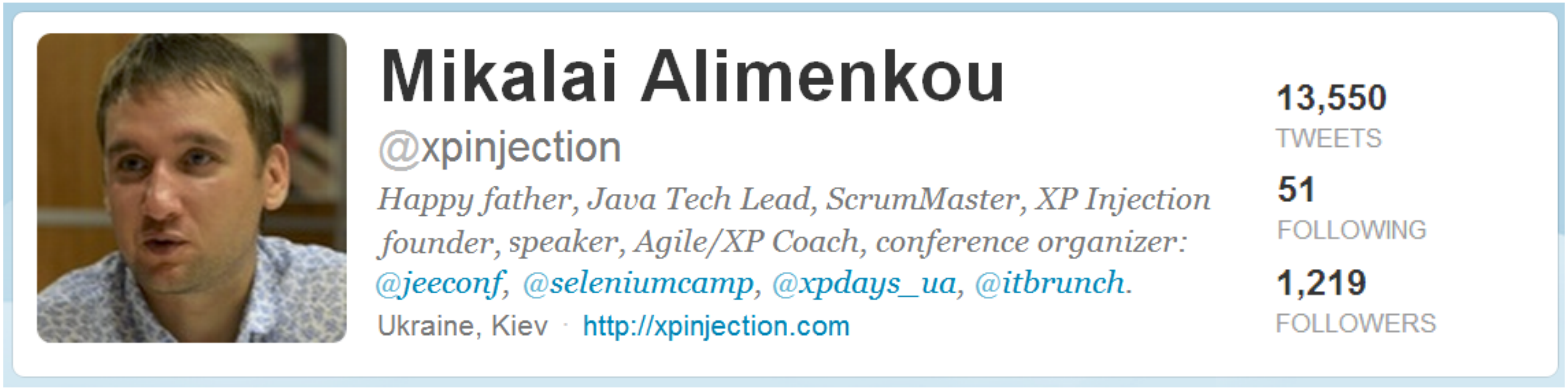
Why do I hate Hibernate?

Mikalai Alimenkou
http://xpinjection.com

18.04.2014



Mikalai Alimenkou

@xpinjection

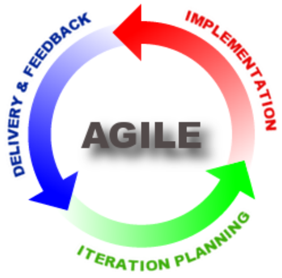
Happy father, Java Tech Lead, ScrumMaster, XP Injection founder, speaker, Agile/XP Coach, conference organizer: @jeeconf, @seleniumcamp, @xpdays_ua, @itbrunch.

Ukraine, Kiev · http://xpinjection.com

13,550 TWEETS

51 FOLLOWING

1,219 FOLLOWERS



7 YEARS, EXPERT

9 YEARS, TECH LEAD

FOUNDER AND COACH:

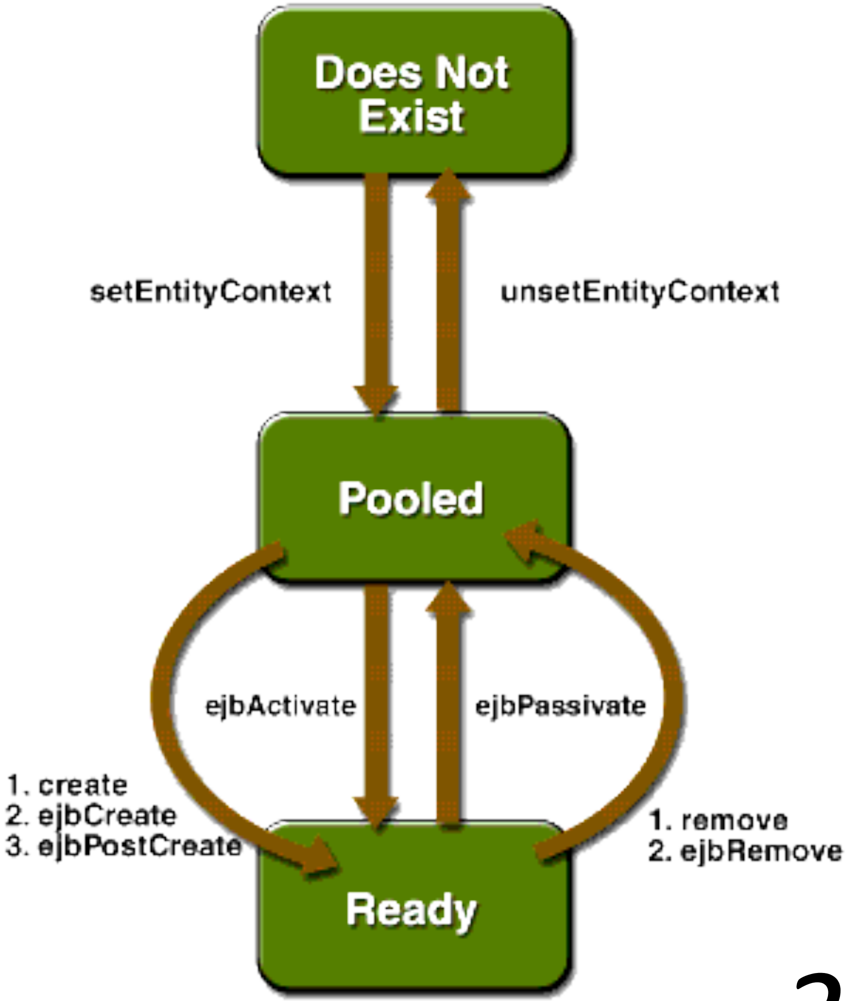
ORGANIZER AND SPEAKER:

HIBERNATE

CRUD

**R – read by ID*

Why Hibernate is so popular?



