Introduction to Terracotta

Cris J. Holdorph
Software Architect
Unicon, Inc.

JASIG Conference
Dallas, TX
March 2, 2009

UNICON

Agenda

1. Introduction
2. Main Parts
3. Terracotta Integration Modules
4. Hello World
5. Lessons Learned using Terracotta
6. Different Uses for Terracotta
7. Resources

Introduction

Introduction

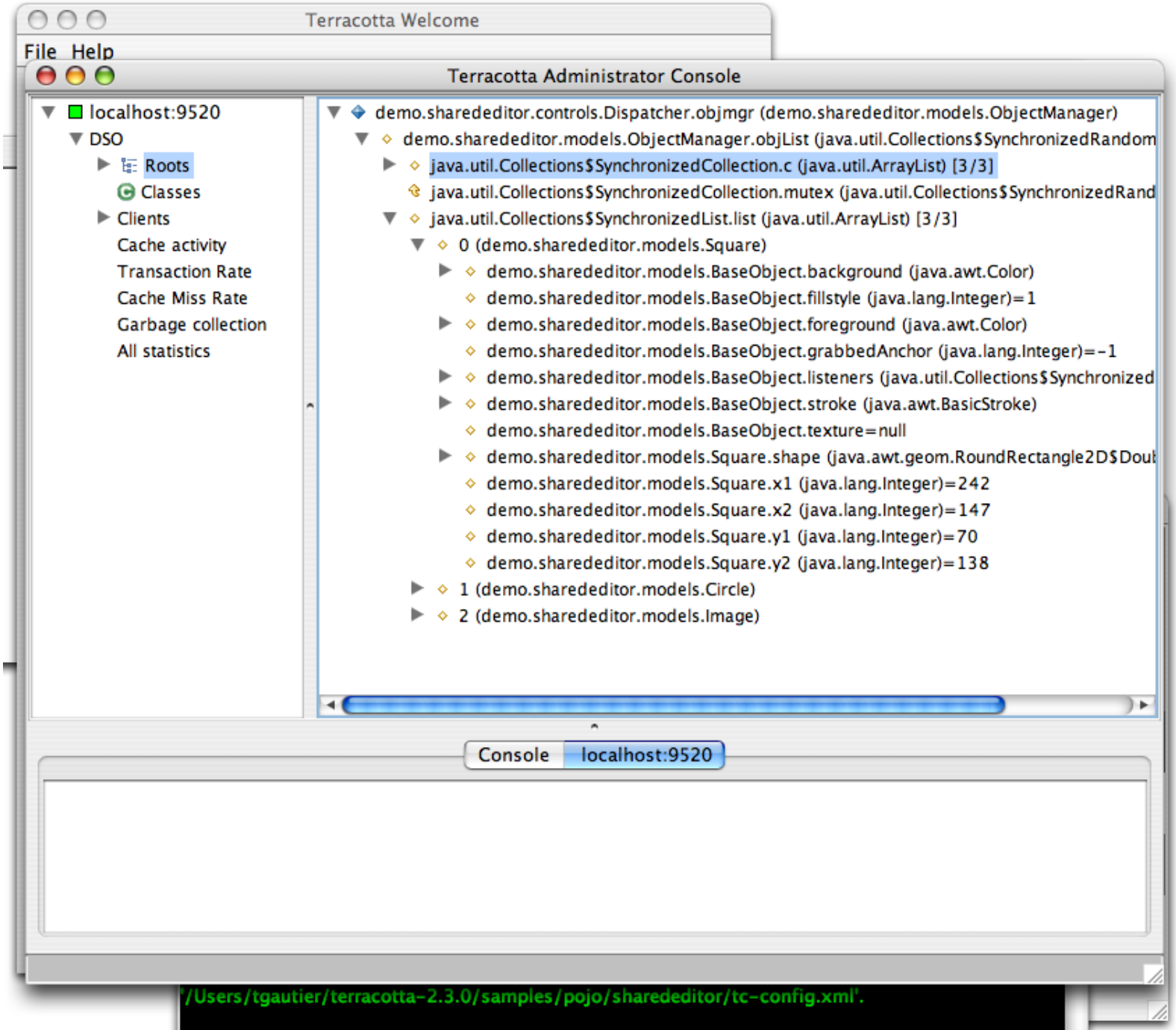
- Terracotta is...
  - A Distributed Cache
  - A Distributed Session Server
  - Network Attached Memory

