



# Hibernate in Practice

Rob Harrop

# Introductions

- Rob Harrop
- VP, Interface21
- Core developer on Spring
- Frequent presenter (JUGs, JavaOne, JavaPolis, OSCON...)
- Consultant with Fortune 500 clients (mainly banking) and government
- Author of best-selling Pro Spring

# Agenda

- What you need to know to be effective
  - Mapping object graphs
  - Session management
  - Fetching strategies
- Hibernate favourites
  - Criteria API
  - Testing



# What you need to know to be effective

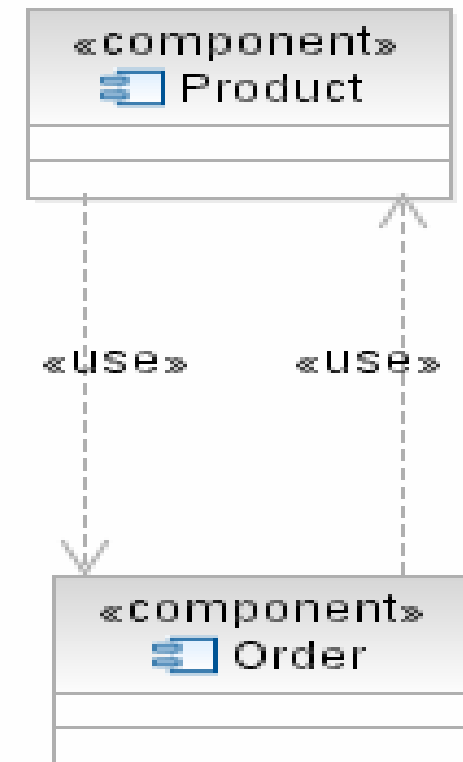
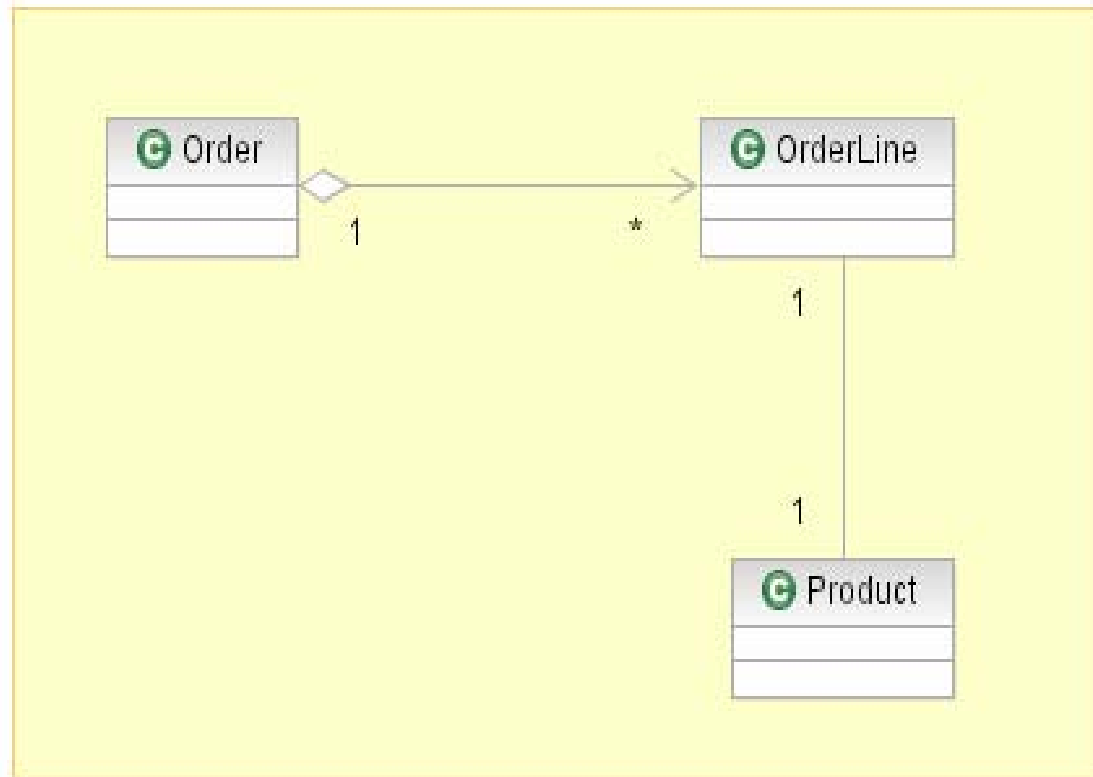
- Getting started with Hibernate is easy
- Unfortunately, so is getting it wrong!
- Mastering some central concepts makes success much more likely
  - Domain mapping
  - Session handling
  - Fetching



