JPA, DAO, Persistence Layer

Martin Ledvinka

martin.ledvinka@fel.cvut.cz

Winter Term 2024

Contents

Persistence Layer

Persistence Layer

- In a layered architecture, persistent data are handled by the *persistence layer*
- Data storage itself is typically handled by a database management system (DBMS), the persistence layer moderates access to it
- Separates business logic from data handling
  - Isolation of data access
  - Possible to develop components separately
  - Clearly defined layer API

Persistence Layer in Layered System

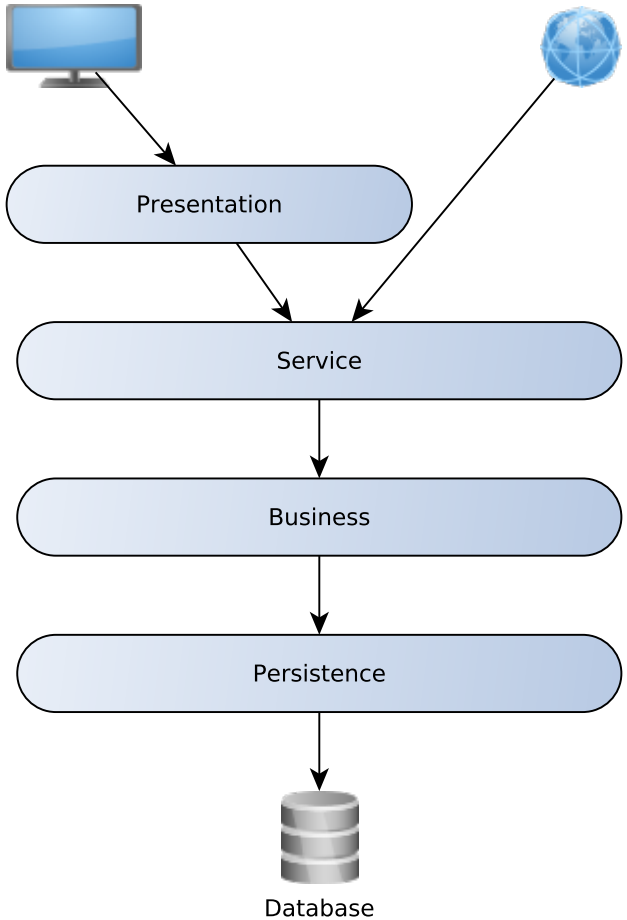

Figure: Layered architecture visualization.

Data Access Objects

Data Access Object (DAO)

- Persistence layer usually consists of *data access objects* (DAOs)

DAO

- Encapsulates access to data via a particular provider
- Provides clear API
- Create, Retrieve, Update, Delete (CRUD) operations + complex operations and queries
- No business logic!
- Entity class $\neq$ dedicated DAO

JPA vs DAO

- `EntityManager` is essentially a generic DAO
- But specific to JPA (relational databases)
- Wrapping JPA/Hibernate in application-specific DAO allows hiding the implementation details of its usage

Tasks

Syncing Your Fork

1. Ensure you have no uncommitted changes in the working tree
   - `git status` → your branch is up to date, nothing to commit
2. Fetch branches and commits from the upstream repository (EAR/B241-eshop)
   - `git fetch upstream`
3. Check out the task branch from **upstream** (one line!)
   - `git checkout -b b241-seminar-04-task upstream/b241-seminar-04-task`
4. Do the task
5. Commit and push the solution to **your** fork
   - `git push -u origin b241-seminar-04-task`

Building a Persistence Layer

- We have a `GenericDao` interface defined in e-shop
- It defines basic CRUD operations
- Let's start implementing it
  - `CartDao`
  - `ItemDao`
  - `CategoryDao`
  - `OrderDao`
  - `ProductDao`
- Does the code repeat? Then let's extract it into a common superclass!

Task: 1 Point

There is one more DAO missing: `UserDao`. Besides basic CRUD operations, it will eventually have to provide a method for getting user based on their username (for login).
Also, we are still missing update and delete methods in our DAOs.

1. Create class `UserDao` and implement method `findByUsername` which retrieves user given the specified username.
   - *Hint: use a named query.*
   - *Hint: do not forget to annotate the DAO with `@Repository` (we will find out more about it next time).*
2. Add the missing implementation of update and removal into our DAOs.

Acceptance criteria: Project is buildable by Maven (`package`) – tests pass.

The End

The End

Thank You

Resources

- https://cw.fel.cvut.cz/b241/_media/courses/b6b36ear/lectures/lecture-02-architectures-s.pdf
- https://cw.fel.cvut.cz/b241/_media/courses/b6b36ear/lectures/lecture-03-jpa-s.pdf
- https://www.baeldung.com/java-dao-pattern