1. EJB 2 sucks!

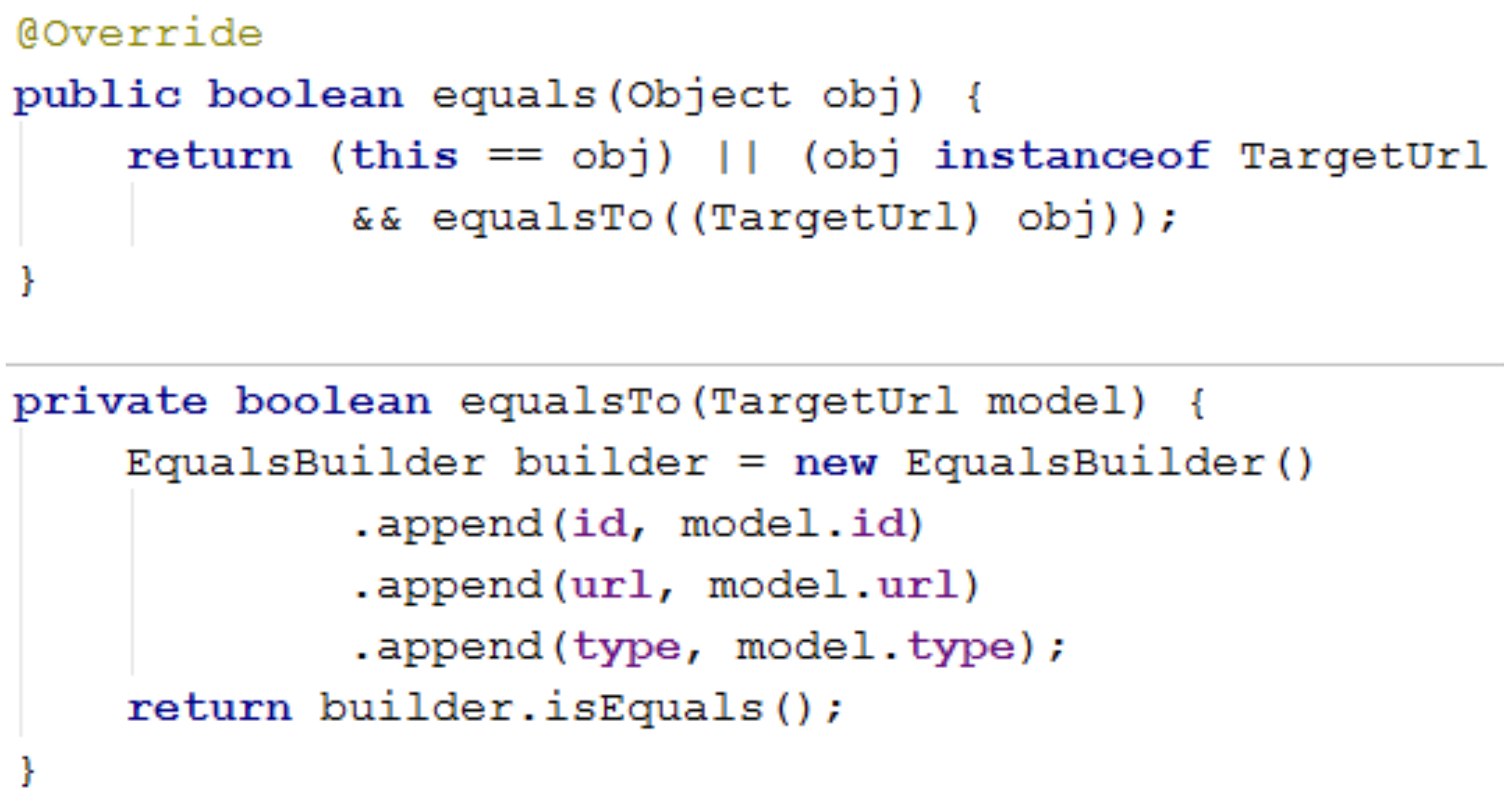
spring

+

HIBERNATE

2. New "standard"

WTF #1. Nobody understand how *equals* and *hashCode* must look like



```
@Override
public boolean equals(Object obj) {
    return (this == obj) || (obj instanceof TargetUrl
            && equalsTo((TargetUrl) obj));
}
```

```
private boolean equalsTo(TargetUrl model) {
    EqualsBuilder builder = new EqualsBuilder()
            .append(id, model.id)
            .append(url, model.url)
            .append(type, model.type);
    return builder.isEquals();
}
```

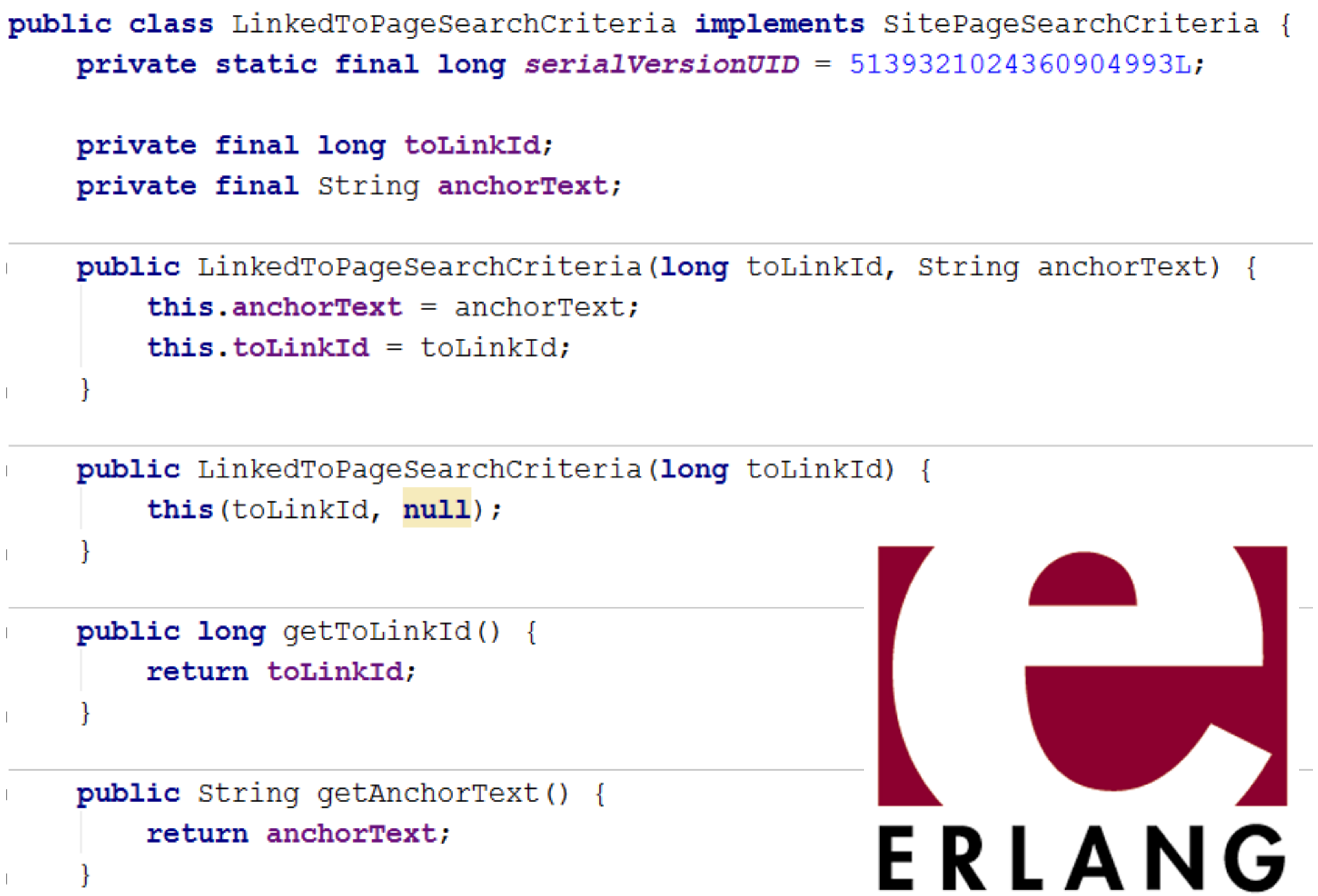
Comparison context is very important

Hard to select real "unique business key"