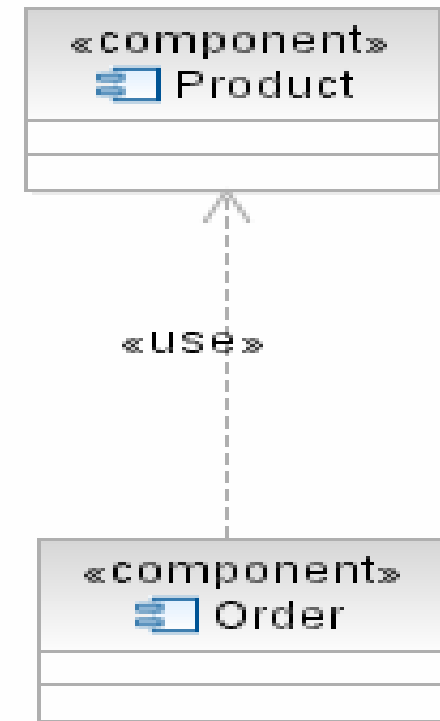
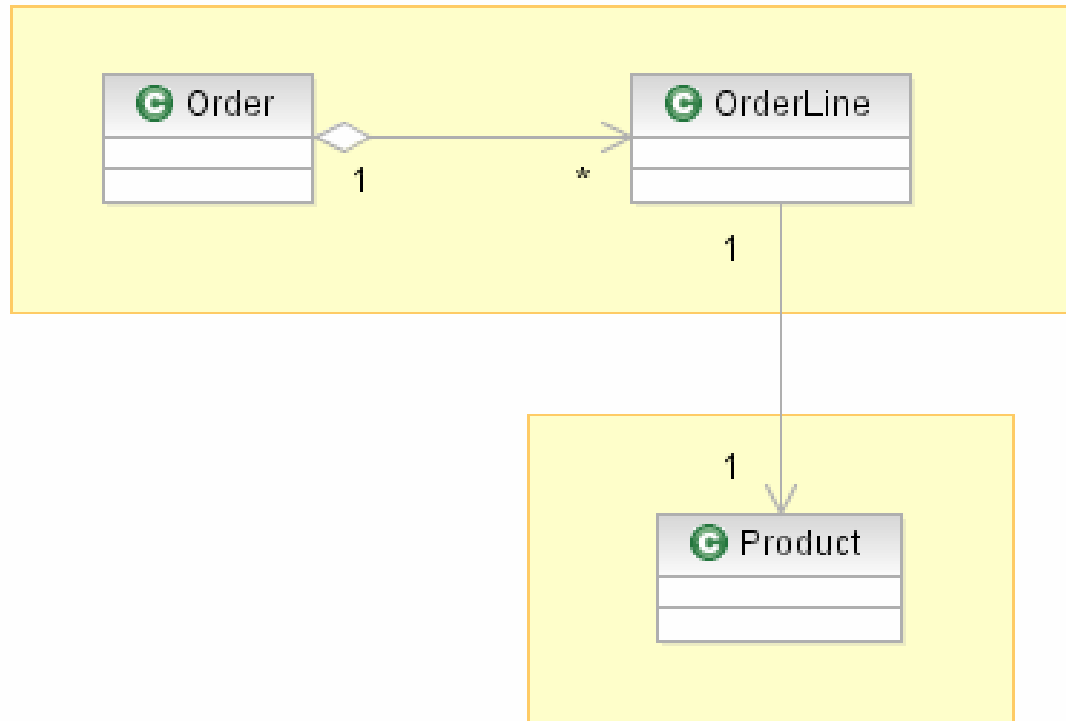
# Mapping object graphs

