMPLOYEE_ID attribute. EMPLOYEE_ID is a long data type.

b. Display the salary.

c. Give the employee a 10% pay raise.

d. Get the value again to confirm the pay raise.

Example 8–3 illustrates a possible implementation of the RunEmployeeExample class.

Example 8–3  Sample Employee Class File

```java
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import java.math.BigDecimal;

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().multiply(BigDecimal.valueOf(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() );
    }
}
```

Create a Run Configuration for RunEmployeeExample

To create a run configuration, modify the run-time properties and edit the class path in Eclipse.

1. Right click RunEmployeeExample.java in the Project Explorer and choose Run As then Run Configurations. In the Run Configurations dialog box, click New launch configurations icon. Enter RunEmployeeExample in the Name field.

2. In the Main tab, click Browse and select the JPA project. In the Main class field, click Search and choose com.oracle.handson.RunEmployeeExample. Click Apply.
3. In the General tab of the Coherence tab, browse to the C:\home\oracle\workspace\JPA\src\jpa-cache-config.xml file in the Cache configuration descriptor field. Select the Disabled (cache client) button. Enter 3155 in the Cluster port field. Click Apply.

4. In the Other tab, ensure that the default pof-config.xml appears in the tangosol.pof.config field.

5. In the Classpath tab, click Add External JARs to add the coherence-jpa.jar file to User Entries.

6. Use the Up and Down buttons to move Coherence37 to the top of Bootstrap Entries. Click Apply.

The Classpath tab should look similar to Figure 8–20.

Figure 8–20  Classpath Tab for the RunEmployeeExample Program

Run the JPA Example

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 in the database. When you use the get(Object key) method, the following happens:

- Coherence looks for the entry with the key.
- If the entry is not already in the cache, or if it is expired from the cache, then coherence calls 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 the put(object key, object value) method, coherence uses jpa through eclipselink to persist any changes to the database.

Click Run to run the RunEmployeeExample configuration. The output from the program should look similar to Example 8–4.

Example 8–4  Output from the RunEmployeeExample Program

```java
TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}
```
Employee Timothy Gates, salary = $17736.19
New Employee details are Timothy Gates, salary = $19509.809

The response from the cache server displays a successful login to retrieve information from the database, as illustrated in Example 8–5.

Example 8–5 Cache Server Response to Logging In to the Database

2011-03-15 19:28:54.573/5.203 Oracle Coherence GE 3.7.0.0 <D5> (thread=Invocation:Management, member=1): Service Management joined the cluster with senior service member 1
2011-03-15 19:28:54.839/5.469 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache:JpaDistributedCache, member=1): Service JpaDistributedCache joined the cluster with senior service member 1
2011-03-15 19:28:54.886/5.516 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=1): Services
	(ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.7, OldestMemberId=1)
		InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
		PartitionedCache{Name=JpaDistributedCache, State=(SERVICE_STARTED), LocalStorage=enabled, PartitionCount=257, BackupCount=1, AssignedPartitions=257, BackupPartitions=0}
)}

Started DefaultCacheServer...

2011-03-15 19:29:10.307/20.937 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 joined Service Management with senior member 1
2011-03-15 19:29:10.417/21.047 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 joined Service JpaDistributedCache with senior member 1


2011-03-15 19:29:15.073/25.703 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 left service Management with senior member 1
2011-03-15 19:29:15.073/25.703 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1): Member 2 left service JpaDistributedCache with senior member 1
Interacting with the Cache and the Database

In this chapter, you will create and configure an Oracle Coherence cache in Eclipse. In this exercise you will create these items:

- A Java class that creates a `NamedCache` instance and can put and get cache entries.
- A cache configuration file to define the mapping for cache names, cache types, and naming patterns.
- A Java class that creates a connection to Oracle Database and can retrieve and store table data.
- A database cache. This class will add cache entries, query the database cache, and retrieve entries.

This chapter contains the following sections:

- Introduction
- Creating a Cache Application
- Creating a Database Cache

Introduction

A Coherence cache is a collection of data objects that acts as an intermediary between the database and the client applications. Database data can 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 can 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 can be performed on the data objects.

Oracle Coherence also provides event-based processing. The state of data objects in a cache can 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, where data is replicated to each of the application server nodes in the cluster. This type of cache is recommended if faster read access is required but not suitable for write operations, because data must be written to each of the nodes. The drawback of replicated caches is that they require a large amount of memory because every node has a copy of every object.

Distributed (or partitioned) caches, where 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.

A 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. Table 9–1 describes some of the cache types commonly used by Coherence.

<table>
<thead>
<tr>
<th>Cache Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>distributed scheme</td>
<td>Defines a distributed cache in which data is stored across a cluster of nodes.</td>
</tr>
<tr>
<td>replicated scheme</td>
<td>Defines a cache in which cache entries are replicated across all the cluster nodes.</td>
</tr>
<tr>
<td>read-write-backing-map scheme</td>
<td>Defines a map, which provides a cache of a persistent store such as a relational database.</td>
</tr>
<tr>
<td>external scheme</td>
<td>Defines an external cache such as a disk.</td>
</tr>
<tr>
<td>class scheme</td>
<td>Defines a custom cache implementation, which is required to implement the java.util.Map interface.</td>
</tr>
</tbody>
</table>

Creating a Cache Application

This section describes how to create and run an application that puts data into the cache and retrieves it.

1. Create an Application that Constructs a Cache
2. Create a Cache Configuration File
3. Configure the Project Properties
4. Create the Cache Server Start-Up Configuration
5. Run the Cache Creation Application

Create an Application that Constructs a Cache

To create a Java class that constructs a Coherence cache:

1. Create a project in Eclipse.
   a. Use the JPA perspective in Eclipse to create a JPA Project called Interact. Select the JPA Configuration you created in the previous chapter.
   b. In the Coherence page, select Coherence37 and EclipseLink2.1.2 - Helios.
c. In the JPA Facet page, ensure that EclipseLink 2.1.x appears in the Platform field. In the Connection field, select the connection you created in "Configure the Project for JPA" on page 8-3 (XE_HR), from the Connection drop down list. Click the Connect link to connect to the Oracle XE Database. Click Finish. (Note, this assumes that the database connection that you created is still running.)

2. Create a Java class, CoherenceCache, that will be used to create a Coherence cache. Include a main method in the class. See "Creating a Java Class" on page 2-11 for detailed information on creating a class.

   a. Create a cache in the CoherenceCache Java class. Import the CacheFactory class and the NamedCache interface.

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

   b. Create the cache (NamedCache) instance by using the CacheFactory. getCache method. Use the cache name VirtualCache, which is mapped to a distributed caching scheme.

      ```java
      NamedCache cache = CacheFactory.getCache ("VirtualCache");
      ```

   c. A NamedCache is a java.util.Map instance that holds resources that are shared across nodes in a cluster. Add a cache entry by using the put method.

      ```java
      cache.put (key, "Hello Cache");
      ```

   d. Retrieve a cache entry by using the get method.

      ```java
      System.out.println((String)cache.get("hello"));
      ```

      Example 9–1 illustrates a possible implementation of CoherenceCache class. You can copy the code to the CoherenceCache application in Eclipse.

### Example 9–1 Implementation of a Coherence Cache

```java
package com.oracle.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)
```
Create a Cache Configuration File

Create an XML document, cache-config.xml, as the cache configuration deployment descriptor. Save the file to the C:\home\oracle\workspace folder.

In the cache configuration file:

- Define mappings for cache names and naming patterns with the cache-mapping elements in the caching-scheme-mapping element.
- Map the 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 illustrated in Example 9–2. Copy the contents of this example to the cache-config.xml.

Example 9–2 Cache Configuration File

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>VirtualCache</cache-name>
      <scheme-name>default-distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <!--
    Default Distributed caching scheme.
    -->
    <distributed-scheme>
      <scheme-name>default-distributed</scheme-name>
      <service-name>DistributedCache</service-name>
      <backing-map-scheme>
        <class-scheme>
          <scheme-ref>default-backing-map</scheme-ref>
        </class-scheme>
      </backing-map-scheme>
    </distributed-scheme>
    <class-scheme>
      <scheme-name>default-backing-map</scheme-name>
      <class-name>com.tangosol.util.SafeHashMap</class-name>
    </class-scheme>
    <autostart>true</autostart>
  </caching-schemes>
</cache-config>
```
Configure the Project Properties

Create a run configuration for the application to add the cache configuration file as a run-time Java option.

1. Right click `CoherenceCache.java` in the **Project Explorer** and choose **Run As** then **Run Configurations**. In the **Run Configurations** dialog box, click the **New launch configuration** icon.

2. Enter `CoherenceCache` in the **Name** field. Ensure that `Interact` is in the **Project** field and `com.oracle.handson.CoherenceCache` is in the **Main class** field.

3. In the **Coherence** tab, enter the path to the cache configuration file, `C:\home\oracle\workspace\cache-config.xml`. Select the **Enabled (cache server)** button. Enter a unique value, such as `3155`, in the **Cluster port** field. Click **Apply**.

4. In the **Classpath** tab, Use the **Add External JARs** button to add the `coherence.jar` file to **User Entries**. When you are finished, the **Classpath** tab should look similar to **Figure 9–1**.

![Figure 9–1 Classpath for the CoherenceCache Program](image)

Create the Cache Server Start-Up Configuration

To create a cache server start-up configuration for the Interact project:

1. Right click the **Interact** project and select **Properties**. In the **Properties for Interact** dialog box, select **Java Build Path**. In the **Order and Export** tab, the Interact project and the JRE, EAR, Coherence37, EclipseLink 2.1.2 - Helios, and the Oracle Database 10g Driver Default libraries should be present. Click **Select All**. The **Order and Export** tab should look similar to **Figure 9–2**.

![Figure 9–2 Order and Export Tab for the Interact Project](image)
Creating a Cache Application

2. Edit the JPA cache server start-up configuration (JPACacheServer) that you created in Chapter 8, "Using JPA with Coherence".

3. Right click the Interact project and select Run As then Run Configurations. In the Main tab click Browse and select the Interact project from the Project Selection dialog box.

4. In the General tab of the Coherence tab, replace the name and path of the configuration file with C:\home\oracle\workspace\cache-config.xml.

5. In the Classpath tab, remove the JPA (default classpath) folder. Click Add Project to add the Interact project. The Classpath tab should look similar to Figure 9–3.

6. In the Shared file field of the Common tab, click Browse to select the Interact project. Click Apply, then Close.

Run the Cache Creation Application

To run the cache creation application CoherenceCache.java:


2. Run the jpa-cache-server.cmd file to start the cache server.
3. Right-click the Oracle Coherence application `CoherenceCache.java` and click **Run As** then **Run Configurations**. Select the `CoherenceCache` configuration and click **Run**. The Eclipse console window displays the output:

- The operational configuration is loaded from the `tangosol-coherence.xml` file. This file specifies the operational and run-time settings used by Coherence for its clustering, communication, and data management services.
- The cache configuration is loaded from the `cache-config.xml` file.
- A new cluster is created and the `DistributedCache` service joins the cluster.
- The output of the `CoherenceCache.java` program, **Hello Cache** is displayed.

**Example 9–3  Output of the Coherence Cache Application**

```
2011-03-15 13:03:17.202/0.313 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2011-03-15 13:03:17.249/0.360 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2011-03-15 13:03:17.280/0.391 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "file:/C:/home/oracle/workspace/Interact/build/classes/tangosol-coherence-override.xml"
2011-03-15 13:03:17.280/0.391 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-15 13:03:17.436/0.547 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from "file:/C:/home/oracle/workspace/cache-config.xml"
2011-03-15 13:03:17.624/0.735 Oracle Coherence GE 3.7.0.0 <D4> (thread=main, member=n/a): TCMP bound to /130.35.99.213:8088 using SystemSocketProvider
2011-03-15 13:03:21.217/4.328 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Started cluster Name=cluster:0x96AB
Group(Address=224.3.7.0, Port=3155, TTL=4)