Performance issues with collections

Auto-generated fields in Set

WTF #2. Mutable objects are evil



```java
public class LinkedToPageSearchCriteria implements SitePageSearchCriteria {
    private static final long serialVersionUID = 5139321024360904993L;

    private final long toLinkId;
    private final String anchorText;

    public LinkedToPageSearchCriteria(long toLinkId, String anchorText) {
        this.anchorText = anchorText;
        this.toLinkId = toLinkId;
    }

    public LinkedToPageSearchCriteria(long toLinkId) {
        this(toLinkId, null);
    }

    public long getToLinkId() {
        return toLinkId;
    }

    public String getAnchorText() {
        return anchorText;
    }
```

ERLANG

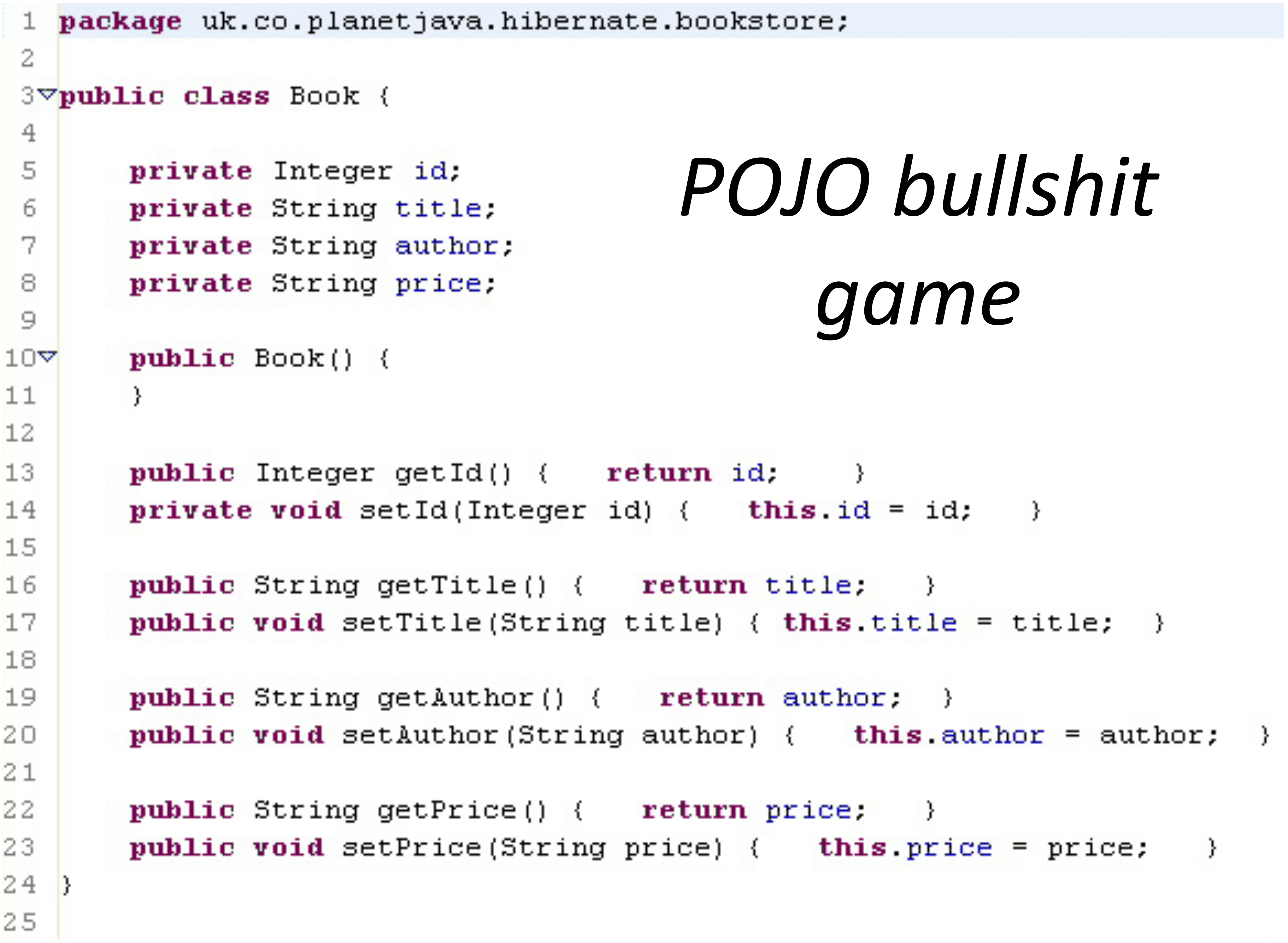
Default constructor must present

Fields can't be final

Inconsistent object state is possible

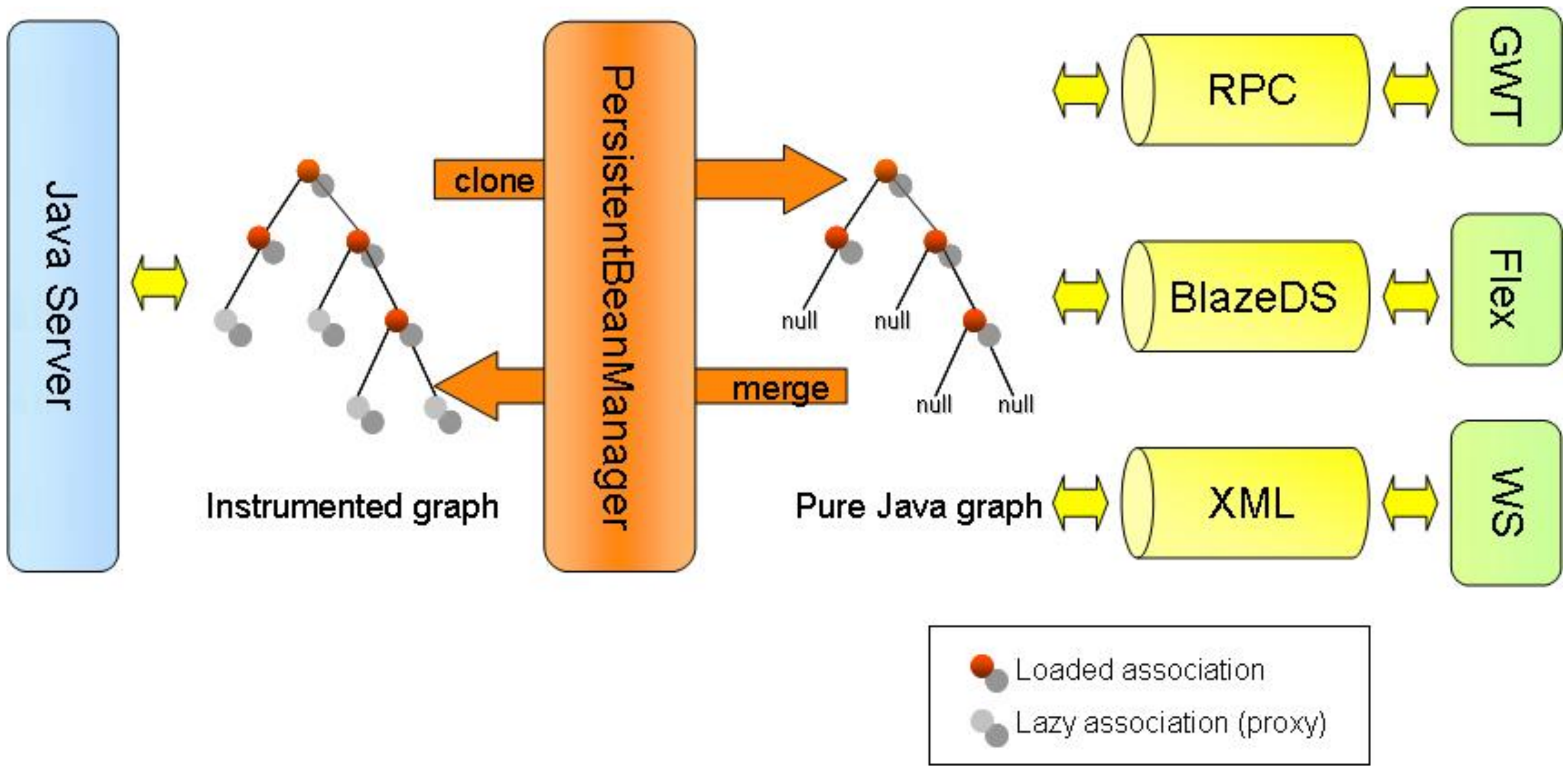
Or… reflection magic happens

POJO bullshit game



```java
package uk.co.planetjava.hibernate.bookstore;

public class Book {

    private Integer id;
    private String title;
    private String author;
    private String price;

    public Book() {
    }

    public Integer getId() {   return id;     }
    private void setId(Integer id) {   this.id = id;   }

    public String getTitle() {   return title;   }
    public void setTitle(String title) { this.title = title;  }