- Aims
  - Loose coupling
  - High cohesion
  - Accurate mapping of the domain
- Avoid
  - Total navigability
  - Relationships to simplify queries

# Comparing Mapping Styles



# Comparing Mapping Styles



# Mapping object graphs

- Encapsulate structural logic
  - Support code evolution
  - Provide a cleaner API
- Avoid mapping **every** relationship
  - Reduce coupling
  - Ease maintenance



# Session Management

- Session handling can make or break an application
- Too fine-grained and you lose any performance benefits and real transparent persistence
- Too coarse-grained and you risk memory usage problems and slow flushes.



# Session per Operation

- **Do not do this!**

# Options

- Session per transaction
  - Session lifecycle is synchronized with transaction boundary
  - **Do not use transaction per operation :)**
- Long running session
  - Sessions live across multiple transactions



# Issues with Long Running Sessions

- Memory usage
  - Keep Sessions around in memory
- Unexpected flushes
- Manual management patterns
  - Filter/ThreadLocal

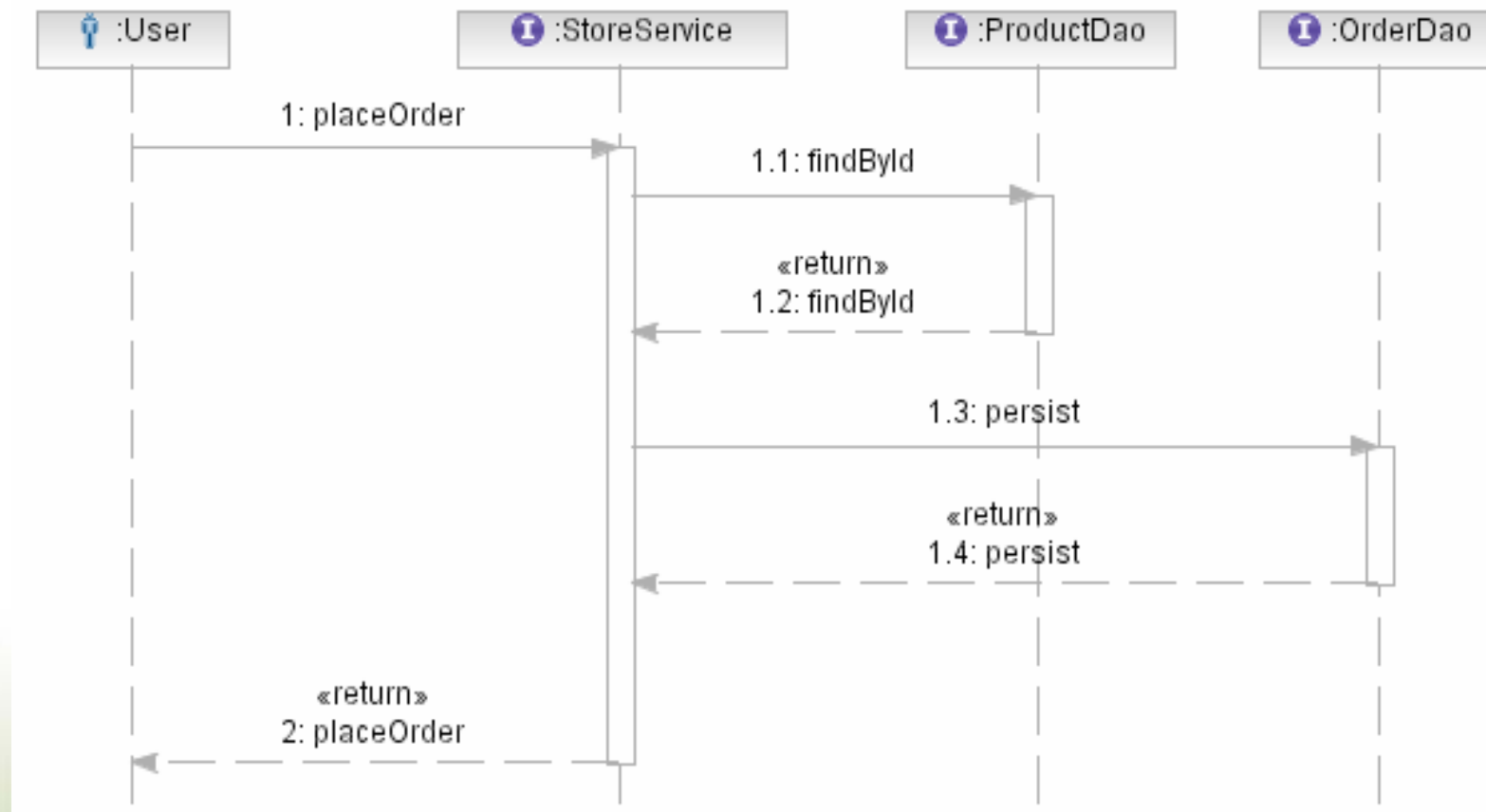


# Session per transaction

- Sessions live for the duration of a transaction
- **DAOs share the same session and transaction**
- **Sessions and transactions are managed transparently**
  - Do not pass them around in method signatures
- **This gives a clean unit of work within which persistence logic can be handled**



# Session per transaction





# Open Session in View

- Steer clear wherever possible
- Unclear transactional boundaries
- Potential for incorrect data
- Real potential for poor performance
  - n+1 selects