MasterMemberSet
{
    ThisMember=Member(Id=1, Timestamp=2011-03-15 13:03:17.639, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeff1-lap7,process:2084, Role=OracleHandsonCoherenceCache)
    OldestMember=Member(Id=1, Timestamp=2011-03-15 13:03:17.639, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeff1-lap7,process:2084, Role=OracleHandsonCoherenceCache)
    ActualMemberSet=MemberSet(Size=1, BitSetCount=2
        Member(Id=1, Timestamp=2011-03-15 13:03:17.639, Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeff1-lap7,process:2084, Role=OracleHandsonCoherenceCache)
    )
RecycleMillis=1200000
RecycleSet=MemberSet(Size=0, BitSetCount=0
```
Creating a Database Cache

In this section, create a cache backed by Oracle Database. This is also referred to as an Oracle Database cache.

1. **Create an Oracle Database Cache**
2. **Create a Class to Define a Custom Cache Store**
3. **Modify the Cache Configuration File**
4. **Create a Class to Construct the Database Cache**
5. **Run the Database Cache Application**

### Create an Oracle Database Cache

To use SQL*Plus and the Oracle Database Express Edition (Oracle XE) to create an Oracle Database cache follow these steps. You must have the database installed on your system.

1. **Invoke SQL*Plus.**
   
   Navigate to **Start** then **All Programs** then **Oracle Database Express Edition** and then **Run SQL Command Line**.

2. **Connect as hr user with hr as the password.**
   
   connect hr/hr;

3. **Create an Oracle Database table.**
   
   Open a text editor and copy the following SQL code. Save it as a file named `dbscript.sql` in the `/home/oracle/workspace/` folder.

   **Example 9–4 SQL Script for Creating a Database Table**

   ```sql
   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');
   ```

4. **Run the SQL script.**

   **Example 9–5** illustrates the output from the script.

   **Example 9–5 Running the SQL Script for Creating a Database Table**

   SQL*Plus: Release 10.2.0.1.0 - Production on Tue March 1 13:38:59 2011
   Copyright (c) 1982, 2005, Oracle. All rights reserved.
   
   SQL> connect hr/hr;
Connected.
SQL> @/home/oracle/workspace/dbscript.sql
Table created
1 row created
1 row created

Create a Class to Define a Custom Cache Store

To create a Java class that connects to the database and retrieves table data:

1. Create a Java class DBCacheStore in Eclipse. See "Creating a Java Class" on page 2-11 for detailed information.

2. Create the code to connect to the database and get table data.

   Example 9–6 illustrates a possible implementation of the DBCacheStore class.
   Copy the code to the DBCacheStore application in the Eclipse IDE. The DBCacheStore application uses Java Database Connectivity (JDBC) to access Oracle Database, but you could use another mechanism, such as Hibernate or Java Data Objects (JDO), instead.

   Example 9–6   Database Cache Store Implementation

   ```java
   package com.oracle.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 = sTabletName;
       }
   }
   ```
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();

Modify the Cache Configuration File

Modify the cache configuration file (cache-config.xml) that you created earlier for the database cache. To connect a cache to a database, you must configure the cachestore-scheme element with a custom class that implements either the com.tangosol.net.cache.CacheLoader or com.tangosol.net.cache.CacheStore interface.

Replace the code in the existing cache-config.xml file in Eclipse with the cache configuration code for the database cache in Example 9–7.

Example 9–7 Database Cache Configuration File

```xml
<?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>
```
In the cache configuration file, you have done the following:

- Defined a cache name pattern `DBBacked*`, which is mapped to a distributed caching scheme `distributed-db-backed`.
- Specified the `CacheStore` scheme in the distributed scheme using the class `coherence.DBCacheStore`, which implements the `CacheStore` interface.
- Specified for the `DBCacheStore` class an `init` parameter for the database table that is at the back end of the cache. The table name is specified in the `init-param` element. The `DBCacheStore` class performs database operations such as reading and writing cache entries.
- Specified a `read-write-backing-map-scheme` as the backing map. This scheme defines a backing map, which provides a size-limited cache of a persistent store. Here, by setting the `write-delay-seconds` parameter to 0, you specify the write-through mechanism.

Table 9–2 describes the types of read/write caching that you can use with Oracle Coherence.

<table>
<thead>
<tr>
<th>Types of Read/Write Caching</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>read-through</td>
<td>A cache entry is read into a cache from the database when required and made available to an application.</td>
</tr>
<tr>
<td>write-through</td>
<td>Updates to cache entries are synchronized with the database without a delay.</td>
</tr>
<tr>
<td>refresh-ahead</td>
<td>Cache entries are refreshed periodically.</td>
</tr>
</tbody>
</table>
Create a Class to Construct the Database Cache

Create a Java class `DatabaseCache` for the database cache in Eclipse. The class must contain a `main` method. See "Creating a Java Class" on page 2-11 for detailed information.

In the class file, provide the 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()`. Example 9–8 illustrates a possible implementation.

Example 9–8 Implementation for the Database Cache Class File

```java
package com.oracle.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");
    }

    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"));
    }

    public void retrieveEntry() {
        System.out.println((String) cache.get("catalog3"));
    }

    public void eraseEntry() {
        cache.remove(new String("catalog3"));
    }
}
```

Table 9–2 (Cont.) Types of Read/Write Caching Supported by Coherence

<table>
<thead>
<tr>
<th>Types of Read/Write Caching</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>write-behind</td>
<td>Updates to cache entries are asynchronously written to a database after a delay specified in the <code>write-delay-seconds</code> element in the cache configuration file.</td>
</tr>
</tbody>
</table>

Create a Database Cache

9-14 Oracle Coherence Tutorial for Oracle Coherence
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);
    Set results = cache.entrySet(filter);

    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.queryCache();
}

Note the following features of the database cache class file:

- A NamedCache object is created using the getCache method of the CacheFactory class in the createCache method.

```java
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.

```java
cache.put(new String("catalog3"), new String("Tuning Grid Management"));
```

- Because the write-through mechanism is used, the new cache entry 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.

```java
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`:
  ```java
  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 can 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 can be performed. Therefore, you can retrieve database data and create a collection of cache entries using the `getAll()` method.
  ```java
  HashSet hashSet=new HashSet();
  hashSet.add(new String("catalog1"));
  hashSet.add(new String("catalog2"));
  hashSet.add(new String("catalog3"));
  Map map=cache.getAll(hashSet);
  ```
- Create a `LikeFilter` filter to search for cache entries starting with `Tuning`.
  ```java
  Filter filter = new LikeFilter(IdentityExtractor.INSTANCE, "Tuning%", '\\', true);
  ```
- Query the database cache using the `entrySet()` method with the `LikeFilter` filter.
  ```java
  Set results = cache.entrySet(filter);
  ```
- Iterate over the results of the query. Display the key and value of the cache entries that are retrieved.
  ```java
  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.
  ```java
  ContinuousQueryCache queryCache = new ContinuousQueryCache(cache, filter );
  ```
- Create a result set by using the `entrySet()` method.
  ```java
  Set results = queryCache.entrySet(filter);
  ```
Run the Database Cache Application

To run the Oracle Database cache application:

1. Stop any running cache servers. See “Stopping Cache Servers” on page 2-13 for more information.

2. Start the cache server (JPACacheServer). Right click the project and select Run As then Run Configurations. In the Run Configurations dialog box, select JPACacheServer to display its configuration. Click Run.

3. Create a run configuration for the DatabaseCache program.
   a. In the Run Configurations dialog box, click the New launch configuration icon.
   b. Enter DatabaseCache in the Name field. Ensure that Interact is in the Project field and com.oracle.handson.DatabaseCache is in the Main class field.
   c. In the Coherence tab, enter the path to the cache configuration file in the Cache configuration descriptor field: C:\home\oracle\workspace\cache-config.xml. Select Local storage: Disabled (cache client). Enter a unique value, such as 3155, in the Cluster port field.
   d. In the Classpath tab, use the Add External JARs button to add the coherence.jar file to User Entries. The Classpath tab should look similar to Figure 9–4.

   Figure 9–4 Classpath for the DatabaseCache Program

   ![](classpath.png)

   e. Click Apply then Run.

Example 9–9 illustrates the expected results.

Example 9–9 Output of the DatabaseCache Program

```
2011-03-15 13:47:59.374/0.329 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml'
```
Creating a Database Cache

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

The log output shown here is from an Oracle Coherence instance that has been started.

It is important to understand the operation of a database cache. In this section, we will focus on the steps required to create a database cache.

1. **Start Coherence**: First, you need to start the Oracle Coherence service. This can be done by executing the following command:

   ```bash
   /path/to/coherence/start_coherence.sh
   ```

2. **Configure the Cache**: Next, you need to configure the cache. The cache configuration file is typically located in the `$COHERENCE_HOME` directory.

   ```bash
   /path/to/coherence/coherence/home/config/cache-config.xml
   ```

3. **Start a Database Cache**: Once the cache is configured, you can start a database cache by executing the following command:

   ```bash
   /path/to/coherence/start_database_cache.sh
   ```

4. **Manage the Cache**: After the cache is started, you can manage it using the Coherence Management Interface (CMI). The CMI allows you to view and modify the cache configuration and state.

5. **Monitor the Cache**: It is important to monitor the cache to ensure it is performing as expected. This can be done using the Coherence Monitor (CM). The CM provides real-time monitoring information about the cache.

The steps for creating a database cache are as follows:

1. Start Coherence
2. Configure the Cache
3. Start a Database Cache
4. Manage the Cache
5. Monitor the Cache

---

9-18 Oracle Coherence Tutorial for Oracle Coherence
Creating a Database Cache

Interacting with the Cache and the Database

9-19

Example 9–10 illustrates the cache server response to the DatabaseCache program.

**Example 9–10 Cache Server Response to the DatabaseCache Program**

... Started DefaultCacheServer...

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 `*`. Re-run the `DatabaseCache` application in Eclipse. You should not see any exceptions.

4. Note that because you are using a write-through cache, the database table is also updated. From the SQL prompt, enter the following code:

```sql
select * from hr.catalog;
```

Example 9–11 illustrates the results.

**Example 9–11 Output from the SELECT Statement**

```
<table>
<thead>
<tr>
<th>ID</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog3</td>
<td>Tuning Grid Management</td>
</tr>
</tbody>
</table>
```
catalog4
Tuning Coherence

catalog5
Tuning Database

ID
-------------------------
VALUE
-------------------------------------------------------------
catalog1
Tuning Undo Tablespace

catalog2
Tuning Your View Objects
This chapter describes how to apply security to a Coherence*Extend client. Coherence*Extend allows a wide range of access to Coherence caches. These include desktop applications, remote servers, and machines located across wide area network (WAN) connections.

This chapter contains the following sections:

- **Introduction**
- **Enabling Token-Based Security**
- **Including Role-Based Access Control to the Cluster**
- **Including Role-Based Access Control to an Invocable Object**

**Introduction**

Coherence*Extend consists of an extend client running outside the cluster and a proxy service running inside the cluster, hosted by one or more cache servers. The client APIs route all requests to the proxy. The proxy responds to client requests by delegating to Coherence clustered services, such as a partitioned or replicated cache service or an invocation service.

Because the extend client exists outside of the cluster, the issue of securing access to the cluster takes on greater importance. This chapter describes three techniques that you can use to secure access between the client and the cluster. The techniques include using identity token-based passwords, an entitled cache service, and an invocation service.

A detailed discussion of these security techniques and extend clients is beyond the scope of this tutorial. For more information on these topics, see the Oracle Coherence Security Guide.

**Enabling Token-Based Security**

You can implement token-based security to enable access between an extend client and an extend proxy in the cluster. To enable access between the extend client and an extend proxy the following files are typically required:

- Client application files, which describe the functionality that will access the cluster
- Cache configuration files, where the extend client and the extend proxy each have their own cache configuration
- Operational override file, which overrides the operational and run-time settings in the default operational deployment descriptor
Server start-up files, where there is a start-up file for the extend proxy and for the cache server in the cluster

POF configuration deployment descriptor, which specifies custom data types when using POF to serialize objects

To add token-based security, you must also provide identity transformer and asserter implementations. The transformer generates the token on the client side and the asserter validates it on the cluster side.

The following steps describe how to create and run an application for an extend client that uses token-based security to access the cluster.

1. **Use a Security Helper File**
2. **Create an Identity Transformer**
3. **Create an Identity Asserter**
4. **Create the Password File**
5. **Enable the Identity Transformer and Asserter**
6. **Create a Cache Configuration File for the Extend Client**
7. **Create a Cache Configuration File for the Extend Proxy**
8. **Create a Start-Up Configuration for a Cache Server with a Proxy Service**
9. **Create a Start-Up Configuration for a Cache Server**
10. **Run the Password Example**

### Use a Security Helper File

The examples in this chapter reference a security helper file that defines role-based security policies and access control to the cache. For the purpose of these examples, a file with simplified mappings is provided for you.

Cache access is determined by a user’s role. The security helper file defines several roles: `role_reader`, `role_writer`, and `role_admin`. It defines the mappings of various users to the roles, such as `BuckarooBanzai` to `ROLE_ADMIN`. It defines the mappings of roles to integer IDs, such as `ROLE_ADMIN` to `9`. The helper file also defines the cache name and the invocation service name used in the examples.

The key features of this file are the `login` and `checkAccess` methods. The `login` method takes a user name and constructs a simplified distinguished name (DN). It then associates a role with the name. `PofPrincipal` provides the `Principal` implementation.

The `checkAccess` method shows where the authorization code is placed. It determines whether the user can access the cache based on a provided user role.

To create a new project and the security helper file:

1. In Eclipse IDE, select the Java EE perspective and create a new Application Client Project called `Security`. Select `CoherenceConfig` from the `Configuration` dropdown list. Ensure that the `Create a default main` is not selected on the Application Client module page.

   In the Coherence page, select *only* `Coherence37`.

   See "Creating a New Project in the Eclipse IDE" on page 2-3 for detailed information on creating a project.
2. Create a new Java file named `SecurityExampleHelper.java`. Ensure that the package path is `com.oracle.handson`.

   See "Creating a Java Class" on page 2-11 for detailed information on creating a Java class.

3. Copy the code illustrated in *Example 10–1* into the file.

**Example 10–1  A Security Helper File**

```java
package com.oracle.handson;

import com.tangosol.io.pof.PofPrincipal;
import com.tangosol.net.security.SecurityHelper;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.security.Principal;
import javax.security.auth.Subject;

