Advanced Topics in Java

Prativa Nyaupane

Overview

- Overview of ORM
- Hibernate
- Web Framework Introduction
- Basics of Spring Boot
- Concurrency and Multithreading in JAVA
- Design Patterns: Singleton, Factory and Abstract Factory

Overview of ORM

- An ORM tool is software designed to help OOP developers interact with relational databases.
- So instead of creating your own ORM software from scratch, you cann make use of these tools.
- Simplifies the interaction between a relational database and the application by mapping database tables to objects.
- Removes the need for complex SQL queries, allowing developers to work with objects instead.

Hibernate

- A powerful Java-based ORM framework that simplifies database interactions.
- Provides an abstraction layer over JDBC, managing database connections and transactions.
- Reduces the need for manual SQL coding and database-specific handling, making database interactions more intuitive.
- Without Hibernate: Manually handling JDBC connections, SQL queries and result sets.
- With Hibernnate: Defining entities, mapping relationships and relying on Hibernate to handle database interactions.

What is a Framework

- A framework is a structure that you can build software on.
- It serves as a foundation, so you are not starting from scratch.
- Frameworks are typically associated with a specific programming language and are suited to different types of tasks.
- Let's say you're building a house:
  - You could pour the foundation and frame the house yourself. It would take a lot of time, but you could do it.
  - If all of that were already done for you, though, it would save you quite a bit of effor - especially if it was done by expert home builders.

- In software development, a framework serves a similar purpose. It's designed and tested by other Software Developers and Engineers, so you know its a solid foundation.

Web Framework Introduction

- A web framework, also known as a web application framework, is a software framework designed to assist in developing web applications, including web services, resources and APIs.
- The primary goal of web frameworks is to provide a standardized approach for creating and deploying web applications on the World Wide Web.
- Web frameworks aim to automate common tasks in web development to reduce the associated overhead.
- Examples of tasks handled by web frameworks include database access, templating and session management.

Web Framework Contd...

- The use of web frameworks promotes code reuse, making it easier to build and maintain web applications.
- While web frameworks are commonly associated with dynamic websites, they are also applicable to static websites.

What is Spring in Java

- Spring is a lightweight and popular open-source Java-based framework developed by Rod Johnson in 2003.
- It is used to develop enterprise-level applications.
- It provides support for to many other frameworks such as Hibernate.
- Spring framework can be used for several tasks, including:
  - Developing serverless applications.
  - Building scalable microservices
  - Securing the server-side of your application.
  - Automating tasks by creating batches.
  - An event-driven architecture.

Basics of Spring Boot

- Spring Boot simplifies the configuration and development of Spring applications.
- Promotes convention over configuration, reducing boilerplate code.
- Minimizes the complexities of setting up a Spring application and allows developers to focus on business logic.
- Without Spring Boot: Manually configuring various Spring components.

Benefit of Spring Boot:

- Spring Boot reduces development time, helps avoid unnecessary configuration and many other benefits, like:
  - We can use it to create standalone applications.
  - There is no need to deploy WAR files while using SpringBoot.
  - It doesn’t require XML configuration.
  - Embeds Tomcat, Jetty and Undertow directly.
  - Offers production-ready features.
  - SpringBoot is easier to launch.
  - Easier customization and management.

Concurrency and Multithreading in Java

- Enables parallel execution, improving application performance.
- Essential for developing responsive and efficient applications.
- Addresses the challenges of managing multiple threads, preventing race conditions and ensuring thread safety.
- Without Concurrency: Single-threaded applications may face performance bottlenecks.
- With Concurrency: Utilizing multiple threads to perform tasks concurrently.

Design Patterns: Singleton, Factory and Abstract Factory

- Design patterns are proven solutions to recurring design problems.
- Improves code readability, maintainability and scalability.
- Singleton: Ensures a class has only and provides a global point of access to it.
- Factory: Defines an interface for creating an object but leaves the choice of its type to the subclasses.
- Abstract Factory: Provides and interface for creating families of related or dependent objects without specifying their concrete classes.

Singleton Pattern

- The Singleton pattern ensures that only one instance of a class is created and provides a global point of access to it.
- It is useful in scenarios where we want to restrict the instantiation of a class to a single object throughout the application.
- Example:
  - Suppose we have a database connection class that needs to be access by multiple parts of our application.
  - We can use the Singleton pattern to ensure that the only instance of the database connection is created and shared across the application.
  - The Singleton class would typically have a static method that returns the single instance, and the constructor of the class would be made private to prevent direct instantiation.
  - This ensures that all parts of the applicationn access the same instance of the database connection, promoting efficient resource utilization and consistency.

Factory Pattern

- The Factory pattern provides an interface or base class for creating objects, but allows subclasses or implementing classes to decide which concrete class to instantiate.
- It abstracts the process of object creation and provides a way to create object without specifying their exact class.

Factory Pattern: Example

- Consider a pizza ordering application.
- The Factory pattern can be used to define a PizzaFactory class with a method for creating different types of pizzas. The PizzaFactory class would have methods like createMargheritaPizz(), createPepperoniPizza() and so on, each returning a specific type of pizza object.
- The client code can then use the PizzaFactory to create pizzas without being concerned about the specific pizza class being instantiated.
- This allows for flexibility and easy addition of new pizza types without modifying the client code.

Abstract Factory Pattern

- The Abstract Factory pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes.
- It allows the client code to work with abstract interfaces and classes making it independent of the specific implementations.

Abstract Factory Pattern: Example

- Suppose you are developing a GUI framework that supports multiple operating systems.
- You can use the Abstract Factory pattern to dfine an abstract GUIFactory interface with methods like createButton() and createCheckBox().
- Then, you can create concrete factory classes like WindowsGUIFactory and LinxuGUIFactory that implements platform-specific implementations of the createButton() and createCheckBox() implements the GUIFactory interface and provide methods.
- The client code can use the abstract GUIFactory interface to create buttons and checkboxes without knowing the specific platform it is running on. This promotes code modularity and easy integration with different operating system.

Thank You!