IIT Lone Gunmen 2.0

Student Track

1.

Spring JDBC Template is a class provided in the Spring Core to simplify JDBC operations. This class executes the queries provided by the programmer, and in case of retrieve queries, it transforms the ResultSet into array of entity objects. This is thus less error prone than pure JDBC.

In the e-Forms application, this is used in the DAO Implementations.

This however, can be optimized. There are lots of duplications in the code that can be removed.

2.

Hibernate is arguably the most reliable object relational mapping library available. Writing SQL queries manually is a error-prone procedure. ORM libraries provides mappings that allows the developer to forget about the SQL operations completely, and instead think form a object oriented point of view. The entity classes are directly mapped with the database, and the programmer won’t have to worry about writing queries.

Using Hibernate in this project will ensure better maintainability. When the project is modified, the programmer won’t have to worry about writing new SQL statements. This will improve the productivity as well. Another advantage is that it’ll enable the application to run on other databases as well, since there will be no SQL dialect specific queries. The programmer will simply have to mention which dialect to use in the Hibernate configuration file.

The main disadvantage is performance. This automatic O-R mapping comes at a price. ORM tools are slower than writing the actual SQL queries. Besides, queries can be optimized better. However, since the application won’t have to work in a high transaction environment (for example, grameenphone’s account balance has to be changed every time someone makes a call), this probably won’t matter much.

3.

Spring JDBC Template is definitely a leaner alternative. In this application, SQL queries are manually written and executed using JDBC Template. Mapping them to the entity objects is manual as well. This is much leaner and performs much faster than Hibernate. Even if prepared statements are used, it’ll be lighter than Hibernate.

The idea of “optimize judiciously” means that optimizations should only performed when they’re clearly required and doesn’t compromise the design principles. Maintainability usually is more important than performance. Therefore, writing high performance code when they’re not required by compromising the design is a bad practice.

In the context of the GovForms application, it can be said that using Hibernate for O-R mapping is a good idea. This application doesn’t require extreme performance; therefore writing raw SQL queries is unhealthy.

4.

Preferring annotations to naming patterns is a good practice. If a programmer makes a spelling mistake in one of the naming patterns, the pattern will simply be ignored, but the program won’t generate any errors. This is extremely hard to detect and debug. Annotations, on the other hand avoids this issue. Spelling mistakes in annotation will generate compile error, and spelling mistakes in names won’t matter, because it’ll work perfectly.

In this project, this practice is followed in the following places:

DAO Implementations

In the controller classes

In the action methods inside the controller classes, for URL mapping

For automatic beans generation, component and repository annotations were used in the DAO interface and implementation we wrote.(DocumentDao, DocumentDaoImpl)

5.

Using List instead of array is a good practice. First of all, using array is considered a code smell, namely primitive obsession. Secondly, Lists are easy to manipulate, can be wrapped into thread safe wrapper, and lesser prone to developer’s mistake.

In the Data Access Object Implementations, sometimes Object[ ] is used. This can be replaced by using the toArray method in List.

Pretty much everywhere else, List is used.