The Learning House

427 S. 4th Street #300

Louisville KY 40202

Java and Databases Unit

Lesson 2- Database Transactions with Spring

SOFTWARE GUILD

Objectives

- Explain Spring Database Transaction support and the associated annotations/settings

Service Layer Overview

- Is the data access API for the application
- Uses the very granular DAO methods for more meaningful/complicated tasks (i.e. TransferFunds)
- This is where we define transaction boundaries

ACID

- Spring help us define and enforce ACID transactions
  - o Atomic
  - o Consistent
  - o Isolated
  - o Durable
- We can configure these properties

Why do we care?

- Bank - transfer of funds
  - o Debit one account
  - o Credit another account
- Purchasing tickets
  - o Verify seat availability
  - o Reserve seats
  - o Process/receive payment
  - o Issue tickets

Transaction Manager

- Spring does not manage transactions itself
- We must configure a TransactionManager
- Many flavors (similar to the Templates)
- We will use DataSourceTransactionManager

Create Service Layer

- Define operations
- Decide where we want transactions
- We will configure and implement transactions in the next steps
- Create both interface and implementation class
- We will inject the DAO into the Service Layer

Transaction Properties: Propagation

- Mandatory
- Nested
- Never
- Not Supported
- Required
- Requires New
- Supports

Transaction Properties: Isolation Level

- Default
- Read Uncommitted
- Read Committed
- Repeatable Read
- Serializable

Transaction Properties: Read Only and Timeout

- Marking Read Only allows the underlying data store to optimize
- Timeout only makes sense for propagation setting that start a new transaction (Required, Requires New, and Nested)

Transaction Properties: Rollback Rules

- By default transactions only roll back on runtime exceptions, NOT checked exceptions
- You can control this behavior using rollbackFor and noRollbackFor properties

Transaction Configuration

- Can code transactions into Service Layer - very invasive
- Can define and configure in XML - a bit messy
- Annotations are the cleanest
  - o Easy config - just add <tx:annotation driven/>
  - o Easy definition - define transactions at method level in Service Layer

Implement Transaction

- Annotate Service Layer
  - This can be done in XML but it is cumbersome - I like to define the transactions on the Service Layer itself
- Add <tx:annotation-driven> to config file
- Use @Transactional annotation

Implement Service Layer Tests

- Similar to DAO tests