/**
 * This class provides extremely simplified role based policies and access control.
 */
public class SecurityExampleHelper{
    // ----- static methods -------------------------------------------------
    /**
     * Login the user.
     *
     * @param sName the user name
     *
     * @return the authenticated user
     */
    public static Subject login(String sName){
        // For simplicity, just create a Subject. Normally, this would be
        // done using JAAS.
        String sUserDN = "CN=" + sName + ",OU=Yoyodyne";
        Set setPrincipalUser = new HashSet();
        setPrincipalUser.add(new PofPrincipal(sUserDN));
        setPrincipalUser.add(new PofPrincipal((String) s_mapUserToRole.get(sName)));
        return new Subject(true, setPrincipalUser, new HashSet(), new HashSet());
    }

    /**
     * Assert that a Subject is associated with the calling thread with a
     * Principal representing the required role.
     */
    
```
public static void checkAccess(String sRoleRequired)
{
    checkAccess(sRoleRequired, SecurityHelper.getCurrentSubject());
}

/**
 * Assert that a Subject contains a Principal representing the required
 * role.
 *
 * @param sRoleRequired  the role required for the operation
 *
 * @param subject        the Subject requesting access
 *
 * @throws SecurityException if a Subject is null or does not have the
 *         specified role Principal
 */
public static void checkAccess(String sRoleRequired, Subject subject)
{
    if (subject == null)
    {
        throw new SecurityException("Access denied, authentication
        required");
    }
    Map     mapRoleToId   = s_mapRoleToId;
    Integer nRoleRequired = (Integer) mapRoleToId.get(sRoleRequired);
    for (Iterator iter = subject.getPrincipals().iterator(); iter.hasNext();)
    {
        Principal principal = (Principal) iter.next();
        String    sName     = principal.getName();
        if (sName.startsWith("role_"))
        {
            Integer nRolePrincipal = (Integer) mapRoleToId.get(sName);
            if (nRolePrincipal == null)
            {
                // invalid role
                break;
            }
            if (nRolePrincipal.intValue() >= nRoleRequired.intValue())
            {
                return;
            }
        }
    }
    throw new SecurityException("Access denied, insufficient privileges");
}

// ----- constants  -----------------------------------------------------
public static final String ROLE_READER = "role_reader";
public static final String ROLE_WRITER = "role_writer";
public static final String ROLE_ADMIN = "role_admin";
/**
* The cache name for security examples
*/
public static final String SECURITY_CACHE_NAME = "security";
/**
* The name of the InvocationService used by security examples.
*/
public static String INVOCATION_SERVICE_NAME = "ExtendTcpInvocationService";

// ----- static data ---------------------------------------------------------
/**
* The map keyed by user name with the value being the user's role.
* Represents which user is in which role.
*/
private static Map s_mapUserToRole = new HashMap();
/**
* The map keyed by role name with the value the role id.
* Represents the numeric role identifier.
*/
private static Map s_mapRoleToId = new HashMap();

// ----- static initializer --------------------------------------------------

static
{
    // User to role mapping
    s_mapUserToRole.put("BuckarooBanzai", ROLE_ADMIN);
    s_mapUserToRole.put("JohnWhorfin", ROLE_WRITER);
    s_mapUserToRole.put("JohnBigboote", ROLE_READER);

    // Role to Id mapping
    s_mapRoleToId.put(ROLE_ADMIN, Integer.valueOf(9));
    s_mapRoleToId.put(ROLE_WRITER, Integer.valueOf(2));
    s_mapRoleToId.put(ROLE_READER, Integer.valueOf(1));
}

Create an Identity Transformer

An identity transformer (com.tangosol.net.security.IdentityTransformer) is a client-side component that converts a subject or principal into an identity token. The token must be a type that Coherence can serialize. Coherence automatically serializes the token at run time and sends it as part of the connection request to the proxy.

To create an identity transformer implementation:
1. Create a new Java class in the Security project named PasswordIdentityTransformer. See "Creating a Java Class" on page 2-11 for detailed information.

2. Import the IdentityTransformer interface. Ensure that the PasswordIdentityTransformer class implements the IdentityTransformer interface.

3. Implement the transformIdentity method so that it performs the following tasks:
   - Tests whether the subject exists and is complete
   - Gets the principal names from the subject and saves them in a String array
   - Constructs the token, as a combination of the password plus the principal names, that can be serialized as POF types

Example 10–2 illustrates a possible implementation of the PasswordIdentityTransformer class.

Example 10–2  Sample Identity Transformer Implementation

```java
package com.oracle.handson;

import com.tangosol.net.security.IdentityTransformer;
import java.security.Principal;
import java.util.Iterator;
import java.util.Set;
import javax.security.auth.Subject;

/**
 * PasswordIdentityTransformer creates a security token that contains the
 * required password and then adds a list of Principal names.
 */
public class PasswordIdentityTransformer
    implements IdentityTransformer
{
    // ----- IdentityTransformer interface -------------------------------

    /**
     * Transform a Subject to a token that asserts an identity.
     * @param subject the Subject representing a user.
     * @return the token that asserts identity.
     * @throws SecurityException if the identity transformation fails.
     */
    public Object transformIdentity(Subject subject, Service service)
        throws SecurityException
    {
        // The service is not needed so the service argument is being ignored.
        // It could be used, for example, if there were different token types
        // required per service.
    }
```
if (subject == null)
{
    throw new SecurityException("Incomplete Subject");
}

Set setPrincipals = subject.getPrincipals();

if (setPrincipals.isEmpty())
{
    throw new SecurityException("Incomplete Subject");
}

String[] asPrincipalName = new String[setPrincipals.size() + 1];
int i = 0;

asPrincipalName[i++] = System.getProperty("coherence.password", "secret-password");

for (Iterator iter = setPrincipals.iterator(); iter.hasNext();)
{
    asPrincipalName[i++] = ((Principal) iter.next()).getName();
}

// The token consists of the password plus the principal names as an
// array of pof-able types, in this case strings.
return asPrincipalName;
}

Create an Identity Asserter

An identity asserter (com.tangosol.net.security.IdentityAsserter) is a
cluster-side component on the cache server that hosts an extend proxy service. The
asserter validates that the token created by the identity transformer on the extend
client contains the required credentials to access the cluster.

To create an identity asserter implementation:

1. Create a new Java class in the Security project named
   PasswordIdentityAsserter.
   See "Creating a Java Class" on page 2-11 for detailed information.

2. Import the IdentityAsserter interface. Ensure that the
   PasswordIdentityAsserter class implements the IdentityAsserter
   interface.

3. Implement the assertIdentity method such that it:
   ■ Validates that the token contains the correct password and that the password
     is the first name in the token

Example 10–3 illustrates a a possible implementation of the
PasswordIdentityAsserter class.

Example 10–3   Sample Identity Asserter Implementation

package com.oracle.handson;

import com.tangosol.io.pof.PofPrincipal;
import com.tangosol.net.security.IdentityAsserter;
import java.util.HashSet;
import java.util.Set;
import javax.security.auth.Subject;

/**
 * PasswordIdentityAsserter asserts that the security token contains the
 * required password and then constructs a Subject based on a list of
 * Principal names.
 */

public class PasswordIdentityAsserter
    implements IdentityAsserter
{
    // ----- IdentityAsserter interface -------------------------------------

    /**
     * Asserts an identity based on a token-based identity assertion.
     * @param oToken the token that asserts identity.
     * @return a Subject representing the identity.
     * @throws SecurityException if the identity assertion fails.
     */
    public Subject assertIdentity(Object oToken, Service service)
        throws SecurityException
    {
        // The service is not needed so the service argument is being ignored.
        // It could be used, for example, if there were different token types
        // required per service.
        if (oToken instanceof Object[])
        {
            String sPassword = System.getProperty(
                "coherence.password", "secret-password");
            Set setPrincipalUser = new HashSet();
            Object[] asName = (Object[]) oToken;

            // first name must be password
            if (((String) asName[0]).equals(sPassword))
            {
                // prints the user name to server shell to ensure we are
                // communicating with it and to ensure user is validated
                System.out.println("Password validated for user: " + asName[1]);
                for (int i = 1, len = asName.length; i < len; i++)
                {
                    setPrincipalUser.add(new PofPrincipal((String)asName[i]));
                }

                return new Subject(true, setPrincipalUser, new HashSet(),
                        new HashSet());
            }
            throw new SecurityException("Access denied");
        }
    }
}
Create the Password File

Create a Java file that requires a password to get a reference to a cache. Use the SecurityExampleHelper.login("BuckarooBanzai") to call the login method in the SecurityExampleHelper file to generate a token. At run time, the user name is associated with its subject defined in the SecurityExampleHelper class. A token is generated from this subject by the PasswordIdentityTransformer class and validated by the PasswordIdentityAsserter class as part of the connection request. If the validation succeeds, then a connection to the proxy and a reference to the cache is granted. Use the Subject.doas method to make the subject available in the security context.

1. Create a new Java class with a main method in the Security project named PasswordExample.
   See "Creating a Java Class" on page 2-11 for detailed information.
2. Implement the main method to get a reference to the cache.
3. Use the SecurityExampleHelper.login method to get a subject for user BuckarooBanzai
4. Implement the doAs method to make the subject part of the Java security context. The subject will be available to any subsequent code. In this case, the doAs method is implemented to validate whether the user can access the cache based on its defined role.

Example 10–4 illustrates a possible implementation of PasswordExample.

Example 10–4 Sample Implementation to Run the Password Example

```java
package com.oracle.handson;
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import java.io.IOException;
import java.security.PrivilegedExceptionAction;
import javax.security.auth.Subject;

/**
 * This class shows how a Coherence Proxy can require a password to get a
 * reference to a cache.
 * <p>
 * The PasswordIdentityTransformer will generate a security token that
 * contains the password. The PasswordIdentityAsserter will validate the
 * security token to enforce the password. The token generation and
 * validation occurs automatically when a connection to the proxy is made.
 * 
 */
public class PasswordExample
{
  // ----- static methods -----------------------------------------------

  /**
   * Get a reference to the cache. Password will be required.
   */
  public static void main (String[] args){
```
Enabling Token-Based Security

```java
getCache();
}
public static void getCache()
{
System.out.println("-----password example begins------");

Subject subject = SecurityExampleHelper.login("BuckarooBanzai");

try
{
    NamedCache cache = (NamedCache) Subject.doAs(
        subject, new PrivilegedExceptionAction()
    {
        public Object run()
            throws Exception
        {
            NamedCache cache;

            cache = CacheFactory.getCache(
                SecurityExampleHelper.SECURITY_CACHE_NAME);
            System.out.println("-----password example succeeded------");
            return cache;
        }
    });

    catch (Exception e)
    {
        // get exception if the password is invalid
        System.out.println("Unable to connect to proxy");
        e.printStackTrace();
    }

    System.out.println("-----password example completed------");
}
```

Enable the Identity Transformer and Asserter

Configure an operational override file (`tangosol-coherence-override.xml`) to identify the classes that define the identity transformer (the class that transforms a `Subject` to a token on the extend client) and the identity asserter (the class that validates the token on the cluster).

1. Open the `tangosol-coherence-override.xml` file from the Project Explorer window. You can find the file under Security/appClientModule.

2. Use the `identity-transformer` and `identity-asserter` elements within the `security-config` stanza to identify the full path to the `PasswordIdentityTransformer` and `PasswordIdentityAsserter` implementation classes, respectively. Set the `subject-scope` parameter to true to associate the identity from the current security context with the cache and remote invocation service references that are returned to the client.

Example 10–5 illustrates a possible implementation of the `tangosol-coherence-override.xml` file.

**Example 10–5  Specifying an Identity Transformer and an Asserter**

```xml
<?xml version='1.0'?>
```
Enabling Token-Based Security

Create a Cache Configuration File for the Extend Client

The cache configuration file for the extend client routes cache operations to an extend proxy in the cluster. At run time, cache operations are not executed locally; instead, they are sent to the extend proxy service.

To create a cache configuration file for an extend client:

1. Open the coherence-cache-config.xml file from the Project explorer window. You can find the file under Security/appClientModule.
2. Save the file as client-cache-config.xml.
3. Write the extend client cache configuration. The following list highlights some key elements:
   - Use the cache-name element to define security as the name of the cache. Note that there must be a cache defined in the cluster-side cache configuration that is also named security.
   - Use the remote-cache-scheme stanza to define the details about the remote cache.
   - Use the address and port elements in the tcp-initiator stanza to identify the extend proxy service that is listening on the localhost address at port 9099.
   - Use defaults and serializer with a value of pof to call the serializer for the custom POF configuration file (which you will create later in this chapter).

Example 10–6 illustrates a possible implementation for the client-cache-config.xml file.

Example 10–6  Sample Extend Client Cache Configuration File

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <defaults>
    <serializer>pof</serializer>
  </defaults>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>security</cache-name>
      <scheme-name>examples-remote</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
</cache-config>
```
Create a Cache Configuration File for the Extend Proxy

To create a cache configuration file for the extend proxy service:

1. Open the coherence-cache-config.xml file from the Project explorer window. You can find the file under Security/appClientModule.
2. Save the file as examples-cache-config.xml.
3. Configure the extend proxy cache configuration file. The following list highlights some key elements:
   - Use the cache-name element to define security as the name of the cache. Note that there must be a cache defined in the extend client cache configuration that is also named security.
   - Use the address and port elements in the acceptor-config stanza to identify the extend proxy service that is listening on the localhost address at port 9099.
- Use the autostart element with the `tangosol.coherence.extend.enabled` system property to prevent the cache server from running a proxy service.

