Enterprise Programmering 1
Lesson 03: JPA

Dr. Andrea Arcuri
Westerdals Oslo ACT
University of Luxembourg

About these slides

- These slides are just high level overviews of the topics covered in class
- The details are directly in the code comments on the Git repository

Locks

- Example: you read a counter variable from the DB, increment it by 1, and then save it back to the DB
- What if someone (ie, thread/process) modifies the counter *after* you read it, but *before* you write the increment back?
- Need mechanism to have atomic operations
- JPA provides *Optimistic* and *Pessimistic* Locks

Optimistic Locks

- Entity will have a numeric value tagged @Version
- Every operation on the entity, version increases
- When writing back a value, check if version has been increased
  - If so, it means someone else did a modification in parallel
- If version mismatches, throw exception, and can try to do operation again
- Optimistic: cheap to do, good for cases in which clashes are rare

Pessimistic Locks

- Handled directly by the DB
- More expensive, as other threads will put on hold until the locks are released when an atomic operation is completed
- Pessimistic: best when it is very likely that there are going to be many concurrent accesses

Validation

- How to say that a String in the DB should not be too long?
- How to say that a String should represent a valid email?
- How to say that an Integer should be constrained in a specify range?
- Can have special validation tags on the @Entity fields
- Can also have custom constrains

Validation Annotations

- @NotNull
- @Size(min=?, max=?)
- @Pattern
- @NotBlank
- @Email
- etc.

Reasons for using constraints

1. If something goes wrong, you want a failure as soon as possible, with a clear reason, ie *fail fast* principle
2. Good for non-ambiguous documentation
3. Security, eg, prevent DOS of username fields filled with 10GB long usernames…

JPA Implementations

- Hibernate is the most used JPA implementation
- EclipseLink is another one
- One role of ORMs is to translate your EntityManager and JPQL operations into efficient SQL commands
- That’s good for many cases, but you can end up with inefficient SQL or straight up nonsense…
- Remember: libraries can have bugs, or very “peculiar”, unexpected behaviors…

Git Repository Modules

- *NOTE: most of the explanations will be directly in the code as comments, and not here in the slides*
- **intro/jee/jpa/lock**
- **intro/jee/jpa/validation**
- **intro/jee/jpa/outerjoin**
- Exercises for Lesson 03 (see documentation)