Week 14: Research

Prompts

2. Compare JDBC & JPA. Compare and contrast the two approaches.

java - JPA or JDBC, how are they different? - Stack Overflow

JPA vs JDBC | Advantages of JPA over JDBC - java4coding

JDBC (Java Database Connectivity) and JPA (Java Persistence API) are both used for database access in Java applications, but they differ in their approach and functionality. Here is a comparison of the two approaches:

JDBC:

- JDBC is a low-level API that provides a direct connection to the database.
- It requires writing SQL queries and handling database-specific code.
- Developers have more control over the database operations and can optimize queries for specific databases.
- JDBC is database-dependent, meaning different scripts must be written for different databases.
- It requires manual mapping of database results to Java objects.
- JDBC is suitable for applications that require fine-grained control over database operations and performance optimizations.

JPA:

- JPA is a higher-level API that provides an abstraction layer over JDBC.
- It allows developers to work with Java objects and perform database operations without writing SQL queries.
- JPA uses object-relational mapping (ORM) techniques to map Java objects to database tables.
- It provides a set of annotations and APIs to define and manage the mapping between Java objects and database tables.
- JPA is database-agnostic, meaning the same code can be used with different databases with minimal modifications.
- It simplifies database access and reduces the amount of boilerplate code required for database operations.
- JPA is suitable for applications that prioritize productivity, code maintainability, and portability across different databases.

4. What is Lombok? How is it used? What benefits come from using Lombok?

Project Lombok: Clean, Concise Java Code (oracle.com)

What are the benefits and drawbacks of using Lombok to generate equals and hashCode methods? (linkedin.com)

Lombok is a Java library that provides annotations to generate boilerplate code for our classes. By using Lombok, we can avoid writing getters, setters, constructors, toString, equals, hashCode, and other common methods. Lombok will generate them for us at compile time, based on the fields and annotations in our class. For example, we can use the @EqualsAndHashCode annotation to generate the equals and hashCode methods for our class. Lombok works by using a Java annotation processor to modify the abstract syntax tree (AST) of your source code during compilation. The annotation processor reads the annotations in our class and adds the corresponding methods to the AST. The compiler then uses the modified AST to produce the bytecode for our class. This means that Lombok does not affect the runtime performance or behavior of our class. It also means that we need to configure our IDE and build tools to recognize and support Lombok annotations. Using Lombok can save us a lot of time and effort when we are writing Java classes. We can reduce the amount of code that we have to write and maintain and focus on the logic and functionality of our class. We can also avoid common pitfalls and bugs that can arise from writing equals and hashCode methods manually, such as forgetting to update them when adding or changing fields, violating the contract of object equality, or causing memory leaks or performance issues. Lombok can also help us enforce consistency and best practices for object equality across your classes and projects.