

# The Definitive Guide to Terracotta

Cluster the JVM™ for Spring,  
Hibernate, and POJO Scalability



Terracotta, Inc.

## **The Definitive Guide to Terracotta: Cluster the JVM™ for Spring, Hibernate, and POJO Scalability**

**Copyright © 2008 by Terracotta, Inc.**

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-986-0

ISBN-10 (pbk): 1-59059-986-1

ISBN-13 (electronic): 978-1-4302-0640-8

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Java™ and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the US and other countries. Apress, Inc., is not affiliated with Sun Microsystems, Inc., and this book was written without endorsement from Sun Microsystems, Inc.

Lead Editor: Steve Anglin

Technical Reviewer: Jeff Genender

Development Editor: Matthew Moodie

Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell,

Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann,

Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Senior Project Manager: Tracy Brown Collins

Copy Editor: Heather Lang

Associate Production Director: Kari Brooks-Copony

Production Editors: Jill Ellis, Laura Cheu

Compositor: Gina Rexrode

Proofreader: Linda Seifert

Indexer: Toma Mulligan

Artist: April Milne

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com), or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail [info@apress.com](mailto:info@apress.com), or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>.

# Contents at a Glance

About Terracotta, Inc. . . . .	xiii
About the Authors . . . . .	xv
About the Technical Reviewer . . . . .	xvii
Introduction . . . . .	xix
<b>CHAPTER 1</b> Theory and Foundation: Forming a Common Understanding . . . . .	1
<b>CHAPTER 2</b> History of Terracotta . . . . .	23
<b>CHAPTER 3</b> Jumping Into Terracotta . . . . .	41
<b>CHAPTER 4</b> POJO Clustering . . . . .	69
<b>CHAPTER 5</b> Caching . . . . .	103
<b>CHAPTER 6</b> Hibernate with Terracotta . . . . .	131
<b>CHAPTER 7</b> Extending HTTP Sessions with Terracotta . . . . .	147
<b>CHAPTER 8</b> Clustering Spring . . . . .	177
<b>CHAPTER 9</b> Integration Modules . . . . .	203
<b>CHAPTER 10</b> Thread Coordination . . . . .	221
<b>CHAPTER 11</b> Grid Computing Using Terracotta . . . . .	243
<b>CHAPTER 12</b> Visualizing Applications . . . . .	293
<b>INDEX</b> . . . . .	323



# Contents

About Terracotta, Inc. . . . .	xiii
About the Authors . . . . .	xv
About the Technical Reviewer . . . . .	xvii
Introduction . . . . .	xix

<b>CHAPTER 1</b>	<b>Theory and Foundation: Forming a Common Understanding . . . . .</b>	<b>1</b>
	Definition of the Terracotta Framework . . . . .	2
	Network Attached Storage Similarities to Terracotta . . . . .	3
	Transparency of Memory Location . . . . .	6
	Putting Transparency and Clustering Together . . . . .	8
	Honoring the Java Memory Model . . . . .	11
	Being a Service Has Advantages . . . . .	12
	Availability . . . . .	12
	Scalability . . . . .	13
	Avoiding Bottlenecks . . . . .	14
	Use Cases . . . . .	17
	Distributed Caching . . . . .	17
	Database Offload . . . . .	18
	Session Replication . . . . .	18
	Workload Partitioning . . . . .	19
	Summary . . . . .	20

<b>CHAPTER 2</b>	<b>History of Terracotta . . . . .</b>	<b>23</b>
	Approaches to Scaling . . . . .	25
	Scale the Database . . . . .	25
	In-Memory Replication . . . . .	28
	All Paths Lead to Partitioning . . . . .	30
	Terracotta's Origin in a Web Monster . . . . .	31
	The Application Is a Giant Computer . . . . .	32

Terracotta Makes L2 General Purpose .....	33
Transparency Equals General Purpose .....	35
Transparent Scalability with Availability .....	36
Transparent Clustering Service Revisited .....	37
Transparency Delivers Higher Scalability .....	38
Clustered Application Servers and Clustered Caches .....	38
Summary .....	39
 <b>CHAPTER 3    Jumping Into Terracotta</b> .....	41
Hello Clustered World .....	41
Clustering HelloClusteredWorld .....	43
Why HelloClusteredWorld Works .....	46
Alternatives to Terracotta .....	49
A Tour Through Terracotta .....	50
Downloading and Installing Terracotta .....	51
Terracotta Installation Layout .....	51
Terracotta Scripts .....	52
Step-by-Step Hello Clustered World in Unix .....	53
Step-by-Step Hello Clustered World in Eclipse .....	63
Implications for Every Developer .....	65
Summary .....	66
 <b>CHAPTER 4    POJO Clustering</b> .....	69
All About Objects and Roots .....	69
Clustered Objects .....	71
Virtual Heap and Memory Management .....	71
Distributed Garbage Collection .....	72
Managing Object Changes and Coordinating Threads .....	72
Locks .....	73
Autolocks and Named Locks .....	74
Lock Level .....	75
Distributed wait() and notify() .....	76
Transactions .....	76
Distributed Method Invocation .....	76
Transparency and Bytecode Instrumentation .....	77
Portability .....	78
Boot JAR Classes .....	78

