How does the bytecode enhancement dirty checking mechanism work in Hibernate 4.3

SEPTEMBER 8, 2014 / VLADMIHALCEA

(Last Updated On: January 29, 2018)

Introduction

Now that you know the basics of Hibernate dirty checking, we can dig into enhanced dirty checking mechanisms. While the default graph-traversal algorithm might be sufficient for most use-cases, there might be times when you need an optimized dirty checking algorithm and instrumentation is much more convenient than building your own custom strategy.

Using Ant Hibernate Tools

Traditionally, The Hibernate Tools have been focused on Ant and Eclipse. Bytecode instrumentation has been possible since Hibernate 3, but it required an Ant task to run the CGLIB or Javassist bytecode enhancement routines.

Maven supports running Ant tasks through the maven-antrun-plugin:

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-antrun-plugin</artifactId>
            <executions>
                <execution>
                    <id>Instrument domain classes</id>
                    <configuration>
                        <tasks>
                            <taskdef name="instrument"
                                     classname="org.hibernate.tool.instrument.javassist.Instrum
                                <classpath>
                                    <path refid="maven.dependency.classpath"/>
                                    <path refid="maven.plugin.classpath"/>
                                </classpath>
                            </taskdef>
                            <instrument verbose="true">
                                <fileset dir="${project.build.outputDirectory}">
                                    <include name="**/flushing/*.class"/>
                                </fileset>
                            </instrument>
                        </tasks>
                    </configuration>
                    <phase>process-classes</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                </execution>
            </executions>
            <dependencies>
                <dependency>
                    <groupId>org.hibernate</groupId>
                    <artifactId>hibernate-core</artifactId>
                    <version>${hibernate.version}</version>
                </dependency>
                <dependency>
```

```
                    <groupId>org.javassist</groupId>
                    <artifactId>javassist</artifactId>
                    <version>${javassist.version}</version>
                </dependency>
            </dependencies>
        </plugin>
    </plugins>
</build>
```

So for the following entity source class:

```
@Entity
public class EnhancedOrderLine {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private Long number;

    private String orderedBy;

    private Date orderedOn;

    public Long getId() {
        return id;
    }

    public Long getNumber() {
        return number;
    }

    public void setNumber(Long number) {
        this.number = number;
    }

    public String getOrderedBy() {
        return orderedBy;
    }

    public void setOrderedBy(String orderedBy) {
        this.orderedBy = orderedBy;
```

```
    }

    public Date getOrderedOn() {
        return orderedOn;
    }

    public void setOrderedOn(Date orderedOn) {
        this.orderedOn = orderedOn;
    }
}
```

During build-time the following class is generated:

```
@Entity
public class EnhancedOrderLine implements FieldHandled {

  @Id
  @GeneratedValue(strategy=GenerationType.AUTO)
  private Long id;
  private Long number;
  private String orderedBy;
  private Date orderedOn;
  private transient FieldHandler $JAVASSIST_READ_WRITE_HANDLER;

  public Long getId() {
    return $javassist_read_id();
  }

  public Long getNumber() {
    return $javassist_read_number();
  }

  public void setNumber(Long number) {
    $javassist_write_number(number);
  }

  public String getOrderedBy() {
    return $javassist_read_orderedBy();
  }

  public void setOrderedBy(String orderedBy) {
    $javassist_write_orderedBy(orderedBy);
  }
```

```
31
32    public Date getOrderedOn() {
33      return $javassist_read_orderedOn();
34    }
35
36    public void setOrderedOn(Date orderedOn) {
37      $javassist_write_orderedOn(orderedOn);
38    }
39
40    public FieldHandler getFieldHandler() {
41      return this.$JAVASSIST_READ_WRITE_HANDLER;
42    }
43
44    public void setFieldHandler(FieldHandler paramFieldHandler) {
45      this.$JAVASSIST_READ_WRITE_HANDLER = paramFieldHandler;
46    }
47
48    public Long $javassist_read_id() {
49      if (getFieldHandler() == null)
50        return this.id;
51    }
52
53    public void $javassist_write_id(Long paramLong) {
54      if (getFieldHandler() == null) {
55        this.id = paramLong;
56        return;
57      }
58      this.id = ((Long)getFieldHandler().writeObject(this, "id", this.id, paramLong));
59    }
60
61    public Long $javassist_read_number() {
62      if (getFieldHandler() == null)
63        return this.number;
64    }
65
66    public void $javassist_write_number(Long paramLong) {
67      if (getFieldHandler() == null) {
68        this.number = paramLong;
69        return;
70      }
71      this.number = ((Long)getFieldHandler().writeObject(this, "number", this.number, paramLong))
72    }
73
```