    public String getAuthor() {   return author;  }
    public void setAuthor(String author) {   this.author = author;  }

    public String getPrice() {   return price;   }
    public void setPrice(String price) {   this.price = price;   }
}

```

WTF #3. DTO is evil pattern with Hibernate



Lots of code duplications

DTO maintenance is nightmare

Many solutions:
- transformers
- beanutils
- reflection
- libraries

Crazy DTO “don’t forget” checklist

WTF #4. LazyInitializationException

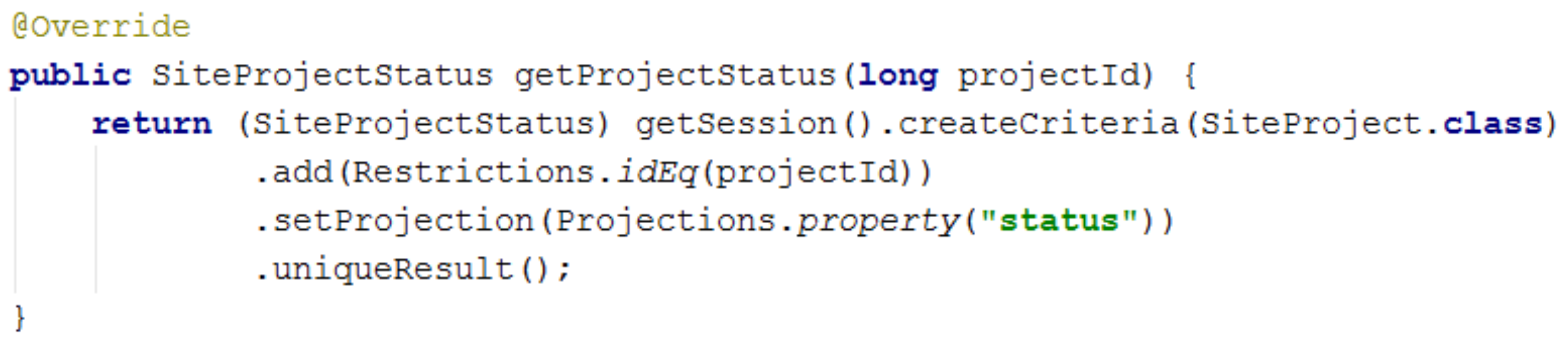
Nightmare for newcomers

Everything is stupidly slow without it

Almost nobody understand how it works

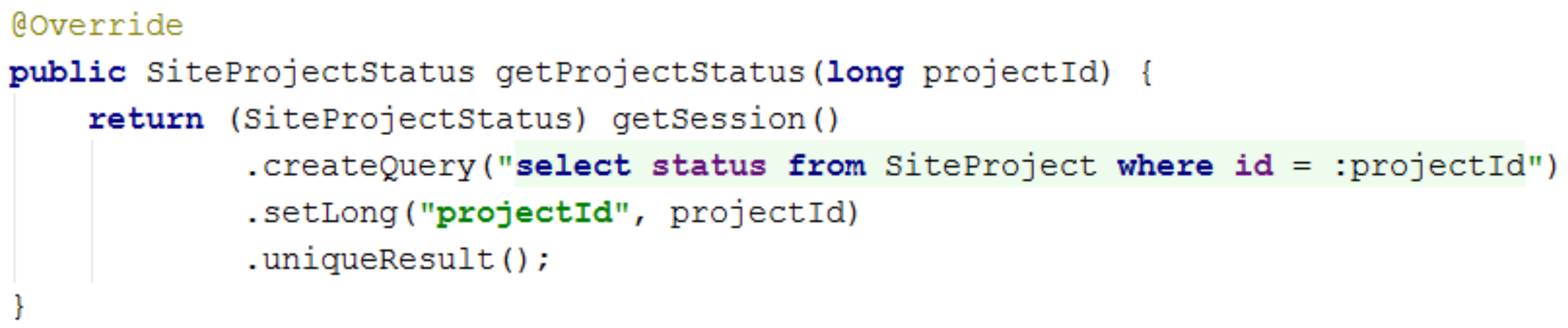
Persistent collections are not serializable

WTF #5. Criteria API or HQL?



```java
@Override
public SiteProjectStatus getProjectStatus(long projectId) {
    return (SiteProjectStatus) getSession().createCriteria(SiteProject.class)
            .add(Restrictions.idEq(projectId))
            .setProjection(Projections.property("status"))
            .uniqueResult();
}
```

=



```java
@Override
public SiteProjectStatus getProjectStatus(long projectId) {
    return (SiteProjectStatus) getSession()
            .createQuery("select status from SiteProject where id = :projectId")
            .setLong("projectId", projectId)
            .uniqueResult();
}
```

WTF?!?
No need to know
SQL???
Bullshit!!!