- Terracotta is...
  - Open Source
  - Java
  - Client / Server

Main Parts

Main Parts

- Terracotta Server
  - tc-config.xml
- Terracotta Enabled Application
  - tc-config.xml
  - JAVA_OPTS
- Terracotta Administration Console


Terracotta Welcome
File Help
Terracotta Administrator Console
localhost:9520
DSO
Roots
Classes
Clients
Cache activity
Transaction Rate
Cache Miss Rate
Garbage collection
All statistics
demo.sharededitor.controls.Dispatcher.objmgr (demo.sharededitor.models.ObjectManager)
demo.sharededitor.models.ObjectManager.objList (java.util.Collections$SynchronizedRandom
java.util.Collections$SynchronizedCollection.c (java.util.ArrayList) [3/3]
java.util.Collections$SynchronizedCollection.mutex (java.util.Collections$SynchronizedRand
java.util.Collections$SynchronizedList.list (java.util.ArrayList) [3/3]
0 (demo.sharededitor.models.Square)
demo.sharededitor.models.BaseObject.background (java.awt.Color)
demo.sharededitor.models.BaseObject.fillstyle (java.lang.Integer)=1
demo.sharededitor.models.BaseObject.foreground (java.awt.Color)
demo.sharededitor.models.BaseObject.grabbedAnchor (java.lang.Integer)=-1
demo.sharededitor.models.BaseObject.listeners (java.util.Collections$Synchronized
demo.sharededitor.models.BaseObject.stroke (java.awt.BasicStroke)
demo.sharededitor.models.BaseObject.texture=null
demo.sharededitor.models.Square.shape (java.awt.geom.RoundRectangle2D$Doub
demo.sharededitor.models.Square.x1 (java.lang.Integer)=242
demo.sharededitor.models.Square.x2 (java.lang.Integer)=147
demo.sharededitor.models.Square.y1 (java.lang.Integer)=70
demo.sharededitor.models.Square.y2 (java.lang.Integer)=138
1 (demo.sharededitor.models.Circle)
2 (demo.sharededitor.models.Image)
Console
localhost:9520
'/Users/tgautier/terracotta-2.3.0/samples/pojo/sharededitor/tc-config.xml'.

tc-config.xml

- Roots
- Locks
- Instrumented Classes
- Transient Fields
  - Marking
  - Resolving

Configuring Roots in tc-config.xml

```
<roots>
  <root>
    <field-name>org.pkg.MyClass.aField</field-name>
  </root>
  <root>
    <field-name>org.pkg.MyClass.field2</field-name>
  </root>
</roots>
```

Terracotta Locks

- Four types of locks
  - One
  - Two
  - Three
  - Four
- All changes to a Terracotta managed object must be done in the scope of a lock
- You do not need a lock on the object you are changing
- Locks are the *Transactions* of Terracotta

Configuring Locks in tc-config.xml

```
<locks>
  <autolock auto-synchronized="false">
    <method-expression>*
org.pkg.MyClass.method(..)</method-expression>
    <lock-level>write</lock-level>
  </autolock>
  <autolock auto-synchronized="false">
    <method-expression>* org.pkg.SomeClass$Inny.run(..)</
method-expression>
    <lock-level>write</lock-level>
  </autolock>
</locks>
```

Instrumented Classes

- Each instrumented class can or must configure the following section
  - class-expression (required)
  - honor-transient (optional)
  - on-load (optional)
- Optional instrumented-classes section
  - exclude (optional)