```
74      public String $javassist_read_orderedBy() {
75        if (getFieldHandler() == null)
76          return this.orderedBy;
77      }
78
79      public void $javassist_write_orderedBy(String paramString) {
80        if (getFieldHandler() == null) {
81          this.orderedBy = paramString;
82          return;
83        }
84        this.orderedBy = ((String)getFieldHandler().writeObject(this, "orderedBy", this.orderedBy,
85      }
86
87      public Date $javassist_read_orderedOn() {
88        if (getFieldHandler() == null)
89          return this.orderedOn;
90      }
91
92      public void $javassist_write_orderedOn(Date paramDate) {
93        if (getFieldHandler() == null) {
94          this.orderedOn = paramDate;
95          return;
96        }
97        this.orderedOn = ((Date)getFieldHandler().writeObject(this, "orderedOn", this.orderedOn, pa
98      }
99  }
```

Although the org.hibernate.bytecode.instrumentation.spi.AbstractFieldInterceptor manages to intercept dirty fields, this info is never really enquired during dirtiness tracking.

The InstrumentTask bytecode enhancement can only tell whether an entity is dirty, lacking support for indicating which properties have been modified, therefore making the InstrumentTask more suitable for "No-proxy" LAZY fetching strategy.

hibernate-enhance-maven-plugin

Hibernate 4.2.8 added support for a dedicated Maven bytecode enhancement plugin.

The Maven bytecode enhancement plugin is easy to configure:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.hibernate.orm.tooling</groupId>
            <artifactId>hibernate-enhance-maven-plugin</artifactId>
            <executions>
                 <execution>
                     <phase>compile</phase>
                     <goals>
                         <goal>enhance</goal>
                     </goals>
                 </execution>
             </executions>
        </plugin>
    </plugins>
</build>
```