- Use `defaults` and `serializer` with a value of `pof` to call the serializer for the custom POF configuration file (which you will create later in this chapter).

Example 10–7 illustrates a possible implementation for the `examples-cache-config.xml` file.

**Example 10–7 Sample Cache Configuration File for the Proxy Server**

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <defaults>
    <serializer>pof</serializer>
  </defaults>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>security</cache-name>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <distributed-scheme>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
      <service-name>PartitionedPofCache</service-name>
      <backing-map-scheme>
        <local-scheme>
          <!-- each node will be limited to 32MB -->
          <high-units>32M</high-units>
          <unit-calculator>binary</unit-calculator>
        </local-scheme>
      </backing-map-scheme>
      <autostart>true</autostart>
    </distributed-scheme>
    <!-- Proxy Service scheme that allows remote clients to connect to the cluster over TCP/IP. -->
    <proxy-scheme>
      <scheme-name>secure-proxy</scheme-name>
      <service-name>ProxyService</service-name>
      <thread-count system-property="tangosol.coherence.extend.threads">2</thread-count>
      <acceptor-config>
        <tcp-acceptor>
          <local-address>
            <address system-property="tangosol.coherence.extend.address">localhost</address>
          </local-address>
          <port system-property="tangosol.coherence.extend.port">9099</port>
        </tcp-acceptor>
      </acceptor-config>
    </proxy-scheme>
  </caching-schemes>
</cache-config>
```

Working with Security 10-13
Create a Start-Up Configuration for a Cache Server

Create a start-up configuration for a cache server cluster node. The configuration must include the system properties to designate a proxy service and the cluster-side cache configuration file. You must also include the application class files and the XML configuration files on the class path.

To create a start-up file for a cache server:

1. Create a start-up configuration in Eclipse. Right-click the Security project and select Run As then Run Configurations. In the Name field, enter SecurityCacheServer.

2. In the Main tab, click Browse in the Project field to select the Security project. Select the Include system libraries when searching for a main class and click the Search button in the Main class field. Enter DefaultCacheServer in the Select type field of the Select Main Type dialog box. Select DefaultCacheServer - com.tangosol.net and click OK. Click Apply.

3. In the Coherence tab, enter the name and absolute path to the cluster-side cache configuration file (in this case, C:\home\oracle\workspace\Security\appClientModule\examples-cache-config.xml). Select Enabled (cache server) in the Local storage field. Enter a unique value (such as 3155) in the Cluster port field. Click Apply.

4. In the Classpath tab, click User Entries then Add External JARs to add the coherence.jar file to the list. Click Apply. The Classpath tab should look similar to Figure 10–1.

![Figure 10–1 Class Path for the Security Cache Server](image)

Figure 10–1 Class Path for the Security Cache Server

**Note:** Ensure that the XML configuration files (in the C:\home\oracle\workspace\Security and Security\build\classes folders) appear before the coherence.jar file on the class path. The classloader must encounter the custom POF configuration file (which must be named pof-config.xml and will be created later in this chapter) before it references the one in the coherence.jar file.
5. In the **Common** tab, click **Browse** in the **Shared file** field to select the **Security** project. Click **Apply**.

**Create a Start-Up Configuration for a Cache Server with a Proxy Service**

Create a configuration to start an extend proxy service on a cache server in the cluster. The extend client connects to this service. The configuration must include the system properties to designate a proxy service and the cluster-side cache configuration file. You must also include the application class files and the XML configuration files on the class path.

For these examples, the cache server with proxy service start-up configuration will have the same configuration as the cache server start-up configuration, but it will include the system property to enable the extend proxy.

To create a start-up file for a cache server with a proxy service:

1. Create a start-up configuration in Eclipse. Right-click the **Security** project and select **Run As** then **Run Configurations**. In the **Name** field, enter **SecurityRunProxy**.

2. In the **Main** tab, click **Browse** in the **Project** field to select the **Security** project. Select the **Include system libraries when searching for a main class** and click the **Search** button in the **Main class** field. Enter **DefaultCacheServer** in the **Select type** field of the **Select Main Type** dialog box. Select **DefaultCacheServer - com. tangosol.net** and click **OK**. Click **Apply**.

3. In the **Coherence** tab, enter the name and absolute path to the cluster-side cache configuration file (in this case, `C:\home\oracle\workspace\Security\appClientModule\examples-cache-config.xml`). Select **Enabled (cache server)** in the **Local storage** field. Enter a unique value (such as 3155) in the **Cluster port** field. Click **Apply**.

4. In the **Arguments** tab, enter the system property `-Dtangosol.coherence.extend.enabled=true` to designate a proxy service in the **VM arguments** field.

*Figure 10–2 Arguments Tab for the Security Proxy Server*
5. In the Classpath tab, click User Entries then Add External JARs to add the coherence.jar file to the list. Click Apply. The Classpath tab should look similar to Figure 10–3.

![Figure 10–3 Class Path for the Proxy Server](image)

**Note:** Ensure that the XML configuration files (in the C:\home\oracle\workspace\Security and Security\build\classes folders) appear before the coherence.jar file on the class path. The classloader must encounter the custom POF configuration file (which must be named pof-config.xml and will be created later in this chapter) before it references the one in the coherence.jar file.

6. In the Common tab, click Browse in the Shared file field to select the Security project. Click Apply.

### Run the Password Example

Run the password example to generate and validate the token, and pass it to the proxy service.

1. Create a run configuration for the PasswordExample.java file.
   a. Right click PasswordExample.java in the Project Explorer, and select Run As then Run Configurations.
   b. Click the New launch configuration icon. Ensure that PasswordExample appears in the Name field, Security appears in the Project field, and com.oracle.handson.PasswordExample appears in the Main class field. Click Apply.
   c. In the Coherence tab, enter the path to the client cache configuration file, C:\home\oracle\workspace\Security\appClientModule\client-cache-config.xml in the Cache configuration descriptor field. Select Disabled (cache client) in the Local cache field. Enter a unique value, such as 3155, in the Cluster port field.
   d. In the Classpath tab, click Add External JARs to add the coherence.jar file to User Entries. Use the Up and Down buttons to move the Security folder to the top of User Entries. When you are finished, the Classpath tab should look similar to Figure 10–4.
2. Stop any running cache servers. See “Stopping Cache Servers” on page 2-13 for more information.

3. Run the security proxy server, the security cache server then the PasswordExample.java program.
   a. Right click the project and select Run As then Run Configurations. Run the SecurityRunProxy configuration from the Run Configurations dialog box.
   b. Right click the project and select Run As then Run Configurations. Run the SecurityCacheServer configuration from the Run Configurations dialog box.
   c. Right click the PasswordExample.java file in the Project Explorer and select Run As then Run Configurations. Select PasswordExample in the Run Configurations dialog box and click Run.

The output of the PasswordExample program should be similar to Example 10–8 in the Eclipse console. It indicates the start of the password example, the opening of the socket to the proxy server, and the completion of the example.

Example 10–8  Password Example Output in the Eclipse Console

------password example begins------
2011-03-15 11:37:48.484/0.296 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml'
2011-03-15 11:37:48.531/0.343 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml'
2011-03-15 11:37:48.547/0.359 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from 'file:/C:/home/oracle/workspace/Security/build/classes/tangosol-coherence-override.xml'
2011-03-15 11:37:48.547/0.359 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override '/custom-mbeans.xml' is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-15 11:37:48.719/0.531 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from 'file:/C:/home/oracle/workspace/Security/appClientModule/client-cache-config.xml'
2011-03-15 11:37:48.844/0.656 Oracle Coherence GE 3.7.0.0 <Info> (thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded POF configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/pof-config.xml'
2011-03-15 11:37:48.875/0.687 Oracle Coherence GE 3.7.0.0 <Info> (thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded included POF configuration from 'jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-pof-config.xml'
Including Role-Based Access Control to the Cluster

This section describes how to create an example that uses role-based policies to access the cluster. The code logs in to get a subject with a user ID assigned to a particular role. It gets a cache reference running in the context of the subject and then attempt various...
cache operations. Depending on the role granted to the user, the cache operation is allowed or denied. Note that the role mapping and role-based authorization in the example is simplified and not intended for real security use.

For example, a user with a writer role can use the **put** and **get** methods. A user with a reader role can use the **get** method, but not the **put** method. A user with a writer role cannot destroy a cache; however, a user with an admin role can.

Note that when the cache reference is created in the context of a subject that identity is permanently associated with that reference. Any use of that cache reference is on behalf of that identity.

The example will use the `PasswordIdentityTransformer` and `PasswordIdentityAsserter` classes that you created in the previous section. The `PasswordIdentityTransformer` class generates a security token that contains the password, the user ID, and the roles. The `PasswordIdentityAsserter` class (running in the proxy) validates the security token to enforce the password and construct a subject with the proper user ID and roles. The production and assertion of the security token happens automatically.

To create the example:

1. **Define Which User Roles Are Entitled to Access Cache Methods**
2. **Apply the Entitlements to the Cache Service**
3. **Create the Access Control Example Program**
4. **Edit the Cluster-Side Cache Configuration File**
5. **Run the Access Control Example**

### Define Which User Roles Are Entitled to Access Cache Methods

Create a Java file that enables access to cache methods on the basis of a user’s role. To do this, you can apply access permissions to a wrapped `NamedCache` using the `Subject` object passed from the client by using Coherence*Extend. The implementation allows only clients with a specified role to access the wrapped `NamedCache`.

The class that you create in this section extends the `com.tangosol.net.cache.WrapperNamedCache` class. This class is a convenience function that enables you to secure the methods on the `NamedCache` interface.

To determine which user role can access a cache method, include a call to the `SecurityExampleHelper.checkAccess` method in each cache method’s implementation. As the argument to `checkAccess`, provide the user role that is permitted to access the method. Close the implementation with a call to `super`.

For example, the following code indicates that users with the `admin` role can destroy the cache.

```java
public void destroy()
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
    super.destroy();
}
```

In this example, users with the `reader` role can call the `aggregate` method.

```java
public Object aggregate(Filter filter, EntryAggregator agent)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
}```
To create a file that determines which user role can call cache methods:

1. Create a new Java class named `EntitledNamedCache` in the Security project. See “Creating a Java Class” on page 2-11 if you need detailed information.
2. Ensure that the class imports and extends `WrapperNamedCache`.
3. Import the `Filter`, `MapListener`, `ValueExtractor`, `Collection`, `Comparator`, `Map`, and `Set` classes. The methods in `WrapperNamedCache` (and by extension, `EntitledNamedCache`) use arguments with these types.
4. Implement the methods in `EntitledNamedCache` such that only a user with a specific role can call the method.

Example 10–10 illustrates a possible implementation of `EntitledNamedCache`.

Example 10–10  Entitled Named Cache

```java
package com.oracle.handson;

import com.tangosol.net.NamedCache;
import com.tangosol.net.security.SecurityHelper;
import com.tangosol.net.cache.WrapperNamedCache;
import com.tangosol.util.Filter;
import com.tangosol.util.MapEvent;
import com.tangosol.util.MapListener;
import com.tangosol.util.ValueExtractor;
import java.util.Collection;
import java.util.Comparator;
import java.util.Map;
import java.util.Set;
import javax.security.auth.Subject;

/**
 * Example WrapperNamedCache that demonstrates how entitlements can be applied to a wrapped NamedCache using the Subject passed from the client through Coherence*Extend. This implementation only allows clients with a specified role to access the wrapped NamedCache.
 */
public class EntitledNamedCache
    extends WrapperNamedCache
{
    /**
     * Create a new EntitledNamedCache.
     *
     * @param cache the wrapped NamedCache
     */
    public EntitledNamedCache(NamedCache cache)
    {
        super(cache, cache.getCacheName());
    }
```
// ----- NamedCache interface -------------------------------------------
/**
 * {inheritdoc}
 */
public void release()
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    super.release();
}

/**
 * {inheritdoc}
 */
public void destroy()
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
    super.destroy();
}

/**
 * {inheritdoc}
 */
public Object put(Object oKey, Object oValue, long cMillis)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.put(oKey, oValue, cMillis);
}

/**
 * {inheritdoc}
 */
public void addMapListener(MapListener listener)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    super.addMapListener(new EntitledMapListener(listener));
}

/**
 * {inheritdoc}
 */
public void removeMapListener(MapListener listener)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.removeMapListener(listener);
}

/**
 * {inheritdoc}
 */
public void addMapListener(MapListener listener, Object oKey, boolean fLite)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    super.addMapListener(new EntitledMapListener(listener), oKey, fLite);
}

/**
 * {inheritdoc}
 */
public void removeMapListener(MapListener listener, Object oKey)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.removeMapListener(listener, oKey);
}

/**
 * @inheritDoc
 */
public void addMapListener(MapListener listener, Filter filter, boolean fLite)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    super.addMapListener(new EntitledMapListener(listener), filter, fLite);
}

/**
 * @inheritDoc
 */
public void removeMapListener(MapListener listener, Filter filter)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.removeMapListener(listener, filter);
}