# Fetch Joins

- Fetch joins allows eager loading to be selected on a per association, per query basis
  - No need to make everything eager
  - No need to use OSiV
- Typically associations are lazy by default and then eagerly fetched as needed



# Nuances with Fetch Joins

- Unexpected number of results
- The exact number of queries may surprise you

# Criteria API

- Programmatic construction of queries
- Support for all common operators
  - =, <>, >, <, IN, EXISTS, BETWEEN, LIKE
- Full support for projections
- Support for custom SQL
  - Projection and selection
- Additional features
  - Paging, sorting, grouping



# Criteria API

- Simple query

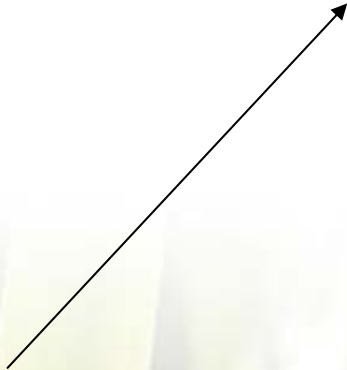
```
Criteria c = session.createCriteria(Product.class);  
c.add(Restrictions.eq("name", value));  
return c.list();
```

# Criteria API

- Query and projection

```
Criteria c = session.createCriteria(Product.class);  
c.add(eq("name", value));  
c.setProjection(distinct(property("name")));
```

Static imports from Projections





# Criteria API

- Using SQL

```
Criteria c = session.createCriteria(Product.class);  
c.add(sqlRestriction("{alias}.name = ?", value,  
    Hibernate.STRING));  
return c.list();
```



# When to use the Criteria API

- Ad-hoc querying
- Saved queries
- Conditional queries



# Testing

- Testing tools:
  - DBUnit
  - Spring Mock
- Testing strategy
  - Test with Hibernate
  - Verify with JDBC
  - Work against known sets of data



# Useful Resources

- [http://www.hibernate.org/hib\\_docs/v3/reference/en/html/](http://www.hibernate.org/hib_docs/v3/reference/en/html/)
- Java Persistence with Hibernate (King, Bauer)
- DBUnit - <http://dbunit.sourceforge.net/>