During project build-time, the following class is being generated:

```
@Entity
public class EnhancedOrderLine
        implements ManagedEntity, PersistentAttributeInterceptable, SelfDirtinessTracker {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private Long number;
    private String orderedBy;
    private Date orderedOn;

    @Transient
    private transient PersistentAttributeInterceptor $$_hibernate_attributeInterceptor;

    @Transient
    private transient Set $$_hibernate_tracker;

    @Transient
    private transient CollectionTracker $$_hibernate_collectionTracker;

    @Transient
```

```
22      private transient EntityEntry $$_hibernate_entityEntryHolder;
23
24      @Transient
25      private transient ManagedEntity $$_hibernate_previousManagedEntity;
26
27      @Transient
28      private transient ManagedEntity $$_hibernate_nextManagedEntity;
29
30      public Long getId() {
31          return $$_hibernate_read_id();
32      }
33
34      public Long getNumber() {
35          return $$_hibernate_read_number();
36      }
37
38      public void setNumber(Long number) {
39          $$_hibernate_write_number(number);
40      }
41
42      public String getOrderedBy() {
43          return $$_hibernate_read_orderedBy();
44      }
45
46      public void setOrderedBy(String orderedBy) {
47          $$_hibernate_write_orderedBy(orderedBy);
48      }
49
50      public Date getOrderedOn() {
51          return $$_hibernate_read_orderedOn();
52      }
53
54      public void setOrderedOn(Date orderedOn) {
55          $$_hibernate_write_orderedOn(orderedOn);
56      }
57
58      public PersistentAttributeInterceptor $$_hibernate_getInterceptor() {
59          return this.$$_hibernate_attributeInterceptor;
60      }
61
62      public void $$_hibernate_setInterceptor(PersistentAttributeInterceptor paramPersistentAttr
63          this.$$_hibernate_attributeInterceptor = paramPersistentAttributeInterceptor;
64      }
```

```
65
66      public void $$_hibernate_trackChange(String paramString) {
67          if (this.$$_hibernate_tracker == null)
68              this.$$_hibernate_tracker = new HashSet();
69          if (!this.$$_hibernate_tracker.contains(paramString))
70              this.$$_hibernate_tracker.add(paramString);
71      }
72
73      private boolean $$_hibernate_areCollectionFieldsDirty() {
74          return ($$_hibernate_getInterceptor() != null) && (this.$$_hibernate_collectionTracker
75      }
76
77      private void $$_hibernate_getCollectionFieldDirtyNames(Set paramSet) {
78          if (this.$$_hibernate_collectionTracker == null)
79              return;
80      }
81
82      public boolean $$_hibernate_hasDirtyAttributes() {
83          return ((this.$$_hibernate_tracker == null) || (this.$$_hibernate_tracker.isEmpty()))
84      }
85
86      private void $$_hibernate_clearDirtyCollectionNames() {
87          if (this.$$_hibernate_collectionTracker == null)
88              this.$$_hibernate_collectionTracker = new CollectionTracker();
89      }
90
91      public void $$_hibernate_clearDirtyAttributes() {
92          if (this.$$_hibernate_tracker != null)
93              this.$$_hibernate_tracker.clear();
94          $$_hibernate_clearDirtyCollectionNames();
95      }
96
97      public Set<String> $$_hibernate_getDirtyAttributes() {
98          if (this.$$_hibernate_tracker == null)
99              this.$$_hibernate_tracker = new HashSet();
100         $$_hibernate_getCollectionFieldDirtyNames(this.$$_hibernate_tracker);
101         return this.$$_hibernate_tracker;
102     }
103
104     private Long $$_hibernate_read_id() {
105         if ($$_hibernate_getInterceptor() != null)
106             this.id = ((Long) $$_hibernate_getInterceptor().readObject(this, "id", this.id));
107         return this.id;
```

```
108     }
109
110     private void $$_hibernate_write_id(Long paramLong) {
111         if (($$_hibernate_getInterceptor() == null) || ((this.id == null) || (this.id.equals(p
112             break label39;
113         $$_hibernate_trackChange("id");
114         label39:
115         Long localLong = paramLong;
116         if ($$_hibernate_getInterceptor() != null)
117             localLong = (Long) $$_hibernate_getInterceptor().writeObject(this, "id", this.id,
118         this.id = localLong;
119     }
120
121     private Long $$_hibernate_read_number() {
122         if ($$_hibernate_getInterceptor() != null)
123             this.number = ((Long) $$_hibernate_getInterceptor().readObject(this, "number", thi
124         return this.number;
125     }
126
127     private void $$_hibernate_write_number(Long paramLong) {
128         if (($$_hibernate_getInterceptor() == null) || ((this.number == null) || (this.number.
129             break label39;
130         $$_hibernate_trackChange("number");
131         label39:
132         Long localLong = paramLong;
133         if ($$_hibernate_getInterceptor() != null)
134             localLong = (Long) $$_hibernate_getInterceptor().writeObject(this, "number", this.
135         this.number = localLong;
136     }
137
138     private String $$_hibernate_read_orderedBy() {
139         if ($$_hibernate_getInterceptor() != null)
140             this.orderedBy = ((String) $$_hibernate_getInterceptor().readObject(this, "ordered
141         return this.orderedBy;
142     }
143
144     private void $$_hibernate_write_orderedBy(String paramString) {
145         if (($$_hibernate_getInterceptor() == null) || ((this.orderedBy == null) || (this.orde
146             break label39;
147         $$_hibernate_trackChange("orderedBy");
148         label39:
149         String str = paramString;
150         if ($$_hibernate_getInterceptor() != null)
```

```
151             str = (String) $$_hibernate_getInterceptor().writeObject(this, "orderedBy", this.o
152         this.orderedBy = str;
153     }
154
155     private Date $$_hibernate_read_orderedOn() {
156         if ($$_hibernate_getInterceptor() != null)
157             this.orderedOn = ((Date) $$_hibernate_getInterceptor().readObject(this, "orderedOn
158         return this.orderedOn;
159     }
160
161     private void $$_hibernate_write_orderedOn(Date paramDate) {
162         if (($$_hibernate_getInterceptor() == null) || ((this.orderedOn == null) || (this.orde
163             break label39;
164         $$_hibernate_trackChange("orderedOn");
165         label39:
166         Date localDate = paramDate;
167         if ($$_hibernate_getInterceptor() != null)
168             localDate = (Date) $$_hibernate_getInterceptor().writeObject(this, "orderedOn", th
169         this.orderedOn = localDate;
170     }
171
172     public Object $$_hibernate_getEntityInstance() {
173         return this;
174     }
175
176     public EntityEntry $$_hibernate_getEntityEntry() {
177         return this.$$_hibernate_entityEntryHolder;
178     }
179
180     public void $$_hibernate_setEntityEntry(EntityEntry paramEntityEntry) {
181         this.$$_hibernate_entityEntryHolder = paramEntityEntry;
182     }
183
184     public ManagedEntity $$_hibernate_getPreviousManagedEntity() {
185         return this.$$_hibernate_previousManagedEntity;
186     }
187
188     public void $$_hibernate_setPreviousManagedEntity(ManagedEntity paramManagedEntity) {
189         this.$$_hibernate_previousManagedEntity = paramManagedEntity;
190     }
191
192     public ManagedEntity $$_hibernate_getNextManagedEntity() {
193         return this.$$_hibernate_nextManagedEntity;
```

```
194         }
195
196         public void $$_hibernate_setNextManagedEntity(ManagedEntity paramManagedEntity) {
197             this.$$_hibernate_nextManagedEntity = paramManagedEntity;
198         }
199     }
```

