Spring Framework

Spring is the most popular application development framework for enterprise Java.

The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform.

Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE.

Spring is a lightweight framework. The spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. IOC makes the code loosely coupled. In spring framework, IOC container is responsible to inject the dependency.

Advantage of Dependency Injection

- makes the code loosely coupled so easy to maintain
- makes the code easy to test

Advantages of Spring Framework

1) Predefined Templates

2) Loose Coupling

3) Easy to test

4) Lightweight

5) Fast Development

6) Powerful abstraction

7) Declarative support

In spring, POJO’s (plain old java object) are called ‘beans’ and those objects instantiated, managed, created by Spring IOC container. Beans are created with the configuration metadata (XML file) that we supply to the container.

Spring supports given scope types for beans:

1) Singleton (a single instance per Spring IOC container (default))

2) Prototype

3) Request

4) Session

5) Global-session

Data Access / Integration

This group comprises of JDBC, ORM, OXM, JMS and Transaction modules. These modules basically provide support to interact with the database.

Web

This group comprises of Web, Web-Servlet, Web-Struts and Web-Portlet. These modules provide support to create web application.

There are two types of IOC containers. They are:

1. **Bean Factory**
2. **Application Context**

Spring JDBC Template is a powerful mechanism to connect to the database and execute SQL queries. It internally uses JDBC API, but eliminates a lot of problems of JDBC API.

Spring MVC:

1) Model - A model contains the data of the application. A data can be a single object or a collection of objects.

2) Controller - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.

3) View - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.

4) Front Controller - In Spring Web MVC, the Dispatcher Servlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

Spring Boot

Spring Boot is a spring module that provides the RAD (Rapid Application Development) feature to the spring framework.

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

It is a spring module that provides the **RAD (*Rapid Application Development*)** feature to the Spring Framework.

We should use Spring Boot Framework because:

- The dependency injection approach is used in Spring Boot.
- It contains powerful database transaction management capabilities.
- It simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts, etc.
- It reduces the cost and development time of the application.

Advantages of Spring Boot

- It creates **stand-alone** spring applications that can be started using Java **-jar**.
- It provides opinionated '**starter**' POMs to simplify our Maven configuration.
- There is no requirement for **XML** configuration.
- It offers the number of **plug-ins**.
- It **increases productivity** and reduces development time.

Spring Boot Features

- Web Development
- Spring Application
- Application events and listeners
- Admin features
- Externalized Configuration
- Properties Files
- YAML Support
- Type-safe Configuration
- Logging
- Security

Spring Boot Annotations

- @GetMapping
- @RequestMapping
- @PostMapping
- @PutMapping
- @DeleteMapping
- @RequestBody
- @RestController

JPA

The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems.

JPA Entity

Entity is a group of states associated together in a single unit. On adding behaviour, an entity behaves as an object and becomes a major constituent of object-oriented paradigm. So, an entity is an application-defined object in Java Persistence Library.

Collection Mapping

- List
- Set
- Map

JPA Type of Mapping:

- One to One
- One to Many
- Many to One
- Many to Many

JPA JPQL

The JPQL (Java Persistence Query Language) is an object-oriented query language which is used to perform database operations on persistent entities

JPQL is an extension of Entity JavaBeans Query Language (EJBQL)

- It can perform join operations.
- It can update and delete data in a bulk.
- It can perform aggregate function with sorting and grouping clauses.
- Single and multiple value result types.

Hibernate

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

Advantages of Hibernate Framework

- Open Source and Lightweight
- Fast Performance
- Database Independent Query

- Automatic Table Creation
- Simplifies Complex Join

Functionalities supported by Hibernate framework

- Hibernate framework support Auto DDL operations. In JDBC manually we have to create table and declare the data-type for each and every column. But Hibernate can do DDL operations for you internally like creation of table, drop a table, alter a table etc.
- Hibernate supports Auto Primary key generation. It means in JDBC we have to manually set a primary key for a table. But Hibernate can this task for you.
- Hibernate framework is independent of Database because it supports HQL (Hibernate Query Language) which is not specific to any database, whereas JDBC is database dependent.
- In Hibernate, Exception Handling is not mandatory, whereas In JDBC exception handling is mandatory.
- Hibernate supports Cache Memory whereas JDBC does not support cache memory.
- Hibernate is a ORM tool means it support Object relational mapping. Whereas JDBC is not object oriented moreover we are dealing with values means primitive data. In hibernate each record is represented as a Object but in JDBC each record is nothing but a data which is nothing but primitive values.

JPA vs Hibernate

- JPA is only a specification, it is not an implementation.
- It is a set of rules and guidelines to set interfaces for implementing object-relational mapping, .
- It needs a few classes and interfaces.
- It supports simple, cleaner, and assimilated object-relational mapping.
- It supports polymorphism and inheritance.
- Dynamic and named queries can be included in JPA.

The main feature of Hibernate is to map the Java classes to database tables. Following are some key features of Hibernate :

- Hibernate is an implementation of JPA guidelines.
- It helps in mapping Java data types to SQL data types.
- It is the contributor of JPA.

The Hibernate architecture is categorized in four layers.

- Java application layer
- Hibernate framework layer
- Backhand API layer
- Database layer

Elements of Hibernate Architecture

SessionFactory

The Session Factory is a factory of session and client of Connection Provider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

Session

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

Transaction

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

ConnectionProvider

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

TransactionFactory

It is a factory of Transaction. It is optional.

Hibernate Lifecycle

The Hibernate lifecycle contains the following states: -

- Transient state
- Persistent state
- Detached state

The databases supported by Hibernate

- DB2
- MySQL
- Oracle
- Sybase SQL Server
- Informix Dynamic Server
- HSQL
- PostgreSQL
- FrontBase