Configuring Instrumented Classes in tc-config.xml

```
<instrumented-classes>
  <include>
      <class-expression>org.pkg.MyClass</class-expression>
      <honor-transient>true</honor-transient>
      <on-load><execute><![CDATA[self.log =
org.apache.commons.logging.LogFactory.getLog(org.pkg.MyClass.class);
]]></execute></on-load>
  </include>
  <include>
      <class-expression>org.pkg.YourClass</class-expression>
      <honor-transient>true</honor-transient>
      <on-load>
        <method>resolveTransientFields</method>
      </on-load>
  </include>
</instrumented-classes>
```

Resolving Transient Fields

- Transient Fields may be specified 2 ways
  - Transient keyword in the Java source file and corresponding property to honor transient keyword in tc-config.xml
  - Explicitly declaring a field as transient in the tc-config.xml file
- Transient Fields may be resolved in 2 ways
  - BeanShell script in tc-config.xml
  - Calling a method on the resolved object

Resolving a Transient Field with BeanShell

```
<instrumented-classes>
  <include>
      <class-expression>org.pkg.MyClass</class-expression>
      <honor-transient>true</honor-transient>
      <on-load>
        <execute><![CDATA[self.log =
org.apache.commons.logging.LogFactory.getLog(org.pkg.MyClass.cla
ss);]]></execute>
      </on-load>
  </include>
</instrumented-classes>
```

Resolving a Transient Field with Java

```
<instrumented-classes>
  <include>
      <class-expression>org.pkg.YourClass</class-expression>
      <honor-transient>true</honor-transient>
      <on-load>
        <method>resolveTransientFields</method>
      </on-load>
  </include>
</instrumented-classes>
```

Terracotta Integration Modules

Terracotta Integration Modules

- TIM (Terracotta Integration Module)
  - OSGi bundle
  - Maven artifact

- Use to decompose your tc-config.xml
- Let different application components manage their own tc-config.xml configuration bits
- Maven support for building a TIM

Using a TIM in tc-config.xml

```
<clients>
   <logs>/var/logs/terracotta/client/</logs>
   <statistics>/var/logs/terracotta/clientstats</statistics>
   <modules>
      <repository>/opt/local/tc-repo/</repository>
      <module name="tim-mytim" version="1.2.3" group-
id="org.pkg"/>
      <module name="tim-yourtim" version="2.0.7" group-
id="org.pkg"/>
   </modules>
</clients>
```

Creating a TIM with Maven

```
<plugins>
  <plugin>
    <groupId>org.terracotta.maven.plugins</groupId>
    <artifactId>tc-maven-plugin</artifactId>
  </plugin>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
  </plugin>
</plugins>
```

For a complete example look at the following pom.xml file
https://source.sakaiproject.org/svn/kernel/trunk/kernel-tim/pom.xml

Hello World

(in Terracotta)

Code Sample

```
Insert Code Here
```

Lessons Learned

Lessons Learned

- Serializable, Transient, Synchronization
- Course Grained Locking versus Fine Grained Locking
- Cluster-wide Locking
- TIMs

Lessons Learned

- Partitioning your data
  - Avoiding iterating over all objects in a Root collection
- Minor field updates vs Full object replacements
- Instrumenting ALL classes that access shared objects
- Avoid direct field access (use setters instead)
- Avoid non-static inner classes (they require storage of the outer class)

Different Uses for Terracotta

Different Uses for Terracotta

- Clustering your application for server failover
  - Transparent session migration
- Hibernate second level cache mirroring
- General purpose cache mirroring
- Pseudo Object Database

Resources

Resources

- *Cluster-enabling Sakai with Terracotta*
  - Tuesday, 10:15 am
- Websites
  - Open Source http://www.terracotta.org/
  - Commercial http://www.terracottatech.com/
- Website Resources
  - Downloads
  - Documentation
  - Forums
- Book
  - *The Definitive Guide to Terracotta*, ISBN-13: 978-1590599860

Questions & Answers

Cris J. Holdorph
Software Architect
Unicon, Inc.

holdorph@unicon.net
www.unicon.net