/**
 * @inheritDoc
 */
public int size()
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.size();
}

/**
 * @inheritDoc
 */
public void clear()
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.clear();
}

/**
 * @inheritDoc
 */
public boolean isEmpty()
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.isEmpty();
}

/**
 * @inheritDoc
 */
public boolean containsKey(Object oKey)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.containsKey(oKey);
}
public boolean containsValue(Object oValue) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.containsValue(oValue);
}

public Collection values() {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.values();
}

public void putAll(Map map) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.putAll(map);
}

public Set entrySet() {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.entrySet();
}

public Set keySet() {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.keySet();
}

public Object get(Object oKey) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.get(oKey);
}

public Object remove(Object oKey) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
return super.remove(oKey);
}
/**
 * @inheritDoc
 */
public Object put(Object oKey, Object oValue)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.put(oKey, oValue);
}
/**
 * @inheritDoc
 */
public Map getAll(Collection colKeys)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.getAll(colKeys);
}
/**
 * @inheritDoc
 */
public boolean lock(Object oKey, long cWait)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.lock(oKey, cWait);
}
/**
 * @inheritDoc
 */
public boolean lock(Object oKey)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.lock(oKey);
}
/**
 * @inheritDoc
 */
public boolean unlock(Object oKey)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.unlock(oKey);
}
/**
 * @inheritDoc
 */
public Set keySet(Filter filter)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.keySet(filter);
}
public Set entrySet(Filter filter) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.entrySet(filter);
}

/**
 * {@inheritDoc}
 */
public Set entrySet(Filter filter, Comparator comparator) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.entrySet(filter, comparator);
}

/**
 * {@inheritDoc}
 */
public void addIndex(ValueExtractor extractor, boolean fOrdered, Comparator comparator) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.addIndex(extractor, fOrdered, comparator);
}

/**
 * {@inheritDoc}
 */
public void removeIndex(ValueExtractor extractor) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.removeIndex(extractor);
}

/**
 * {@inheritDoc}
 */
public Object invoke(Object oKey, EntryProcessor agent) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.invoke(oKey, agent);
}

/**
 * {@inheritDoc}
 */
public Map invokeAll(Collection collKeys, EntryProcessor agent) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.invokeAll(collKeys, agent);
}

/**
 * {@inheritDoc}
 */
public Map invokeAll(Filter filter, EntryProcessor agent) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.invokeAll(filter, agent);
}
/**
 * @inheritDoc
 */
public Object aggregate(Collection collKeys, EntryAggregator agent) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.aggregate(collKeys, agent);
}

/**
 * @inheritDoc
 */
public Object aggregate(Filter filter, EntryAggregator agent) {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.aggregate(filter, agent);
}

// ----- inner class ----------------------------------------------------
/**
 * Example MapListener that adds authorization to map events.
 */
public class EntitledMapListener implements MapListener {
    // ----- constructors -------------------------------------------
    /**
     * Construct an EntitledMapListener with the current subject.
     * The subject will not be available in the security context
     * when events are received by the proxy at runtime.
     *
     * @param  listener  the MapListener
     */
    public EntitledMapListener(MapListener listener) {
        m_listener = listener;
        m_subject = SecurityHelper.getCurrentSubject();
    }

    // ----- MapListener interface ----------------------------------
    /**
     * @inheritDoc
     */
    public void entryInserted(MapEvent mapEvent) {
        try {
            SecurityExampleHelper.checkAccess(
                SecurityExampleHelper.ROLE_WRITER, m_subject);
        }
        catch (SecurityException e) {
            System.out.println("Access denied for entryInserted");
            return;
        }
    }
}
/**
 *  (\$\{inheritDoc\})
 */

public void entryUpdated(MapEvent mapEvent) {
    try {
        SecurityExampleHelper.checkAccess(
            SecurityExampleHelper.ROLE_WRITER, m_subject);
    }
    catch (SecurityException e) {
        System.out.println("Access denied for entryUpdated");
        return;
    }
    m_listener.entryUpdated(mapEvent);
}

/**
 *  (\$\{inheritDoc\})
 */

public void entryDeleted(MapEvent mapEvent) {
    try {
        SecurityExampleHelper.checkAccess(
            SecurityExampleHelper.ROLE_WRITER, m_subject);
    }
    catch (SecurityException e) {
        System.out.println("Access denied for entryDeleted");
        return;
    }
    m_listener.entryDeleted(mapEvent);
}

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

/**
 *  Subject from security context when the MapListener was registered
 */
private Subject m_subject;

/**
 *  Registered listener
 */
private MapListener m_listener;

// ----- helper methods ----------------------------------------

/**
 *  Return the wrapped NamedCache.
 */
Apply the Entitlements to the Cache Service

Create a file that demonstrates how access entitlements can be applied to a wrapped CacheService using the Subject passed from the client through Coherence*Extend. The implementation delegates access control for cache operations to the EntitledNamedCache you created in the previous section.

The class that you create extends the com.tangosol.net.WrapperCacheService class. This is a convenience function that allows you to secure the methods on the CacheService. It also provides a mechanism to delegate between the cache service on the proxy and the client request.

Implement the methods ensureCache, releaseCache, and destroyCache to ensure that only users with specific roles can use them. In the implementations, include a call to the SecurityExampleHelper.checkAccess method with a specific user role as its argument. For example, the following code ensures that only users with the role admin can destroy the cache.

```java
public void destroyCache(NamedCache map)
{
    if (map instanceof EntitledNamedCache)
    {
        EntitledNamedCache cache = (EntitledNamedCache) map;
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
        map = cache.getNamedCache();
    }
    super.destroyCache(map);
}
```

To create a file that applies entitlements to access the cache service:

1. Create a new Java class named EntitledCacheService in the Security project.
2. Ensure that the class imports and extends the WrapperCacheService class.
3. Implement the ensureCache, releaseCache, and destroyCache methods to ensure that only users with specific roles can use them.

Example 10–11 illustrates a possible implementation of EntitledCacheService.

Example 10–11  Entitled Cache Service

```java
package com.oracle.handson;

import com.tangosol.net.CacheService;
import com.tangosol.net.NamedCache;
import com.tangosol.net.WrapperCacheService;

/**
 * Example WrapperCacheService that demonstrates how entitlements can be
 * applied to a wrapped CacheService using the Subject passed from the
 * client through Coherence*Extend. This implementation delegates access control
```
public class EntitledCacheService
extends WrapperCacheService
{
/**
 * Create a new EntitledCacheService.
 *
 * @param service the wrapped CacheService
 */
public EntitledCacheService(CacheService service)
{
    super(service);
}

// ----- CacheService interface -----------------------------------------

/**
 * {@inheritDoc}
 */
public NamedCache ensureCache(String sName, ClassLoader loader)
{
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return new EntitledNamedCache(super.ensureCache(sName, loader));
}

/**
 * {@inheritDoc}
 */
public void releaseCache(NamedCache map)
{
    if (map instanceof EntitledNamedCache)
    {
        EntitledNamedCache cache = (EntitledNamedCache) map;
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        map = cache.getNamedCache();
    }
    super.releaseCache(map);
}

/**
 * {@inheritDoc}
 */
public void destroyCache(NamedCache map)
{
    if (map instanceof EntitledNamedCache)
    {
        EntitledNamedCache cache = (EntitledNamedCache) map;
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
        map = cache.getNamedCache();
    }
    super.destroyCache(map);
}
Create the Access Control Example Program

Create a file to run the access control example. The role policies are defined in the SecurityExampleHelper class. The EntitledCacheService and EntitledNamedCache classes enforce the policies.

The program should specify various users as arguments to the SecurityHelperFile.login method, and then attempt to perform cache read, write, and destroy operations. Based on the entitlement policies defined in the EntitledCacheService and EntitledNamedCache classes, the operations succeed or fail.

1. Create a new Java class with a main method in the Security project named AccessControlExample.
   See “Creating a Java Class” on page 2-11 for detailed information.
2. Implement the main method to access the cache.
3. Specify users defined in the SecurityExampleHelper file as arguments to its login method.
4. For each user, execute read (get), write (put), and destroy operations on the cache and provide success or failure messages in response.

Example 10–12 illustrates a possible implementation of the AccessControlExample.java class.

Example 10–12  Sample Program to Run the Access Control Example

```java
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.InvocationService;
import com.tangosol.net.NamedCache;
import com.tangosol.util.MapEvent;
import com.tangosol.util.MapListener;
import java.security.PrивilegedExceptionAction;
import javax.security.auth.Subject;

/**
 * This class demonstrates simplified role based access control.
 * <p>
 * The role policies are defined in SecurityExampleHelper. Enforcement
 * is done by EntitledCacheService and EntitledNamedCache.
 * *
 */
public class AccessControlExample {

    // ----- static methods -----------------------------------------------

    public static void main (String[] args){
        accessCache();
    }

    /**
     * Demonstrate role based access to the cache.
     */
    public static void accessCache()
    {
        // ...
    }
```
Including Role-Based Access Control to the Cluster

System.out.println("------cache access control example begins------");

Subject subject = SecurityExampleHelper.login("JohnWhorfin");

// Someone with writer role can write and read
try {
    NamedCache cache = (NamedCache) Subject.doAs(
        subject, new PrivilegedExceptionAction()
        {
            public Object run()
            throws Exception
            {
                return CacheFactory.getCache(SecurityExampleHelper.SECURITY_CACHE_NAME);
            }
        });
    cache.put("myKey", "myValue");
    cache.get("myKey");
    System.out.println("    Success: read and write allowed");
} catch (Exception e) {
    // get exception if not allowed to perform the operation
    e.printStackTrace();
}

// Someone with reader role can read but not write
subject = SecurityExampleHelper.login("JohnBigboote");
try {
    NamedCache cache = (NamedCache) Subject.doAs(
        subject, new PrivilegedExceptionAction()
        {
            public Object run()
            throws Exception
            {
                return CacheFactory.getCache(SecurityExampleHelper.SECURITY_CACHE_NAME);
            }
        });
    cache.get("myKey");
    System.out.println("    Success: read allowed");
    cache.put("anotherKey", "anotherValue");
} catch (Exception e) {
    // get exception if not allowed to perform the operation
    System.out.println("    Success: Correctly cannot write");
}

// Someone with writer role cannot call destroy
subject = SecurityExampleHelper.login("JohnWhorfin");
try {
    NamedCache cache = (NamedCache) Subject.doAs(
        subject, new PrivilegedExceptionAction()
        {
            public Object run()
            throws Exception
Edited the Cluster-Side Cache Configuration File

Edit the cluster-side cache configuration file `examples-cache-config.xml`. Specify the full path of the class name of the cache service in the `cache-service-proxy` stanza under `proxy-config`. The `cache-service-proxy` stanza contains the configuration information for a cache service proxy managed by a proxy service.

In this case, the cache service class name is `com.oracle.handson.EntitledCacheService` proxy and the `param-type` is `com.tangosol.net.CacheService`.

Example 10–13 illustrates the XML code to add to the configuration.

**Example 10–13** Cache Service Proxy Configuration for a Cluster-Side Cache Configuration

```xml
...<proxy-config>
  <cache-service-proxy>
```
Run the Access Control Example

Run the access control example to demonstrate how access to the cache can be granted or denied, based on a user’s role.

1. Create a run configuration for the AccessControlExample.java.
   a. Right click AccessControlExample in the Project Explorer. Select Run As then Run Configurations.
   b. Click Oracle Coherence then New launch configuration icon. Ensure that AccessControlExample appears in the Name field, Security appears in the Project field, and com.oracle.handson.AccessControlExample appears in the Main class field. Click Apply.
   c. In the Coherence tab, enter the path to the client-cache-config.xml file in the Cache configuration descriptor field. Select Disabled (cache client) in the Local storage field.
   d. In the Classpath tab, click Add External JARs to add the coherence.jar file to User Entries. The Classpath tab should look similar to Figure 10–5. Click Apply then Close.


3. Run the security proxy server, the security cache server then the AccessControlExample.java program.
   a. Right click the project and select Run As then Run Configurations. Run the SecurityRunProxy configuration from the Run Configurations dialog box.
   b. Right click the project and select Run As then Run Configurations. Run the SecurityCacheServer configuration from the Run Configurations dialog box.
c. Right click the AccessControlExample.java file in the Project Explorer and select Run As then Run Configurations. Run the AccessControlExample configuration from the Run Configurations dialog box.

The output is similar to Example 10–14 in the Eclipse console. The messages correspond to the various users specified in AccessControlExample executing read, write, and destroy operations on the cache.