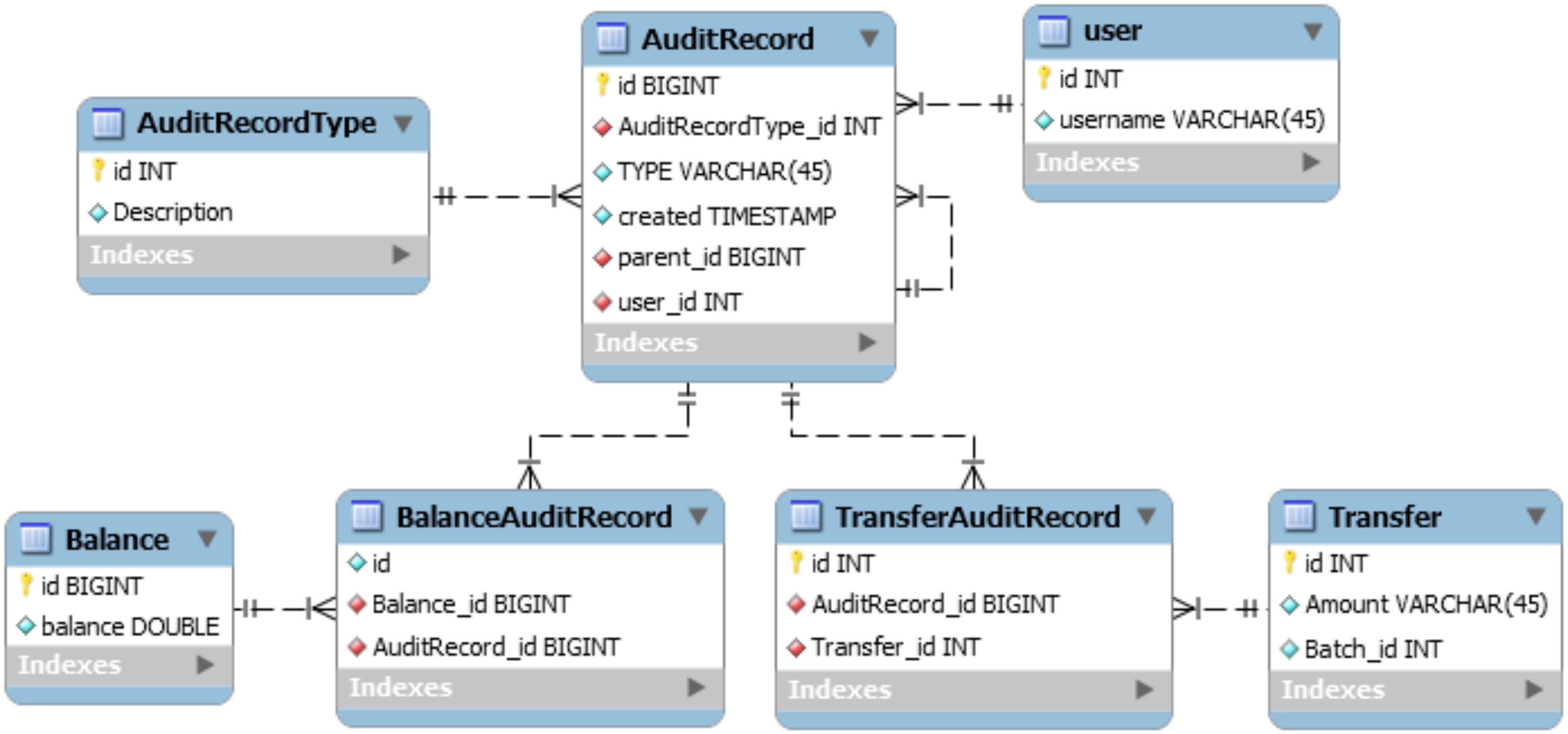
Different features set in HQL and Criteria API

Hand made query builders in HQL

No check for NULL in HQL

Criteria API only for READ

WTF #6. To update single field you should read full entity



Not everybody understand how 'lazy loading' works

Children and large fields are loaded

HQL update is not object-oriented

At least 2 SQL queries for simple update

WTF #7. Temporary table usage on update or delete

```
"delete Human h where h.firstName = 'Steve'"
```

=

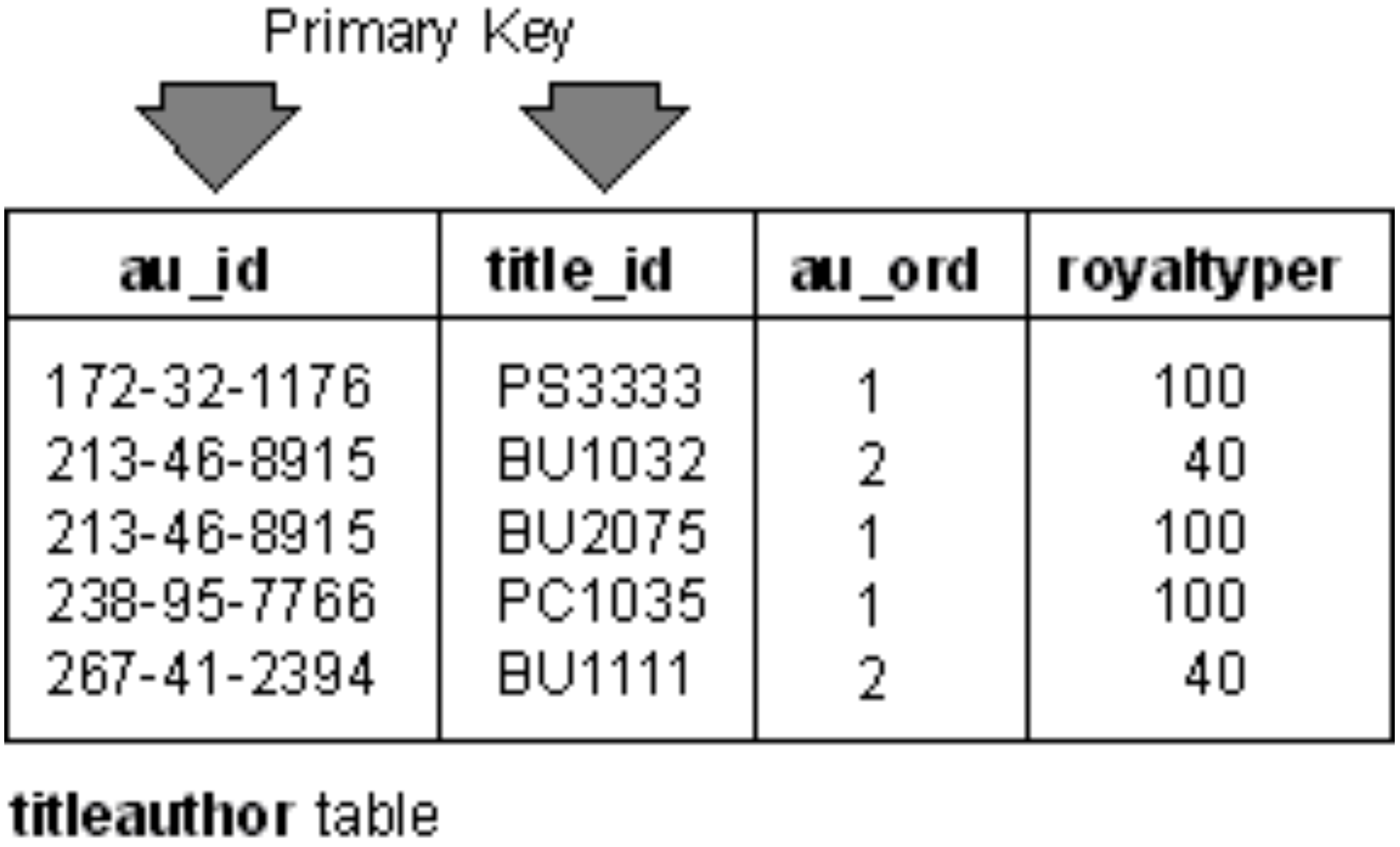
```
"insert into HT HUMAN (ID) select ID
       from HUMAN where f_name = 'Steve'"
```

+

```
"delete from HUMAN where ID
        IN (select ID from HT_HUMAN)"
```

```
Animal
  /   \
 /     \
Mammal   Reptile
    / \
   /   \
Human   Dog
```