It's easy to realize that the new bytecode enhancement logic is different than the one generated by the previous InstrumentTask.

Like the custom dirty checking mechanism, the new bytecode enhancement version records what properties have changed, not just a simple dirty boolean flag. The enhancement logic marks dirty fields upon changing. This approach is much more efficient than having to compare all current property values against the load-time snapshot data.

Consider upgrading to Hibernate 5

Even if the entity class bytecode is being enhanced, somehow with Hibernate 4.3.6 there are still missing puzzle pieces.

For instance, when calling **setNumber(Long number)** the following intercepting method gets executed:

```
private void $$_hibernate_write_number(Long paramLong) {
    if (($$_hibernate_getInterceptor() == null) || ((this.number == null) || (this.number.equal
        break label39;
    $$_hibernate_trackChange("number");
    label39:
    Long localLong = paramLong;
    if ($$_hibernate_getInterceptor() != null)
        localLong = (Long) $$_hibernate_getInterceptor().writeObject(this, "number", this.numbe
    this.number = localLong;
}
```

In my examples, $$_hibernate_getInterceptor() is always null, which bypasses the **$$_hibernate_trackChange("number")** call. Because of this, no dirty property is going to be recorded, forcing Hibernate to fall-back to the default deep-comparison dirty checking algorithm.

So, even if Hibernate has made considerable progress in this particular area, the dirty checking enhancement still requires additional work to become readily available.

Code available on GitHub.

- **3 chapters** from my book, **High-Performance Java Persistence**,
- a **10% discount** coupon for my book.

Get the most out of your persistence layer!

