Enterprise systems (starting spring)

Lecture 13

Contents

• Why Spring?
  • Description of the problem
  • Overall view
  • Inspecting a sample of an Spring-Hibernate Application → List of Questions!
• Detailed study of Spring:
  • Dependency Injection
  • Beans and their wiring
  • Aspect oriented Programming
  • Interaction with db
  • Transactions
• Security

Why Spring?

You have learnt Hibernate. It allows ORM + nice functionalities such as association implementation, Lazy loading, Cascading

Object model

Data model

No matter what technology, we need to access data!

1. We tend to re-execute the same types of query in different part of the code
2. We may need to move to a new technology or a newer version of a technology

Spring proposes Data Access Object (DAO)

DAO

DAO allows reading and writing data to database. Application delegates task of access to data to DAO

What is an interface?
Advantages of using DAO for data

One implementation and reuse
• Less code- cheaper, less chance of error, less testing effort

Single point of change
• If migrating to new technology, version, implementation

Questions
• How does the interaction happen?
• Why interfaces?

Why Spring? Simpler Exceptions

In accessing data various things can fail. SQL exception can be thrown if:
• application is unable to connect to the database.
• query being performed has errors in its syntax.
• tables and/or columns referred to in the query do not exist.
• An attempt is made to insert or update values that violate a database constraint.

Exceptions

Some exceptions if caught are useful, such as rollback
But most are not!
If a connection is failed- no matter what the reason, we can’t do much 😞
Spring promotes the use of unchecked exceptions, which are independent of technology!

Why Spring? Reuse Business Logic

Apart from DAO for data we have “Application DAO”
Business functionality which is reused

Why spring? Decoupled applications

Build, Test, Modify one “part” of the application independent of another parts
• Dependency Injection
• Aspect oriented Programming
This although fundamental, we will look at briefly!

A Birdseye view of Architecture via Spring and Hibernate

Note the separation of duties!

Presentation tier
JSP, Wicket...

Business Services
Layer- independent of persistence layer

DAO interface
for persistent operations

Persistent domain model
POJO

Hibernate
ORM

DAO interface implementation using ORM

Data Access Template

Persistence Framework

Database
Description of the example

Consider a simplified e-business system for the bookshop [download the code from www.cs.bham.ac.uk/~bxb/courses/es1213/code/shop-spr-hib.jar]

1) create/drop database
2) list-customers
3) print-customer details given the email
4) get a list of Books
...

Description of the example

ehcache.xml seen it before
-log4j.properties seen it before –shop
spring-hib.xml this is important
- shop/
  |- dao/ DAO for data access
  |- model/ POJO models
  |- usecases/ DAO for Business
  |- Main.java logic (controller)