- The Success: read and write allowed message corresponds to the user with role writer attempting to read from and write to the cache
- The Success: read allowed message corresponds to the user with role reader attempting to read from the cache
- The Success: Correctly cannot write message corresponds to the user with role reader attempting to write to the cache
- The Success: Correctly cannot destroy the cache message corresponds to the user with role writer attempting to destroy the cache
- The Success: Correctly allowed to destroy the cache message corresponds to the user with role admin attempting to destroy the cache

**Example 10–14  Access Control Example Output in the Eclipse Console**

```bash
------cache access control example begins------
2011-03-15 14:05:12.953/0.313 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2011-03-15 14:05:13.000/0.360 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2011-03-15 14:05:13.000/0.360 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "file:/C:/home/oracle/workspace/Security/appClientModule/tangosol-coherence-override.xml"
2011-03-15 14:05:13.000/0.360 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified
```

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2011-03-15 14:05:13.156/0.516 Oracle Coherence GE 3.7.0.0 <Info> (thread=main, member=n/a): Loaded cache configuration from "file:/C:/home/oracle/workspace/Security/appClientModule/client-cache-config.xml"
2011-03-15 14:05:13.281/0.641 Oracle Coherence GE 3.7.0.0 <Info> (thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded POF configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/pof-config.xml"
2011-03-15 14:05:13.312/0.672 Oracle Coherence GE 3.7.0.0 <Info> (thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded included POF configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-pof-config.xml"
2011-03-15 14:05:13.359/0.719 Oracle Coherence GE 3.7.0.0 <D5> (thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Started: TcpInitiator(Name=ExtendTcpCacheService:TcpInitiator, State=(SERVICE_STARTED), ThreadCount=0, Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.ConfigurablePofContext, PingInterval=0, PingTimeout=30000, RequestTimeout=30000, ConnectTimeout=30000, SocketProvider=SystemSocketProvider, RemoteAddresses=[/130.35.99.213:9099], SocketOptions(LingerTimeout=0, KeepAliveEnabled=true, TcpDelayEnabled=false))
2011-03-15 14:05:13.375/0.735 Oracle Coherence GE 3.7.0.0 <D5> (thread=main, member=n/a): Connecting Socket to 130.35.99.213:9099

---
Example 10–15 lists the output in the shell where the cache server is running the proxy service. Notice that the security exceptions in the output correspond to the Success: Correctly cannot write and Success: Correctly cannot destroy the cache messages in the Eclipse console.
Including Role-Based Access Control to the Cluster

Password validated for user: role_reader
Password validated for user: role_reader
Password validated for user: role_reader

Password validated for user: role_writer
Password validated for user: role_writer
Password validated for user: role_writer

Password validated for user: role_reader
Password validated for user: role_reader
2011-03-15 14:05:05/39.755 Oracle Coherence GE 3.7.0.0 <D5>
(thread=Proxy:ProxyService:TcpAcceptorWorker:1, member=1): An exception occurred while processing a
    at com.tangosol.coherence.component.net.extend.messageFactory.CacheServiceFactory$DestroyCacheRequest.onRun(CacheServiceFactory.CDB:6)
    at com.tangosol.coherence.component.net.extend.message.Request.run(Request.CDB:4)
    at com.tangosol.coherence.component.net.extend.proxy.serviceProxy.CacheServiceProxy.onMessage(CacheServiceProxy.CDB:9)
    at com.tangosol.coherence.component.net.extend.Channel$MessageAction.run(Channel.CDB:13)
    at java.lang.Thread.run(Thread.java:619)

2011-03-15 14:05:13.500/40.985 Oracle Coherence GE 3.7.0.0 <D5>
(thread=Proxy:ProxyService:TcpAcceptorWorker:1, member=1): An exception occurred while processing a
    at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:104)
    at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:58)
    at com.oracle.handson.EntitledNamedCache.put(EntitledNamedCache.java:69)
    at com.tangosol.coherence.component.net.extend.proxy.NamedCacheProxy.put$Router(NamedCacheProxy.CDB:1)
    at com.tangosol.coherence.component.net.extend.proxy.NamedCacheProxy.put(NamedCacheProxy.CDB:2)
    at com.tangosol.coherence.component.net.extend.messageFactory.NamedCacheFactory$PutRequest.onRun(NamedCacheFactory.CDB:6)
    at com.tangosol.coherence.component.net.extend.message.Request.run(Request.CDB:4)
    at com.tangosol.coherence.component.net.extend.proxy.serviceProxy.CacheServiceProxy.onMessage(CacheServiceProxy.CDB:9)
    at com.tangosol.coherence.component.net.extend.Channel$MessageAction.run(Channel.CDB:13)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:337)
    at com.tangosol.coherence.component.net.extend.Channel.execute(Channel.CDB:29)
    at com.tangosol.coherence.component.net.extend.Channel.receive(Channel.CDB:26)

2011-03-15 14:05:05/40.985 Oracle Coherence GE 3.7.0.0 <D5>
(thread=Proxy:ProxyService:TcpAcceptorWorker:1, member=1): An exception occurred while processing a
    at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:104)
    at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:58)
    at com.oracle.handson.EntitledCacheService.destroyCache(EntitledCacheService.java:63)
    at com.tangosol.coherence.component.net.extend.messageFactory.CacheServiceFactory$DestroyCacheRequest.onRun(CacheServiceFactory.CDB:6)
    at com.tangosol.coherence.component.net.extend.message.Request.run(Request.CDB:4)
    at com.tangosol.coherence.component.net.extend.proxy.serviceProxy.CacheServiceProxy.onMessage(CacheServiceProxy.CDB:9)
    at com.tangosol.coherence.component.net.extend.Channel$MessageAction.run(Channel.CDB:13)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:337)
    at com.tangosol.coherence.component.net.extend.Channel.execute(Channel.CDB:29)
    at com.tangosol.coherence.component.net.extend.Channel.receive(Channel.CDB:26)
Including Role-Based Access Control to an Invocable Object

An invocation service cluster service enables extend clients to execute invocable objects on the cluster. This example demonstrates how you can use role-based policies to determine which users can run the invocable objects.

For example, a user with a writer role can run an invocable object. A user with a reader role cannot.

In this example, you create a simple invocable object that can be called from a client program. Since the invocable object will be serializable, you must also list it in a POF configuration file. You also create an invocation service program that tests whether a user can execute methods on the service based on the user’s role.

As in the previous example, this example uses the PasswordIdentityTransformer class to generate a security token that contains the password, the user ID, and the roles. The PasswordIdentityAsserter (running in the proxy) will be used to validate the security token to enforce the password and construct a subject with the proper user ID and roles. The production and assertion of the security token happens automatically.

To create the example:

1. **Create an Invocable Object**
2. **Create an Entitled Invocation Service**
3. **Create the Access Invocation Service Example Program**
4. **Edit the Cluster-Side Cache Configuration File**
5. **Create a POF Configuration File**
6. **Edit the Run Configurations for the Servers**
7. **Run the Access Invocation Service Example**

**Create an Invocable Object**

Create an implementation of a simple invocable object that will be used by an entitled invocation service. For example, the invocable object can be written to increment and return an integer.

To create an invocable object:

1. Create a new Java class named `ExampleInvocable` in the `Security` project. See "Creating a Java Class" on page 2-11 for detailed information.
2. Import the `Invocable` and `InvocationService` interfaces. Because this class will be working with serializable objects, import the `PortableObject`, `PofReader` and `PofWriter` classes.
3. Ensure that the ExampleInvocable class implements Invocable and PortableObject.

4. Implement the ExampleInvocable class to increment an integer and return the result.

5. Implement the PofReader.readExternal and PofWriter.writeExternal methods.

Example 10–16 illustrates a possible implementation of ExampleInvocable.java.

**Example 10–16  A Sample Invocable Object**

```java
package com.oracle.handson;

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

import com.tangosol.net.Invocable;
import com.tangosol.net.InvocationService;

import java.io.IOException;

/**
 * Invocable implementation that increments and returns a given integer.
 */
public class ExampleInvocable
    implements Invocable, PortableObject
{
    // ----- constructors ---------------------------------------------

    /**
     * Default constructor.
     */
    public ExampleInvocable()
    {
    }

    // ----- Invocable interface --------------------------------------

    /**
     * init(InvocationService service)
     */
    public void init(InvocationService service)
    {
        m_service = service;
    }

    /**
     * run()
     */
    public void run()
    {
        if (m_service != null)
        {
            m_nValue++;
        }
    }

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

    /**
     * PofReader.readExternal
     */
    public void readExternal(PofReader reader)
    {
    }

    /**
     * PofWriter.writeExternal
     */
    public void writeExternal(PofWriter writer)
    {
    }
}
```
Create an Entitled Invocation Service

This example shows how a remote invocation service can be wrapped to provide access control. Access entitlements can be applied to a wrapped InvocationService using the Subject passed from the client by using Coherence*Extend. This implementation enables only clients with a specified role to access the wrapped invocation service.

The class that you create should extend the com.tangosol.net.WrapperInvocationService class. This is a convenience function that enables you to secure the methods on InvocationService. It also provides a mechanism to delegate between the invocation service on the proxy and the client request.

To create an entitled invocation service:
1. Create a new Java class named `EntitledInvocationService` in the `Security` project.

2. Import the `Invocable`, `InvocationObserver`, `InvocationService`, `WrapperInvocationService`, `Map`, and `Set` interfaces. Ensure that the `EntitledInvocationService` class extends the `WrapperInvocationService` class.

3. Implement the `query` and `execute` methods. In the implementations, include a call to the `SecurityExampleHelper.checkAccess` method to determine whether a specified user role, in this case, `ROLE_WRITER`, can access these operations.

Example 10–17 illustrates a possible implementation of `EntitledInvocationService.java`.

**Example 10–17  A Sample Entitled Invocation Service**

```java
package com.oracle.handson;

import com.tangosol.net.Invocable;
import com.tangosol.net.InvocationObserver;
import com.tangosol.net.InvocationService;
import com.tangosol.net.WrapperInvocationService;
import java.util.Map;
import java.util.Set;

/**
 * Example WrapperInvocationService that demonstrates how entitlements can be
 * applied to a wrapped InvocationService using the Subject passed from the
 * client through Coherence*Extend. This implementation only allows clients with a
 * specified role to access the wrapped InvocationService.
 * *
 */
public class EntitledInvocationService
    extends WrapperInvocationService
{
    /**
     * Create a new EntitledInvocationService.
     *
     * @param service the wrapped InvocationService
     */
    public EntitledInvocationService(InvocationService service)
    {
        super(service);
    }

    // ----- InvocationService interface ------------------------------------
    /**
     * {@inheritDoc}
     */
    public void execute(Invocable task, Set setMembers, InvocationObserver observer)
    {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.execute(task, setMembers, observer);
    }
}
```
Create the Access Invocation Service Example Program

Create a program to run the access invocation service example. The objective of the program is to test whether various users defined in the SecurityExampleHelper class are able to access and run an invocable object. The enforcement of the role-based policies is provided by the EntitledInvocationService class.

To create a program to run the Access Invocation Service example:

1. Create a Java class with a main method in the Security project named AccessInvocationServiceExample.java. See "Creating a Java Class" on page 2-11 for detailed information.

2. Among other classes, import ExampleInvocable, CacheFactory, and InvocationService.

3. Implement the main method to invoke the accessInvocationService method.

4. Implement the accessInvocationService class so that various users defined in the SecurityExampleHelper class attempt to log in to the service and run the object defined in ExampleInvocable. Use the SecurityExampleHelper.login method to test whether various users can access the invocable service.

Example 10–18 illustrates a possible implementation of AccessInvocationServiceExample.java.

Example 10–18 Sample Program to Run the Access Invocation Service Example

```java
package com.oracle.handson;

import com.oracle.handson.ExampleInvocable;
import com.tangosol.net.CacheFactory;
import com.tangosol.net.InvocationService;
import java.security.PrivilegedExceptionAction;
import javax.security.auth.Subject;

/**
 * This class demonstrates simplified role based access control for the
 * invocation service.
 * <p>
 * The role policies are defined in SecurityExampleHelper. Enforcement
 * is done by EntitledInvocationService.
 * *
 */
```
public class AccessInvocationServiceExample
{
   /**
     * Invoke the example
     * @param asArg  command line arguments (ignored in this example)
     */
    public static void main(String[] asArg)
    {
        accessInvocationService();
    }

    /**
     * Access the invocation service
     */
    public static void accessInvocationService()
    {
        System.out.println("------InvocationService access control example " + 
                           "begins------");

        // Someone with writer role can run invocables
        Subject subject = SecurityExampleHelper.login("JohnWhorfin");

        try
        {
            InvocationService service = (InvocationService) Subject.doAs(
                subject, new PrivilegedExceptionAction()
            {
                public Object run()
                {
                    return CacheFactory.getService(
                        SecurityExampleHelper.INVOCATION_SERVICE_NAME);
                }
            });
            service.query(new ExampleInvocable(), null);
            System.out.println("    Success: Correctly allowed to " + 
                               "use the invocation service");
        }
        catch (Exception e)
        {
            // get exception if not allowed to perform the operation
            e.printStackTrace();
        }

        // Someone with reader role cannot cannot run invocables
        subject = SecurityExampleHelper.login("JohnBigboote");
        try
        {
            InvocationService service = (InvocationService) Subject.doAs(
                subject, new PrivilegedExceptionAction()
            {
                public Object run()
                {
                    return CacheFactory.getService(
                        SecurityExampleHelper.INVOCATION_SERVICE_NAME);
                }
            });
            service.query(new ExampleInvocable(), null);
        }
        catch (Exception ee)
Edit the Cluster-Side Cache Configuration File

Edit the examples-cache-config.xml file to add the full path to the invocation service to the invocation-service-proxy stanza under the proxy-config stanza. The invocation-service-proxy stanza contains the configuration information for an invocation service proxy managed by a proxy service.