Physically vs. Logically Managed Objects . . . . .	78
Nonportable and Logically Managed Classes . . . . .	79
Portability Contexts . . . . .	80
Field Change . . . . .	80
Logical Action . . . . .	80
Object Graph Traversal . . . . .	80
Transient Fields and On-Load Behavior . . . . .	80
Initializing Transient Fields on Load . . . . .	81
Clustered POJOs Step by Step . . . . .	82
What Do We Mean by POJO? . . . . .	83
What's a Clustered POJO? . . . . .	83
Why Clustered POJOs? . . . . .	84
An Example of Clustered POJOs . . . . .	85
The Example Classes . . . . .	85
Clustering Requirements for the Example . . . . .	94
Configuration for the Example . . . . .	94
Running the Example . . . . .	95
What the Example Shows . . . . .	100
Summary . . . . .	102

## ■ CHAPTER 5    **Caching** . . . . . 103

The Pain of Caching . . . . .	103
The Pain of Large Data Sets . . . . .	103
The Pain of Staleness . . . . .	104
The Pain of Duplication . . . . .	104
How Terracotta Can Help . . . . .	105
A Simple Example of Caching . . . . .	107
Distributed Caching . . . . .	109
Transparent Distributed Caching . . . . .	109
Caching with Maps . . . . .	112
Unsorted Maps . . . . .	112
Sorted Maps . . . . .	112
Autolocked Collections . . . . .	113
Bulk Loading a Map . . . . .	115
Locking and Maps . . . . .	119
Maps of Maps . . . . .	119

A Deeper Look at Caching . . . . .	120
Eviction and Expiration . . . . .	120
Persistence . . . . .	121
Distributed Caching . . . . .	121
Partitioned Data . . . . .	121
Leveraging Terracotta . . . . .	122
Caching with Ehcache . . . . .	123
Rewriting the Cache Example . . . . .	123
Using Ehcache with Terracotta . . . . .	127
Summary . . . . .	129

## ■ CHAPTER 6    **Hibernate with Terracotta** . . . . . 131

Object-Relational Mapping . . . . .	131
How Does Hibernate Work? . . . . .	132
Typical Hibernate Invocation . . . . .	133
Using Terracotta to Improve Hibernate . . . . .	133
How Terracotta Improves Hibernate Performance . . . . .	133
Basic Steps for Improving Performance of a Hibernate Application . . . . .	134
Baseline Performance . . . . .	134
Add Hibernate to Save Three DB Calls . . . . .	135
Enable the Second-Level Cache to Save Five SQL Calls . . . . .	136
Clustering the Session Objects . . . . .	137
Benchmarking the Enhancements . . . . .	139
Configuring Hibernate to Work with Terracotta . . . . .	139
Configuring a Second-Level Cache . . . . .	140
Configuring Terracotta for Use with Detached Instances . . . . .	141
Debugging, Testing, and Tuning Concerns . . . . .	143
Verifying Ehcache Configuration Clustering . . . . .	144
Tuning the Second-Level Cache . . . . .	144
Tuning Detached Instances . . . . .	145
Summary . . . . .	145

## ■ CHAPTER 7    **Extending HTTP Sessions with Terracotta** . . . . . 147

What Are HTTP Sessions? . . . . .	147
Server-Side State Management . . . . .	148
Automatic Time-Out of Stale Data . . . . .	149
Manual Invalidation to Clear Out Data . . . . .	150
Events to Coordinate State Modification . . . . .	150
Session State Persistence . . . . .	151