WTF #8. Composite key requires separate class


Primary Key

au_id	title_id	au_ord	royaltyper
172-32-1176	PS3333	1	100
213-46-8915	BU1032	2	40
213-46-8915	BU2075	1	100
238-95-7766	PC1035	1	100
267-41-2394	BU1111	2	40

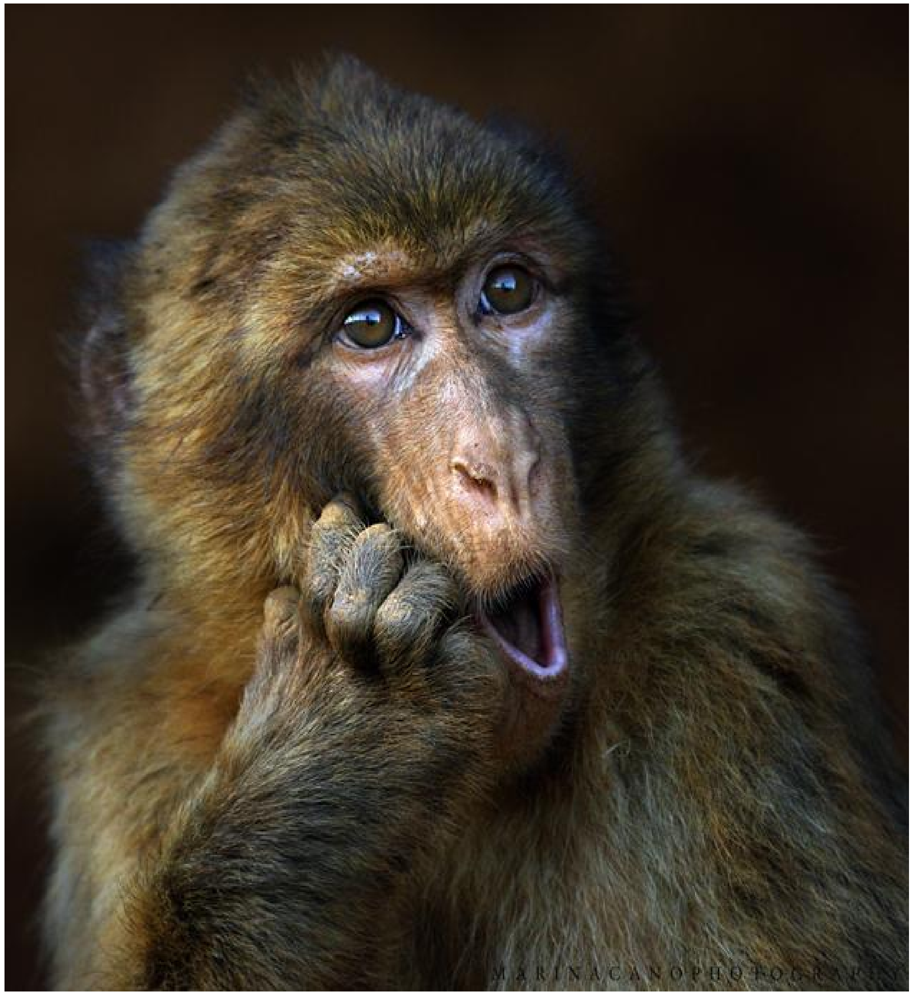
titleauthor table

WTF #9. Entity state can be managed only from one side in parent-child relationship

```
<set name="children" inverse="true">
    <key column="parent_id"/>
    <one-to-many class="Child"/>
</set>
```

```
Parent p = (Parent) session.load(Parent.class, pid);
Child c = new Child();
c.setParent(p);
p.getChildren().add(c);
session.save(c);
session.flush();
```

WTF #10. Flush on commit is too unpredictable



Entity changes are persisted

Collections operations are ‘live’