In this case, the invocation service class name is com.oracle.handson.EntitledInvocationService and its param-type is com.tangosol.net.InvocationService.

Example 10–19 Invocation Service Proxy Configuration for a Cluster-Side Cache

```xml
<proxy-config>
  ...
  <invocation-service-proxy>
    <class-name>com.oracle.handson.EntitledInvocationService</class-name>
    <init-params>
      <init-param>
        <param-type>com.tangosol.net.InvocationService</param-type>
        <param-value>{service}</param-value>
      </init-param>
    </init-params>
  </invocation-service-proxy>
  ...
</proxy-config>
```

Create a POF Configuration File

Create a POF configuration file to declare ExampleInvocable as a user type.

1. Locate the pof-config.xml file under Security\appClientModule in the Project Explorer and open it in the Eclipse IDE.
2. Enter the code to declare ExampleInvocable as a user type and save the file.

The contents of the file should look similar to Example 10–20. The file will be saved to the C:\home\oracle\workspace\Security\appClientModule folder.

Example 10–20 POF Configuration File with ExampleInvocable User Type

```xml
<?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>
```
Including Role-Based Access Control to an Invocable Object

```xml
<!-- com.tangosol.examples package -->
<user-type>
  <type-id>1007</type-id>
  <class-name>com.oracle.handson.ExampleInvocable</class-name>
</user-type>
</user-type-list>

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

**Edit the Run Configurations for the Servers**

The classloader must encounter the custom POF configuration file (which must be named `pof-config.xml`) before it references the one in the `coherence.jar` file. If it does not, then the custom POF configuration file will be ignored and the default file in the `coherence.jar` file will be used instead.

To ensure that the XML configuration files in the `C:\home\oracle\workspace\Security\appModule` are used, delete the Coherence37 library from the **Bootstrap Entries** section of the servers’ class path. Also, position the `coherence.jar` file after the Security folder in the **User Entries** section.

1. Right click the project in the **Project Explorer** and select **Run As** then **Run Configurations**.
2. Select **SecurityRunProxy**. In the **Classpath** tab, remove **Coherence37** from the list of **Bootstrap Entries**. Click **Apply**.
3. Select **SecurityCacheServer**. In the **Classpath** tab, remove **Coherence37** from the list of **Bootstrap Entries**. Click **Apply**.

When you are finished, the **Classpath** tab for **SecurityRunProxy** and **SecurityCacheServer** should look similar to **Figure 10–6**.

**Figure 10–6  Class Path for the Cache Server and the Server Running the Proxy Service**

![Class Path for the Cache Server and the Server Running the Proxy Service](image)

**Run the Access Invocation Service Example**

Run the access invocation service example to demonstrate how access to the invocable object can be granted or denied, based on a user’s role.

1. Create a run configuration for the `AccessInvocationServiceExample.java` file.
   a. Right click `AccessInvocationServiceExample.java` in the **Project Explorer** and select **Run As** then **Run Configurations**.

---

10-44  Oracle Coherence Tutorial for Oracle Coherence
b. Click Oracle Coherence, then the New launch configuration icon. Ensure that AccessInvocationServiceExample appears in the Name field, Security appears in the Project field, and com.oracle.handson.AccessInvocationServiceExample.java appears in the Main class field. Click Apply.

c. In the Coherence tab, enter the path to the client-cache-config.xml file in the Cache configuration descriptor field. Select Disabled (cache client) in the Local storage field.

d. In the Classpath tab, remove Coherence37 from the Bootstrap Entries list. Click Add External JARs to add the coherence.jar file to User Entries. Move the Security folder to the top of User Entries followed by the coherence.jar file. When you are finished, the Classpath tab should look similar to Figure 10–7. Click Apply then Close.

**Figure 10–7 Classpath Tab for the AccessInvocationServiceExample Program**

```
Name: AccessInvocationServiceExample

Main | Coherence | Arguments | JRE | Classpath

Classpath:
- Bootstrap Entries
  - JRE System Library [JavaSE-1.6]
- User Entries
  - Security (default classpath)
    - Security
    - EAR Libraries
    - coherence.jar - C:\oracle\product\coherence\lib
```


3. Run the security proxy server, the security cache server then the AccessInvocationServiceExample.java program.

   a. Right click the project and select Run As then Run Configurations. Run the SecurityRunProxy configuration from the Run Configurations dialog box.

   b. Right click the project and select Run As then Run Configurations. Run the SecurityCacheServer configuration from the Run Configurations dialog box.

   c. Right click the AccessInvocationServiceExample.java file in the Project Explorer and select Run As then Run Configurations. Select AccessControlExample in the Run Configurations dialog box and click Run.

   The output is similar to Example 10–21 in the Eclipse console. The messages correspond to the users specified in the AccessInvocationServiceExample class that are trying to run the invocable object ExampleInvocable.

   - The message Success: Correctly allowed to use the invocation service corresponds to the user with role writer attempting to run ExampleInvocable.

   - The message Success: Correctly unable to use the invocation service corresponds to the user with role reader attempting to run ExampleInvocable.
Example 10–21  Client Program Response in the Eclipse Console

-----InvocationService access control example begins-----
2011-03-15 17:54:40.468/0.453 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2011-03-15 17:54:40.515/0.500 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2011-03-15 17:54:40.515/0.500 Oracle Coherence 3.7.0.0 <Info> (thread=main, member=n/a): Loaded operational overrides from "file:/C:/home/oracle/workspace/Security/appClientModule/tangosol-coherence-override.xml"
2011-03-15 17:54:40.515/0.500 Oracle Coherence 3.7.0.0 <D5> (thread=main, member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.7.0.0 Build 22913
Grid Edition: Development mode

Example 10–22 lists the output in the shell where the cache server is running the proxy service. Notice that the security exception in the output corresponds to the Success: Correctly unable to use the invocation service message in the Eclipse console, where the user with role reader attempts to run ExampleInvocable.
Example 10–22  Proxy Service Response in the Eclipse Console

... Started DefaultCacheServer...

2011-03-15 17:54:08.375/37.750 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1):
Member(Id=2, Timestamp=2011-03-15 17:54:08.226, Address=130.35.99.213:8090, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5340, Role=CoherenceServer) joined
Cluster with senior member 1

2011-03-15 17:54:08.890/38.265 Oracle Coherence GE 3.7.0.0 <D5> (thread=Cluster, member=1):
Member 2 joined Service PartitionedPofCache with senior member 1

2011-03-15 17:54:08.968/38.343 Oracle Coherence GE 3.7.0.0 <D4> (thread=DistributedCache:PartitionedPofCache, member=1):
> Transferring primary PartitionSet{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102,
123, 124, 125, 126, 127) to member 2 requesting 128

2011-03-15 17:54:09.000/38.375 Oracle Coherence GE 3.7.0.0 <D5> (thread=DistributedCache:PartitionedPofCache, member=1):
> Transferring 0KB of backup[1] for PartitionSet(128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144,
145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164,
165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184,
185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204,
205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224,
225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244,
245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256) to member 2

Password validated for user: role_writer
Password validated for user: role_writer
Password validated for user: role_reader
Password validated for user: role_reader

2011-03-15 17:54:41.156/70.531 Oracle Coherence GE 3.7.0.0 <D5> (thread=Proxy:ProxyService:TcpAcceptor:1, member=1):
denied, insufficient privileges

at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:104)
at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:58)
at com.oracle.handson.EntitledInvocationService.query(EntitledInvocationService.java:50)
at com.tangosol.coherence.component.net.extend.messageFactory.
InvocationServiceFactory$InvocationRequest.onRun(InvocationServiceFactory.CDB:12)
at com.tangosol.coherence.component.net.extend.proxy.serviceProxy.InvocationServiceProxy.
onMessage(InvocationServiceProxy.CDB:9)
at com.tangosol.coherence.component.net.extend.Channel$MessageAction.run(Channel.CDB:13)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:337)
at com.tangosol.coherence.component.net.extend.Channel.execute(Channel.CDB:29)
at com.tangosol.coherence.component.net.extend.Channel.receive(Channel.CDB:26)
at com.tangosol.coherence.component.util.daemon.queueProcessor.service.
Peer$DaemonPool$WrapperTask.run(Peer.CDB:9)
at com.tangosol.coherence.component.util.DaemonPool$WrapperTask.run(DaemonPool.CDB:32)
at com.tangosol.coherence.component.util.DaemonPool$Daemon.onNotify(DaemonPool.CDB:63)
at com.tangosol.coherence.component.util.Daemon.run(Daemon.CDB:42)
at java.lang.Thread.run(Thread.java:619)
This chapter describes how to cache session information for Web application instances that are deployed across WebLogic Server instances.

This chapter has the following sections:

- Introduction
- Caching Session Information for Web Application Instances

**Introduction**

WebLogic Server includes features that enable deployed applications to use Coherence data caches and seamlessly incorporate Coherence*Web for session management and TopLink Grid as an object-to-relational persistence framework. Collectively, these features are referred to as *ActiveCache*.

ActiveCache is employed by applications running on WebLogic Server and provides replicated and distributed caching services that make an application's data available to all servers in a Coherence data cluster. Applications can obtain direct access to data caches either through resource injection or component-based JNDI lookup. You can display, monitor, create, and configure Coherence clusters using the WebLogic Server Administration Console and WLST.

Using ActiveCache with WebLogic Server instances enables you to create a data tier dedicated to caching application data and storing replicated session state. This is separate from the application tier, where the WebLogic Server instances are dedicated to running the application.

Using Coherence*Web with ActiveCache enables you to provide Coherence-based HTTP session state persistence to applications running on WebLogic Server. Coherence*Web enables HTTP session sharing and management across different Web applications, domains, and heterogeneous application servers. Session data can be stored in data caches outside of the application server, thus freeing application server heap space and enabling server restarts without losing session data.

Coherence and Coherence*Web are included in the default installation of WebLogic Server 11g Release 1 (10.3.4). If you do not already have WebLogic Server 11g Release 1 (10.3.4) on your system, you can get it at the following URL:


For more information on the integration of Oracle WebLogic Server, Coherence, and Coherence*Web, see *Using ActiveCache* and *User’s Guide for Oracle Coherence*Web*. 
Caching Session Information for Web Application Instances

The following example demonstrates how to use ActiveCache to cache session information for Web application instances that are deployed across WebLogic Server instances. To do this, you will create a Web application and deploy it to two 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. A browser is used to access each application instance to demonstrate that the same session attribute is used among the instances.

1. Ensure That You are Using Coherence 3.7 with WebLogic Server 10.3.4
2. Start a Cache Server
3. Configure and Start the WebLogic Server
4. Create a Machine
5. Create the WebLogic Servers
6. Create a Coherence Cluster
7. Deploy the Shared Library Files
8. Create the Counter Web Application
9. Deploy the Application
10. Start the Node Manager and the WebLogic Servers
11. Verify the Example

Ensure That You are Using Coherence 3.7 with WebLogic Server 10.3.4

By default, the installation of WebLogic Server 11g release 1 (10.3.4) installs Coherence 3.6 in the coherence_3.6 folder. To complete this example, ensure that your server start-up files and class paths continue to point to the Coherence 3.7 files you have already installed. Also, ensure that when you are deploying files (such as coherence.jar and coherence-web-spi.war) later in this example, that you are deploying release 3.7 versions.

Start a Cache Server

Start a Coherence cache server. Example 11–1 illustrates a sample script to start the cache server. In this example, tangosol.coherence.clusterport=7777 is the default multicast listen port of a Coherence cluster and tangosol.coherence.clusteraddress=231.1.1.1 is the default multicast listener address.

Example 11–1 Script to Start the Cache Server

```bash
setlocal

set COHERENCE_HOME=c:\oracle\product\coherence

set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.jar;%COHERENCE_HOME%\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.distributed.localstorage=true -Dtangosol.coherence.clusterport=7777 -Dtangosol.coherence.clusteraddress=231.1.1.1 -Dtangosol.coherence.session.localstorage=true

java %COH_OPTS% -Xms512m -Xmx512m -XX:MaxPermSize=256m com.tangosol.net.
```
DefaultCacheServer

:exit

---

**Note:** If you define and start the Coherence cache server using the WebLogic Administration Console, you must indicate the location of the cache configuration file, and set distributed local storage and session local storage to true in the **Arguments** field of the **Server Start** tab. For example:

