Enterprise Programmering 1

Lesson 06: EJB

Prof. Andrea Arcuri

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

Proxy Class Enhancements

- You focus on business logic when writing EJBs
- JEE Container will automatically add functionalities
- Eg, starting/committing transaction for each method invocation
- But also other functionalities are available, and can be activated via **@** annotations

@Asynchronous

- A method marked *@Asynchronous* will be executed on the background in a different thread, and caller returns immediately (not going to wait for method to end)
- Useful for long operations, and when caller does not need to get the result (important thing is the side-effect)
- You do not need to create threads (which is expensive), or manually manage a pool of them… the JEE Container will do it for you automatically by using this annotation

@Schedule

- Let’s say you need to do an action one or more times at a certain interval of time
  - eg, check an external resource for updated every 30 seconds
  - eg., send special offers to customers on his/her birthday
- You can implement your own threads and scheduler… or you can just annotate a EJB method with @Schedule
  - Fine grained settings to specify seconds/minutes/hours/etc.

Transactions

- EJB methods are executed in a transaction…
- But what if EJB call a method in another EJB?
  - Would it be in a new transaction?
  - Are the transaction joined in a single one?
- What if EJB method calls another method in the same EJB?
  - This one is actually quite tricky…
- What if an exception is thrown during a EJB method call?
  - Are the changes so far on the EntityManager cache committed?
  - Or everything is rolledback?
- All these cases can be set with annotations…
- … default settings are going to be fine most of the time, but need to have a clear understanding of when transactions start/end

Transaction Handling

- EJB public methods will handle transactions by default
- Can use @annotations to fine tune them
  - *REQUIRED*: default setting, start new transaction if none is active, or join current active one
  - *SUPPORTS*: if there is an ongoing transaction, join it
  - *REQUIRES_NEW*: always start a new transaction. If any ongoing, suspend them first
  - *MANDATORY*: must be run in an ongoing transaction, otherwise fail
  - *NOT_SUPPORTED*: put any ongoing transaction on hold
  - *NEVER*: throw exception if in a transaction

Git Repository Modules

- *NOTE: most of the explanations will be directly in the code as comments, and not here in the slides*
- **intro/jee/ejb/async**
- **intro/jee/ejb/time**
- **intro/jee/ejb/transactions**
- Exercises for Lesson 06 (see documentation)