Constraints are checked on commit

No control for SQL queries order

WTF #11. Dirty entity after transaction rollback can't be reused

WTF #12. Documentation is simple but primitive

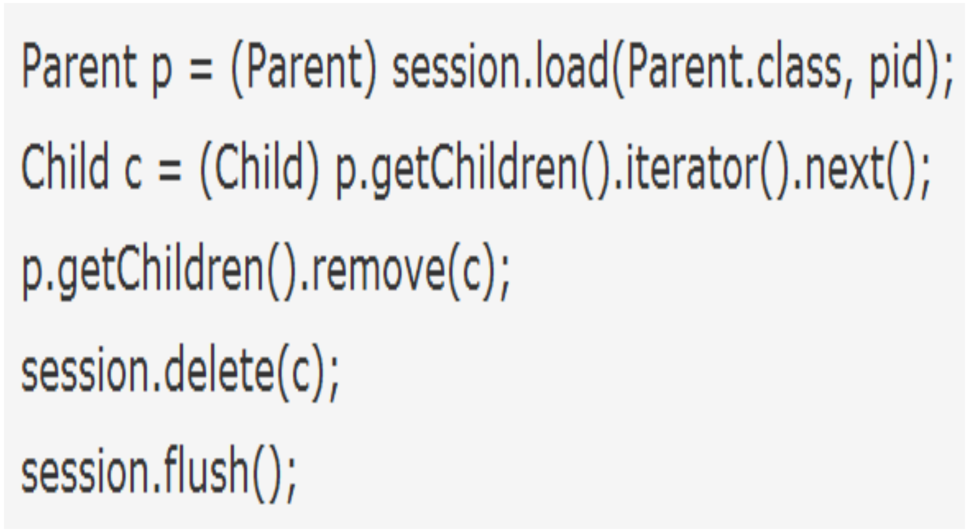
For complex things you need books

Only basic samples are covered

Bug reports are ignored

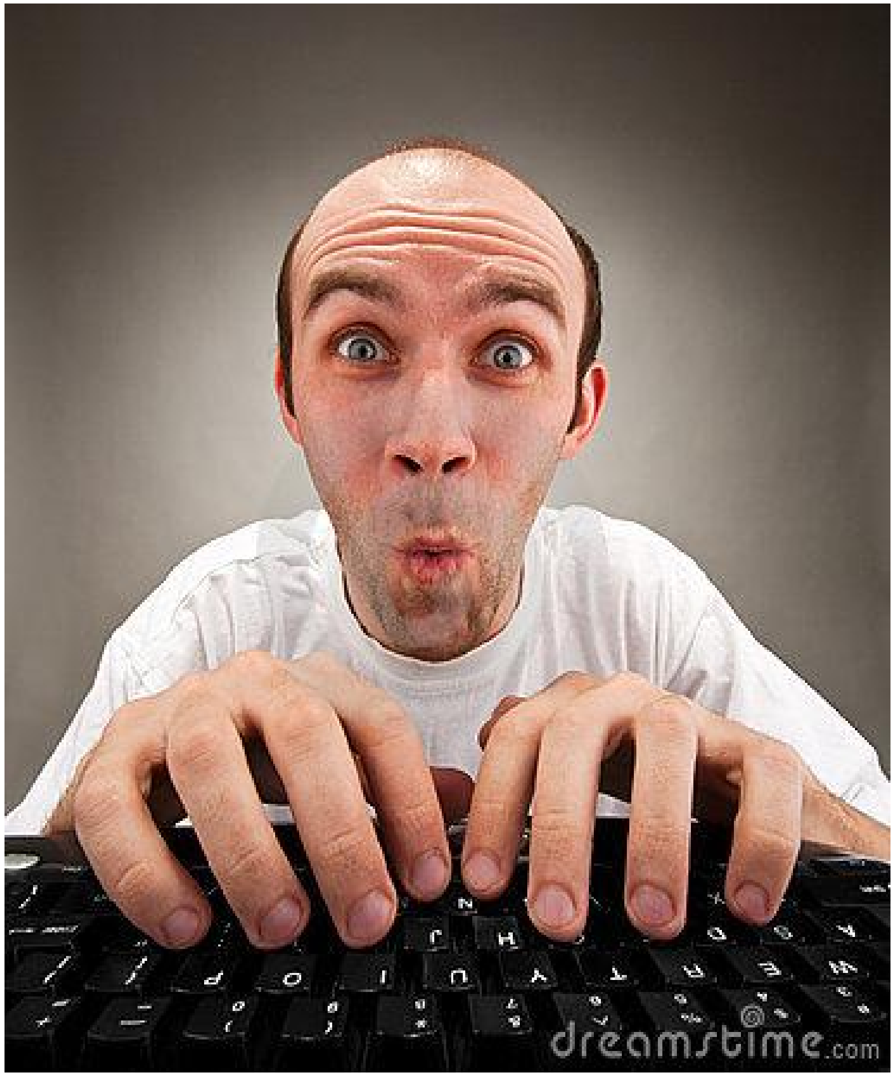
Debug hell to understand how things work

WTF #13. Hibernate makes developers stupid



```
Parent p = (Parent) session.load(Parent.class, pid);
Child c = (Child) p.getChildren().iterator().next();
p.getChildren().remove(c);
session.delete(c);
session.flush();
```

Are you joking???