```
-Dtangosol.coherence.
cacheconfig=/WEB-INF/classes/session-cache-config.xml
-Dtangosol.coherence.distributed.
localstorage=true -Dtangosol.coherence.session.
localstorage=true
```

---

**Configure and Start the WebLogic Server**

To configure and start a Coherence cluster:

1. Run the Oracle WebLogic Configuration Wizard (**Start** then **All Programs** then **Oracle WebLogic** then **WebLogic Server 11gR1** then **Tools** then **Configuration Wizard**) to create a new WebLogic Server 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 WebLogic 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.

**Create a Machine**

To create a Machine on which to host WebLogic Server instances:

From the **Domain Structure** window, select **Environment** and then **Machines**. Click **New**. The **Create a New Machine** page displays. Enter a name for the Machine (in this case, **Test**) and click **Next**. Click Finish on the following page. **Figure 11–1** illustrates the **Create a New Machine** page.
Figure 11–1 Creating a New Machine

The Summary of Machines page should look similar to Figure 11–2.
Create the WebLogic Servers

Create two server instances associated with the Machine. The application will be deployed to these servers in a later step.

To create server instances:

1. Click the name of the Machine in the Summary of Machines page to open the Settings for machine page. Click the Servers tab and then click Add to create a server.

2. Select Create a new server and associate it with this machine in the Add a Server to Machine page, and click Next.

3. Provide details about the server in the Create a New Server page.

   Enter ServerA as the Server Name and 8081 as the Server Listen Port. Enter the appropriate value for the Server Listen Address. This is illustrated in Figure 11–3. Click Finish.
4. When you are returned to the Settings for machine page, click Add, and repeat the previous three steps to create a second server.

   Enter ServerB as the Server Name and 8082 as the Server Listen Port. Enter the appropriate value for the Server Listen Address. Click Finish.

5. Expand Environment in the Domain Structure menu and click Servers.

   The Summary of Servers page displays and is similar to Figure 11–4.
A Coherence cluster is a group of Coherence nodes that share a group address which allows them to communicate. Coherence clusters consist of nodes formed by applications, modules, or application servers (WebLogic Server instances or cache servers).

To create a Coherence Cluster:

1. Click Environment in the domain Structure Window and then click Coherence Clusters. In the Summary of Coherence Clusters page, click New. In the Coherence Cluster Properties page of the Create Coherence Cluster Configuration wizard, enter CoherenceCluster in the Name field, then click Next.

Figure 11–5 illustrates the Create Coherence Cluster Configuration page.
2. Enter a value such as 8085 in the Unicast Listen Port field. Do not change any of the other values and click Next.
3. In the Coherence Cluster Targets page of the Create Coherence Cluster Configuration wizard, select ServerA and ServerB as the targets. Click Finish.

**Figure 11–7 Choosing Coherence Cluster Targets**

The Summary of Coherence Clusters page looks similar to Figure 11–8.

**Figure 11–8 Summary of Coherence Clusters**

Coherence provides replicated and distributed data management and caching services that you can use to reliably make an application’s objects and data available to all servers in a Coherence cluster. To do this, WebLogic Server retains configuration information used to locate and communicate with a Coherence cluster.

This page displays the Coherence cluster configurations that have been created in this domain.
Deploy the Shared Library Files

In addition to the coherence.jar file, Coherence provides a deployable shared library, coherence-web-spi.war, that contains a native plug-in to WebLogic Server's HTTP Session Management interface. Coherence also provides the active-cache-1.0.jar file that contains the classes that enable WebLogic Server to interact with Coherence.

You do not have to deploy coherence.jar for this example. It will be bundled with the application in a later step.

To deploy the coherence-web-spi.war and active-cache-1.0.jar files:

1. From the Domain Structure menu, click Deployments. The Summary of Deployments page displays.
2. Click Install. The Install Application Assistant window displays.
3. Use the Install Application Assistant to deploy coherence-web-spi.war as a library to ServerA and ServerB.
   a. Locate and select the coherence-web-spi.war file as illustrated in Figure 11–9. The WAR file can be found in the coherence\lib folder of the Coherence installation. Click Next.
b. In the Choose targeting style page, ensure that Install this deployment as a library is selected, as illustrated in Figure 11–10. Click Next.
c. Select ServerA and ServerB as the deployment targets (do not deploy coherence-web-spi.war to AdminServer) as illustrated in Figure 11–11. Click Next.
d. In the Optional Settings page, select the Copy this application onto every target for me option in the Source accessibility section.

Figure 11–11 Selecting Deployment Targets

Install Application Assistant

Select deployment targets
Select the servers and/or clusters to which you want to deploy this application. (You can reconfigure deployment targets later).

Available targets for coherence-web-spi:

<table>
<thead>
<tr>
<th>Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminServer</td>
</tr>
<tr>
<td>ServerA</td>
</tr>
<tr>
<td>ServerB</td>
</tr>
</tbody>
</table>

Figure 11–12 Copying Files to Targets

Source accessibility

How should the source files be made accessible?

- Use the defaults defined by the deployment's targets
  Recommended selection.
- Copy this application onto every target for me
  During deployment, the files will be copied automatically to the managed servers to which the application is targeted.
- I will make the deployment accessible from the following location
  Location: C:\oracle\product\coherence\lib\coherence-web-spi.war

Provide the location from where all targets will access the application's files. This is often a shared directory. You must ensure the application files exist in this location and that each target can reach the location.
e. 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.

4. Repeat Steps 1 through 3 to deploy active-cache-1.0.jar to ServerA and ServerB (do not deploy active-cache-1.0.jar to the AdminServer).

When you reach the Optional Settings page, select the I will make the deployment accessible from the following location option. This will mimic the nostage WLST option.

Enter the path to the active-cache-1.0.jar file. The file is included in the WebLogic Server installation. Assuming that you installed the WebLogic Server at C:\oracle\product, you will find the file in the C:\oracle\product\wlserver_10.3\common\deployable-libraries folder.

**Figure 11–13 Summary of Deployments Page**

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 folder as follows:

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

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

3. Create a weblogic.xml file in the /WEB-INF folder.
   - Add a library reference for the coherence-web-spi.war file.
   - Reference the Coherence Cluster in a coherence-cluster-ref stanza.

   Example 11–2 illustrates a sample weblogic.xml file.

```
Example 11–2 Sample weblogic.xml File

<weblogic-web-app xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <library-ref>
        <library-name>coherence-web-spi</library-name>
    </library-ref>
    <coherence-cluster-ref>
        <coherence-cluster-name>CoherenceCluster</coherence-cluster-name>
    </coherence-cluster-ref>
</weblogic-web-app>
```

4. Bundle the coherence.jar file with the application: copy coherence.jar from the coherence\lib folder of the Coherence 3.7 installation to the WEB-INF/lib folder.

5. 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 folder.

```html
<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>
</html>
```

**Example 11–3 Sample manifest.mf File**

```manifest.mf
Extension-List: active-cache
active-cache-Extension-Name: active-cache
active-cache-Specification-Version: 1.0
active-cache-Implementation-Version: 1.0
```

7. The structure of the Web application folder should appear as follows:

```
/counter.jsp
/META-INF/manifest.mf
/WEB-INF/web.xml
/WEB-INF/weblogic.xml
/WEB-INF/lib/coherence.jar
```

8. ZIP or JAR the Web application folder 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 wizard opens.

3. Use the Install Application Assistant to deploy the counter.war file to ServerA and ServerB. In the Optional Settings page, select the Copy this application onto every target for me option in the Source accessibility section.

The Summary of Deployments page displays after the application is deployed. Figure 11–14 illustrates the page with the deployed active-cache.jar, coherence-web-spi.war, and counter.war files.
Start the Node Manager and the WebLogic Servers

Start the Node Manager then start the WebLogic Server instances from the WebLogic Server Administration Console. The Node Manager is a Java utility that runs as a separate process from Oracle WebLogic Server, and enables you to perform common operations for a Managed Server, regardless of its location with respect to its Administration Server.

1. To start the Node Manager, go to Start, then All Programs, then Oracle WebLogic, then WebLogic Server 11gR1, then Tools, and then Node Manager.

2. Click Environment then Servers in the domain Structure Window. From the Summary of Servers page in the WebLogic Server Administration Console, click the Control tab and start both server instances.

Figure 11–15 illustrates the deployments table after the servers have been started.

Start the Node Manager and the WebLogic Servers

Start the Node Manager then start the WebLogic Server instances from the WebLogic Server Administration Console. The Node Manager is a Java utility that runs as a separate process from Oracle WebLogic Server, and enables you to perform common operations for a Managed Server, regardless of its location with respect to its Administration Server.

1. To start the Node Manager, go to Start, then All Programs, then Oracle WebLogic, then WebLogic Server 11gR1, then Tools, and then Node Manager.

2. Click Environment then Servers in the domain Structure Window. From the Summary of Servers page in the WebLogic Server Administration Console, click the Control tab and start both server instances.

Figure 11–15 illustrates the deployments table after the servers have been started.
Verify the Example

To verify the example:

1. Open a browser and access the ServerA counter instance using the following URL:
   
   http://host:8081/counter/counter.jsp

   The counter page displays and the counter is set to 1 as illustrated in Figure 11–16.

   Figure 11–16  Counter Page with Counter Set to 1

2. In a new browser (or new browser tab), access the ServerB counter instance using the following URL:

   http://host:8082/counter/counter.jsp
The counter page displays and the counter increments to 2 based on the session data as illustrated in Figure 11–17.

Figure 11–17  Counter Page with Counter Set to 2

3. If you refresh the page, the counter increments to 3. Return to the original browser (or browser tab), refresh the instance, and the counter displays 4.
AbstractProcessor interface, 7-6
acceptor-config element, 10-12
ActiveCache, 11-1
active-cache-1.0.jar file, 11-10, 11-14
address element, 10-12
aggregate method, 6-3
aggregating cache data, 5-15
aggregating data, 5-15
AlwaysFilter class, 6-1
autostart element, 10-13

cache
  loading data, 5-1
  populating, 5-1
cache client, setting up, 1-2
cache server, setting up, 1-2
cache types, 9-1
cache types, described, 9-2
cache, creating with Eclipse, 9-1
cache-config.xml file, 8-1, 9-2, 9-4, 9-19
CacheFactory class, 3-2, 3-10
CacheLoader interface, 9-12
cache-mapping element, 9-2, 9-4
cache-name element, 9-19, 10-11, 10-12
CacheService interface, 10-28, 10-32
cache-service-proxy element, 10-32
CacheStore interface, 9-12
cachestore-scheme element, 9-12
caching complex objects, 4-2
 caching-scheme-mapping element, 9-2, 9-4
caching-schemes element, 9-2
ChainedExtractor class, 6-1
ChainedExtractor interface, 5-17
clustering, troubleshooting, 1-13
Coherence
  configuring, 1-1
  installing, 1-1
  restricting to your own host, 1-13
  testing the installation, 1-2
Coherence Query Language, 6-1
Coherence*Extend, 10-1
Coherence*Web, 11-1

cache-config.xml file, 1-8
cache.jar file, 11-10
cache-pof-config.xml file, 4-18
coherence-web-spi.war file, 11-10, 11-12
complex objects
  caching, 4-2
  creating, 4-2
configuring Coherence, 1-1
configuring Eclipse IDE, 2-1
createExtractor method, 6-1
createFilter method, 6-1
data grid, accessing from Java, 3-1
defaults element, 10-13
distributed-scheme element, 9-4

Eclipse IDE
  configuring, 2-1
Eclipse, installing, 2-1
EclipseLink, 8-1
EntryProcessor interface, 7-1, 7-5
EqualsFilter class, 6-1
Extend proxy service, 10-1

Filter class, 6-1
Filter interface, 5-16
filter package, 5-16, 9-16

identity-asserter element, 10-10
identity-transformer element, 10-10
IdentityTransformer interface, 10-5
init-param element, 4-19
installing Eclipse, 2-1
installing OEPE, 2-1
invocable objects, 10-37
InvocableMap interface, 7-5
InvocableMap.EntryAggregator interface, 5-22
Invocation Service, 10-37
invocation-service-proxy element, 10-43

J
Java Persistence API, 8-1
JAVA_HOME environment variable, 2-2
JPA, 8-1

K
KeyExtractor class, 6-1

L
LikeFilter class, 6-1
listeners, 7-2
loading data into the cache, 5-1

M
manifest.mf file, 11-16
MapEvent class, 7-1
MapListener interface, 7-1

N
NamedCache interface, 1-8, 3-1, 4-1, 10-19
network addresses, 1-2

O
Object Relational Mapping (ORM), 8-1
object-relational mapping, 8-1
ObservableMap interface, 7-1
OEPE, installing, 2-1
Oracle Database 10g Express Edition (OracleXE), 8-1

P
param-type element, 10-32, 10-43
persistence, 8-1
persistence.xml file, 8-20
PofPrincipal interface, 10-2
PofReader interface, 4-1
PofWriter interface, 4-1
populating the cache, 5-1
port element, 10-12
PortableObject interface, 4-1, 5-1, 7-6
proxy-config element, 10-32, 10-43

Q
QueryHelper class, 6-1
querying cache data, 5-15
querying data, 5-15
QueryMap interface, 5-15, 5-18

R
readExternal method, 4-1
read-write-backing-map scheme, 9-13
ReflectionExtractor class, 6-1
ReflectionExtractor interface, 5-17
remote-cache-scheme element, 10-11

S
security, access control, 10-1
security-config element, 10-10
serializer element, 10-13
subject-scope element, 10-10

T
tangosol.coherence.clusteraddress system property, 11-2
tangosol.coherence.clusterport system property, 11-2
tangosol.coherence.extend.enabled system property, 10-15
tangosol-coherence-override.xml, 10-10
tangosol-coherence.xml file, 9-7
tcp-initiator element, 10-11
troubleshooting clustering, 1-13
types of read and write caching, 9-13

U
user-type element, 4-18

V
ValueExtractor class, 6-1
ValueExtractor interface, 5-16

W
WebLogic Server, 11-1
web.xml file, 11-15
WHERE clause, 6-1
WrapperCacheService class, 10-28
WrapperInvocationService class, 10-39
WrapperNamedCache class, 10-19
writeExternal method, 4-1