Distributed HTTP Sessions . . . . .	151
Motivation for Using HTTP Sessions . . . . .	152
Alternatives to HTTP Sessions . . . . .	154
Distributed HTTP Sessions with Terracotta . . . . .	155
How Terracotta Sessions Work . . . . .	155
Benefits . . . . .	157
Using Terracotta and HTTP Sessions . . . . .	160
Shared Session State . . . . .	160
HTTP Sessions with Structured Session Attributes . . . . .	165
Understanding HTTP Sessions with Terracotta . . . . .	172
Supported Platforms . . . . .	173
Architecture Considerations . . . . .	173
Summary . . . . .	175
 <b>CHAPTER 8   Clustering Spring</b> . . . . .	 177
A Sample: Spring Beans Without a Database . . . . .	178
Running the Example on a Single JVM . . . . .	187
Running the Example Clustered with Terracotta . . . . .	190
A Note on DI and Terracotta . . . . .	193
Growing Spring Beans . . . . .	194
Clustering Spring Beans . . . . .	197
Running with Clustered Spring Beans . . . . .	200
Terracotta Session Clustering for Spring . . . . .	201
Summary . . . . .	201
 <b>CHAPTER 9   Integration Modules</b> . . . . .	 203
Clustering by Configuration . . . . .	203
Using TIMs . . . . .	203
Clustering a Synchronized Map . . . . .	203
Importing a TIM . . . . .	204
Module Repositories . . . . .	205
The Terracotta Forge . . . . .	206
Creating a TIM . . . . .	207
TIM Anatomy . . . . .	207
Creating a Simple TIM . . . . .	210
Creating TIMs with Maven . . . . .	211
Including Code in a TIM . . . . .	213
Summary . . . . .	220

<b>CHAPTER 10 Thread Coordination</b>	221
Terracotta and Thread Coordination	221
Clustering Threads	222
Thread Coordination Using Concurrency Primitives	222
Thread Coordination Using the <code>java.util.concurrent</code> Package	227
Thread Coordination and JMX Cluster Events	230
Thread Coordination with Queues	235
Summary	241
<b>CHAPTER 11 Grid Computing Using Terracotta</b>	243
What Are Grids?	243
Compute Grids vs. Data Grids	243
How Grids Handle Scalability	244
How Grids Handle Failover and High Availability	244
Use Cases	245
Introducing the Master/Worker Pattern	245
Master/Worker in Java	246
Using the Java Language's Thread Coordination Primitives	246
Using the <code>java.util.concurrent</code> Abstractions	247
Using the CommonJ Work Manager Specification	248
Getting Started: Naïve Single Work Queue Implementation	250
Implementing a Single-JVM Naïve Master/Worker Container	250
Clustering Our Master/Worker with Terracotta	257
Handling Real-World Challenges	258
Refactoring for Large Workloads, Routing, and Work Failover	259
Minimizing Contention and Maximizing Locality of Reference	259
Routing Strategies	268
Dealing with Work Failure and Recovery	270
Refactoring the Worker	271
Update the Terracotta Configuration	272
Using the Master/Worker System	272
Running the Master/Worker System	274
Refactoring Work Batching, Worker Failover, and	
Dynamic Worker Management	274
Work Batching	274
Dynamic Worker Management	275
Worker Failover	279
How to Use and Run the Fault-Tolerant Master/Worker System	281
Building a Distributed Web Spider	281
When Not to Use the Master/Worker Pattern	290
Summary	291

<b>CHAPTER 12 Visualizing Applications</b>	293
Observing an Application	293
Visualization Tools and Architecture	294
The Sample Inventory Application	295
Modifying to Generate Load	297
Taking Performance Snapshots	307
Potential Bottlenecks	310
Recalling the Motherboard	310
Inside the Terracotta Server	314
Looking for Locality Issues	316
Specific Bottlenecks in inventory.pounder	316
Visualizing the Bottlenecks	317
Managing Production Environments	320
Summary	321
<b>INDEX</b>	323



# About Terracotta, Inc.

Founded in 2003, **TERRACOTTA, INC.**, is a private firm headquartered in San Francisco, California. Terracotta delivers runtime plug-in capacity and availability for Java applications. Terracotta products simplify development, testing, deployment, and management of enterprise Java applications by adding clustering and distributed computing services to the JVM, allowing application development to focus on business features. Terracotta customers include leaders in a number of industries such as the packaged software development, financial services, telecommunications, travel, and online entertainment sectors.



# About the Authors

■ **ARI ZILKA** is the CTO of Terracotta, Inc. Ari spends most of his time on the product roadmap and customer use cases, as well as speaking at conferences. Ari has worked as an architecture lead for several large-scale systems, most notably as the chief architect at Walmart.com.