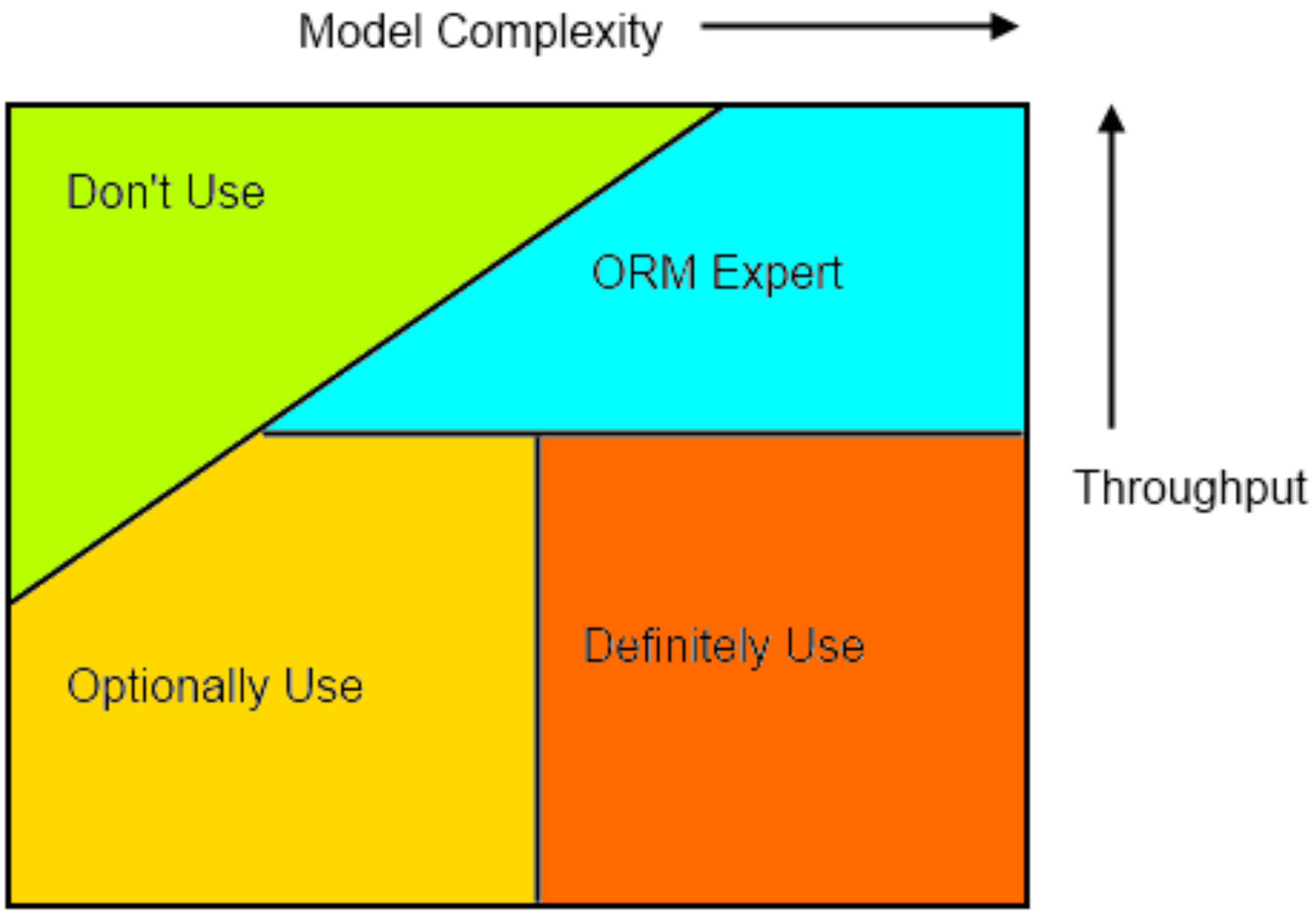


People don’t learn JDBC and SQL

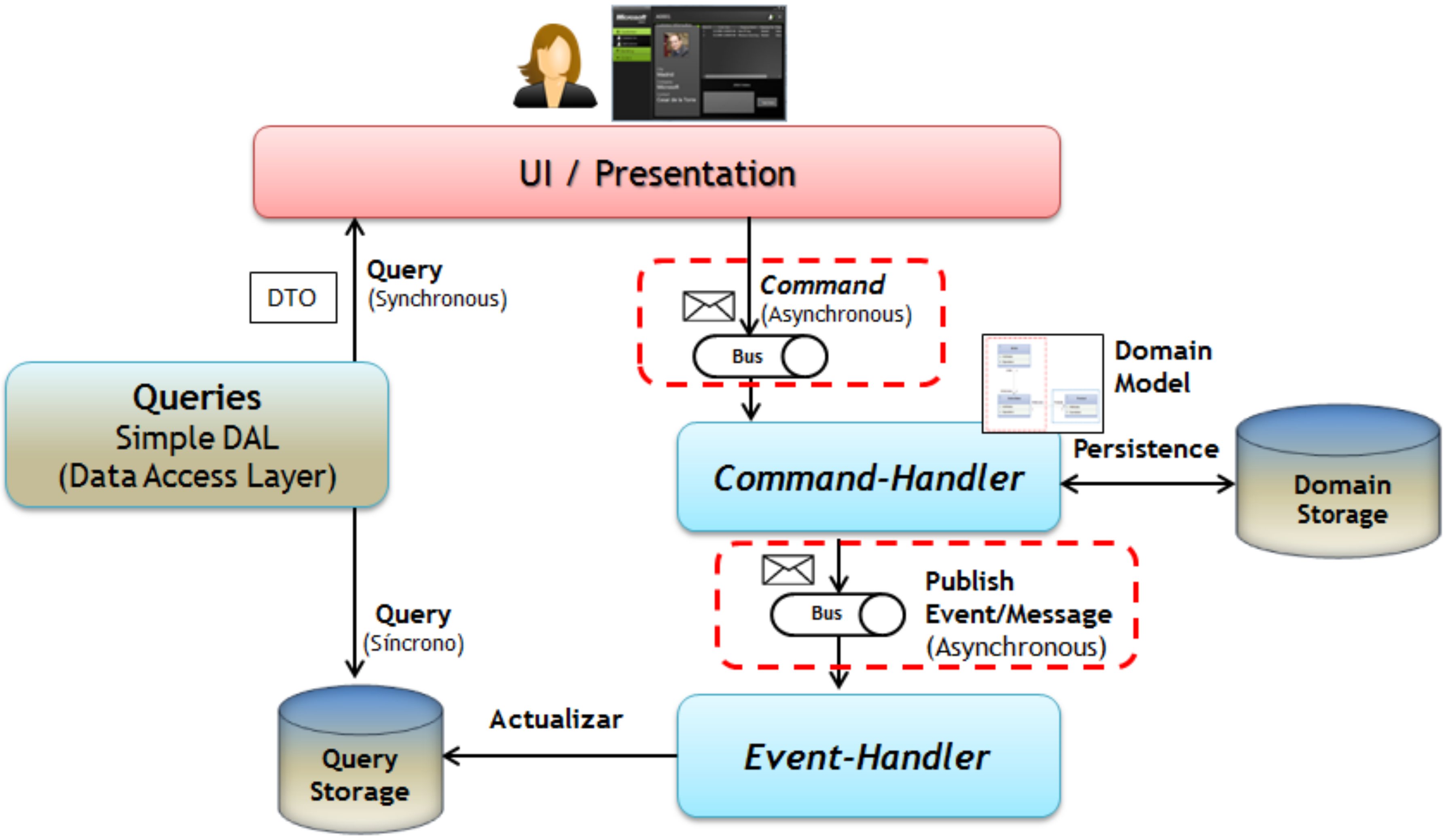
Nobody count SQL queries and care about their structure

Developers don’ t tune JDBC properties

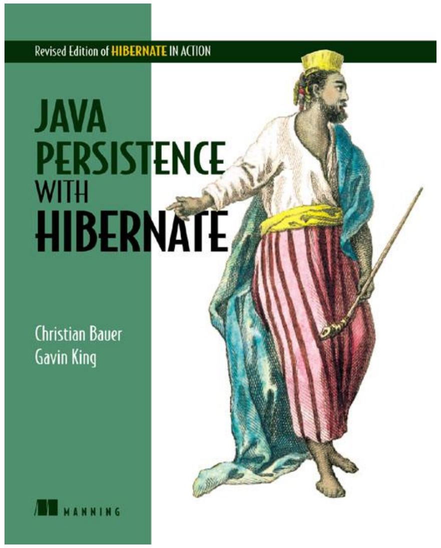
When we really need to use ORM?



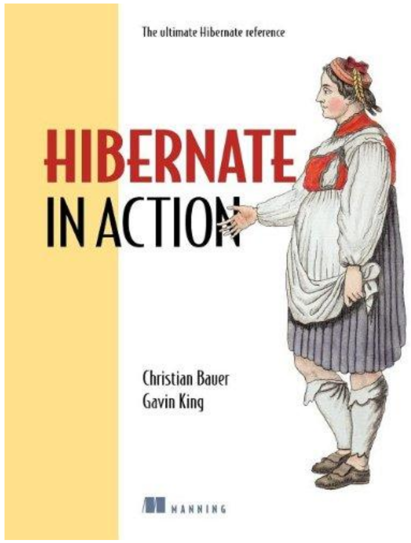
Is CQRS more actual now?

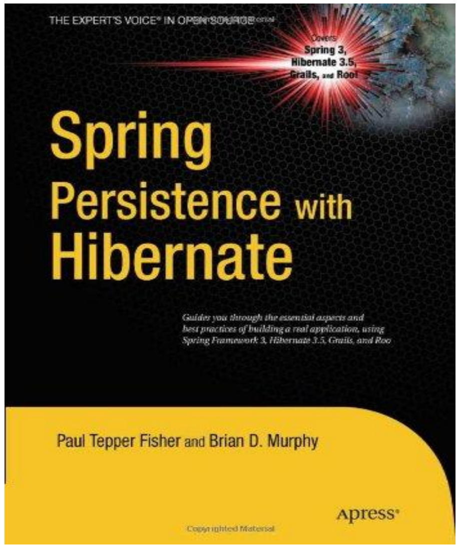


Useful books

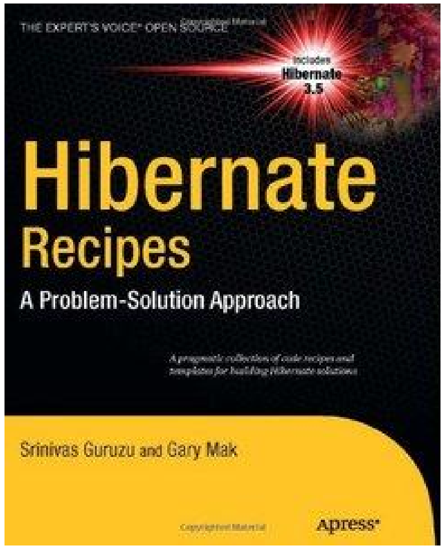


Revised Edition of HIBERNATE IN ACTION
JAVA PERSISTENCE WITH HIBERNATE
Christian Bauer
Gavin King
MANNING





THE EXPERT'S VOICE® IN OPEN SOURCE
Covers Spring 3, Hibernate 3.5, Grails, and Roo!
Spring Persistence with Hibernate
Guides you through the essential aspects and best practices of building a real application, using Spring Framework 3, Hibernate 3.5, Grails, and Roo
Paul Tepper Fisher and Brian D. Murphy
Apress®



THE EXPERT'S VOICE® OPEN SOURCE
Includes Hibernate 3.5
Hibernate Recipes
A Problem-Solution Approach
A pragmatic collection of code recipes and templates for building Hibernate solutions
Srinivas Guruzu and Gary Mak
Apress®

@xpinjection
http://xpinjection.com
mikalai.alimenkou@xpinjection.com