Subscribe

Related

How to enable bytecode enhancement dirty checking in Hibernate
February 11, 2016
In "Hibernate"

The anatomy of Hibernate dirty checking mechanism
August 21, 2014
In "Hibernate"

The best way to lazy load entity attributes using JPA and Hibernate
September 20, 2016
In "Hibernate"

Categories: Hibernate Tags: automatic dirty checking, bytecode enhancement, hibernate, instrumentation, performance tuning, Training, Tutorial

← From mostly interested to most interesting

A beginner's guide to database locking and the lost update phenomena →

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

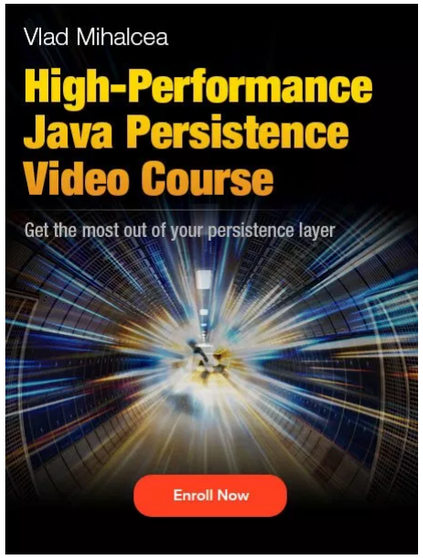
Save my name, email, and website in this browser for the next time I comment.

Sign me up for the newsletter!

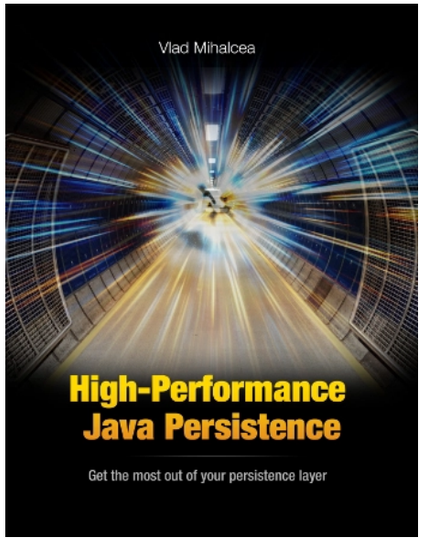
Post Comment

Notify me of follow-up comments by email.

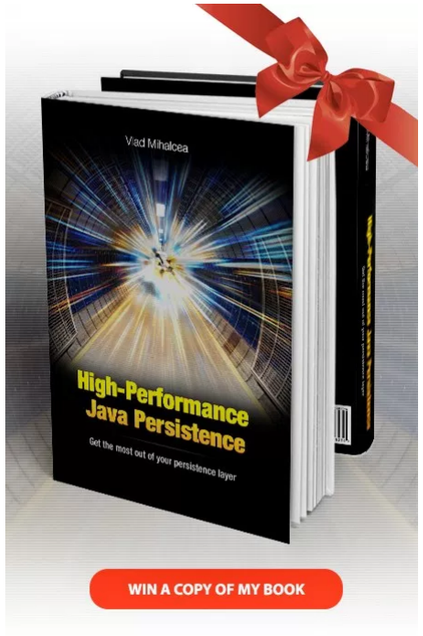
VIDEO COURSE



HIGH-PERFORMANCE JAVA PERSISTENCE



WIN A COPY OF MY BOOK!

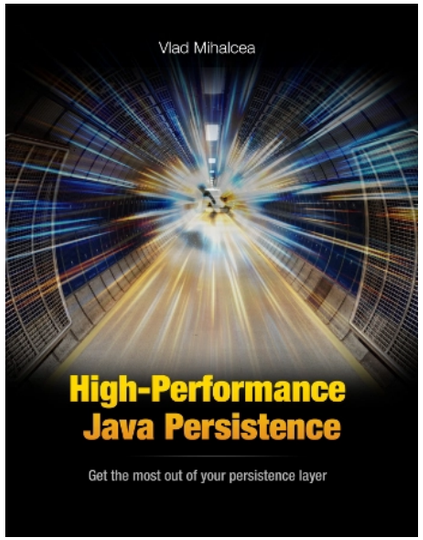


HIBERNATE PERFORMANCE TUNING TIPS

HYPERSISTENCE

WIN A COPY OF MY BOOK!

HIGH-PERFORMANCE JAVA PERSISTENCE



VIDEO COURSE

Search …

RSS - Posts

RSS - Comments