■ **GEERT BEVIN** is a senior developer at Terracotta, Inc. Geert spends most of his time developing and optimizing the JVM internal hooks and libraries that make Terracotta transparent, as well as being an invited speaker at the largest developer conferences worldwide. Geert has worked as creator and contributor on several popular open source development projects such as RIFE, OpenLaszlo, and Gentoo Linux in addition to consulting with very large companies on implementing and improving their Java-based web applications. His work on providing continuations natively to Java contributed to him being elected as an official Sun Java Champion.

■ **JONAS BONÉR** is a senior developer at Terracotta, Inc. Jonas spends most of his time developing large-scale financial systems and integrating Terracotta with various products and tools, as well as lecturing and speaking at developer conferences worldwide. He has worked on the Terracotta core and the JRockit JVM at BEA and is an active contributor to the open source community; most notably, he created the AspectWerkz aspect-oriented programming framework and has been part of the Eclipse AspectJ team.

■ **TAYLOR GAUTIER** is a product manager at Terracotta, Inc. Taylor spends most of his time designing and using Terracotta's products. He is a converted, yet still devout, developer who has helped write parts of systems-level servers and technology at companies like Scale8 and Reactivity.

■ **ORION LETIZI** is a cofounding engineer at Terracotta, Inc. Orion spends most of his time designing and documenting clear use cases and guides for using Terracotta. He, of course, helped write the first two versions of the product but has since transitioned to helping others maximize their use of the technology. Orion has worked as a lead at several of the most popular e-commerce web sites and content portals on the Internet including Walmart.com.

■ **ALEX MILLER** is a technical lead at Terracotta, Inc. Alex leads and manages the transparency team at Terracotta. His focus is on transparently integrating cluster support into a user's application, the JDK, and third-party libraries. The transparency and server teams work together to optimize scaling and minimize the costs of clustered applications. Alex previously served as Chief Architect at MetaMatrix and worked on enterprise products at BEA. Alex speaks at conferences and blogs at <http://tech.puredanger.com>.





# About the Technical Reviewer

■ **JEFF GENENDER** is the CTO of Savoir Technologies, Inc., a Java enterprise software consultancy. Jeff is an open source evangelist and Apache member who is an active committer on several Apache projects and a committer for Terracotta. He is the author of three books and serves as a member of the Java Community Process (JCP) expert group for JSR-316 (the Java Platform, Enterprise Edition 6 specification).

Jeff has successfully brought open source development efforts, initiatives, and success stories into a number of Global 2000 companies, saving these organizations millions in licensing costs.



# Introduction

Imagine being able to call `wait()` on a Java object in one JVM, and imagine that later a thread in another JVM on another computer calls `notify()` on that same object and that `notify()` call penetrates the impermeable process boundary between the two JVMs to wake your thread from its slumber. Further imagine that all of the changes to that object made in the other JVM while you were waiting are now visible to your thread. Imagine now that the code you wrote to make this happen looked *no different* than code you would have written were this program to run only on a single JVM—no concessions to special frameworks, no callouts to special APIs, no stubs, no skeletons, no copies, no put-backs, and no magic beans.

What would it be like if you could automatically share Java heap between virtual machines so that threads running in different JVMs could interact with each other exactly as if they were running in the same JVM? Imagine what you could do with that kind of power—the power to express a computing problem in the simplest possible terms that stay true to the problem at hand yet be able to run that program on multiple computers at once while the program thinks it’s running on one big computer. What would such a thing be? And wouldn’t you really, really want it if you could have it?

Terracotta was born out of just such imaginings.

Terracotta is Java infrastructure software that allows you to scale your application to as many computers as needed, without expensive custom code or databases. Throughout this book, it is referred to as JVM-level clustering or network-attached memory. The question you now face is how to understand and leverage this technology best.

The goal of this book is to help you answer that question. While a total of almost fifty person-years of engineering have gone into the product, to date very little has been documented about Terracotta. After years of underground lab work and hundreds of production deployments under its belt, Terracotta’s team of authors have come together to write this definitive guide to Terracotta. This book will help you learn the official definition and detailed architecture of the technology. It will also walk you through the most popular applications of JVM-level clustering. It will teach you how to tune and manage a production application running in a Terracotta environment.

Use this guide as a gateway. Terracotta is infrastructure just like a database, a file server, or even a networking switch. With it you can achieve many things and design many different use cases. Any technology that claims it is well suited to all use cases should be viewed with caution. We have put forth this work to help you both understand and trust Terracotta for certain use cases. While file servers and relational databases have been used with great success over the past 30 years, they have also been abused with varying levels of success (for example, we have seen a single relational database handle 1.5 billion inserts a day, filling it with data that was retained for only 30 days, all achieved by disabling rollback). We realize that databases succeed because a community of developers rallied around the concept and built a shared toolset and understanding of where databases fit and where they do not.

The Java community acts as a gateway to this understanding of database technology. Together, the community members help each other design systems that are generally accepted as successful. Terracotta has invented new nomenclature—calling itself network-attached memory. As a result, the community needs a foothold with which to establish a shared knowledge of this new thing, this network-attached memory. This book is your gateway into the universe of network-attached memory and the power it represents. Use it to help begin the conversation and thought process around what are good and bad use cases for a JVM-level clustering approach.

I think an example is in order. (By the way, we communicate mostly by example in this book. Hopefully, this style of teaching will be helpful to you.) Engineer A wants to use Terracotta to act as a point-to-point communication protocol on the Internet to stream movies to web browsers. Meanwhile, Engineer B wants to use Terracotta to maintain a centralized or shared view of her application configuration data without having to maintain each JVM's system properties or XML-based configuration separately. Engineer C wants to use Terracotta to replace his web application's current stateless architecture for a more natural stateful one. The product will integrate to each engineer's application with minimal initial effort, so how is engineer A, B, or C to know what to expect? For engineer A, Terracotta will seem unstable, slow, and generally cumbersome, while for engineers B and C, Terracotta will seem almost ideal.

If you can tell the differences among these application patterns, consider yourself among the few. If you are currently wrestling with an application running on Terracotta or are contemplating where to use it and where to not use it, this guide represents the foremost thinking on the subject.

Here, you will learn about several popular cases for using Terracotta. You will learn in detail how to configure the system in each case, and you will learn the details of what makes that case a potential success or sweet spot.

By the end of this book, you will see the patterns for yourself and be able to help your friends understand the differences too. It's an exciting time for those of us at Terracotta. The concepts we are discussing and suggesting to you will not prove new or radical. The concepts embodied within the product—centralized management, stateful development, stateless runtime, and more—have never been brought together or assembled into a package like this. This assembly of approaches and techniques is what makes Terracotta exciting. Communicating the new capabilities this technology provides to the application development team is challenging yet rewarding. We hope you find the book as engaging to read as we found it to write. With your help, this will be the first of several books on the topics of Terracotta, network-attached memory, and JVM-level clustering.

## How This Book Is Structured

This book is structured as follows:

- *Chapter 1, Theory and Foundation:* We start with a look at what Terracotta is and the theory behind it.
- *Chapter 2, History of Terracotta:* This chapter provides a brief history of Terracotta to give the rest of the book some context.
- *Chapter 3, Jumping Into Terracotta:* In this chapter, we get into the practicalities of Terracotta, introducing the basic concepts of its configuration and showing a simple example of Terracotta clustering.

- *Chapter 4, POJO Clustering*: This practical chapter covers how to cluster an existing POJO application.
- *Chapter 5, Caching*: This chapter explains how to use Terracotta to alleviate some of the problems inherent in caching and how to bring the benefits of caching to distributed systems.
- *Chapter 6, Hibernate with Terracotta*: In this chapter, we show you how to scale Hibernate-enabled applications. We cover the concepts on which Hibernate is built so that you gain a full understanding of clustering with Hibernate.
- *Chapter 7, Extending HTTP Sessions with Terracotta*: You'll learn how to use Terracotta to cluster HTTP sessions in your web application in this chapter.
- *Chapter 8, Clustering Spring*: This chapter takes you through the process of turning a non-Spring clustered application into a Spring-enabled clustered application. It shows you how you can integrate Terracotta and Spring in a simple and effective fashion.
- *Chapter 9, Integration Modules*: We show you how to use Terracotta Integration Modules to cluster an external library or framework or modularize the configuration of your own application in this chapter.
- *Chapter 10, Thread Coordination*: In this chapter, we explain how to deal with threads and concurrency in Terracotta. We take a cookbook approach to give you a number of use case models that you can apply to your own applications.
- *Chapter 11, Grid Computing Using Terracotta*: This chapter showcases Terracotta's abilities to act as the platform for your own grid computing applications.
- *Chapter 12, Visualizing Applications*: The final chapter describes how to tune your Terracotta applications using Terracotta's visualization tools.

## Prerequisites

To run the code in this book, you need to install the Java Software Development Kit 1.4 or greater.

## Contacting the Authors

For more information about Terracotta, visit the web site at <http://www.terracotta.org>. There, you can find forums (<http://forums.terracotta.org>) and mailing lists dedicated to Terracotta, where members of the author team monitor and respond to queries from Terracotta users.

