Spring Framework

What is Spring Framework?

Spring Framework is an open-source framework for building enterprise Java applications. It provides comprehensive infrastructure support for developing Java applications.

What are the main features of Spring Framework?

Dependency Injection, Inversion of Control (IoC), Aspect-Oriented Programming, Transaction Management, Spring MVC framework, Spring Data, Spring Security, Spring Boot, Spring JDBC, Testing Support, and Integration with various frameworks and technologies.

What is Dependency Injection (DI)?

Dependency Injection is a design pattern that allows an object to receive its dependencies from an external source rather than creating them itself, Dependency Injection promotes loose coupling. we can do two types of Dependency Injection in spring - Constructor Injection and Setter Injection.

What is a Spring Bean?

A Spring Bean is an object that is instantiated, managed, and configured by the Spring IoC container.

What is the Spring IoC container?

The IoC container is responsible for managing the lifecycle and configuration of application objects through Dependency Injection.

What are the different types of IoC containers in Spring?

BeanFactory and ApplicationContext.

What is a Spring BeanFactory?

BeanFactory is like a factory class that contains a collection of beans. It creates the bean whenever clients request it.

What is a Spring ApplicationContext?

ApplicationContext is a central interface for providing configuration information to an application. It extends BeanFactory and provides more enterprise-specific functionality.

What is Spring AOP?

Spring AOP is a powerful technique for improving code modularity and reusability. It breaks the program logic into distinct parts/ concerns. It is used to increase modularity by cross-cutting concerns.

The difference between AOP & OOP is AOP focuses on the separation of concerns while OOP focuses on the encapsulation of data and behavior into objects

What is Spring Data JPA?

Spring Data uses JPA to store data in a relational database. it makes it easier to work with a database and perform CRUD actions. Its most attractive feature is that it can automatically create repository implementations at runtime from a repository interface.

What is Spring Security?

Spring Security is a framework that provides authentication, authorization, and other security features for enterprise applications.

What is Spring MVC?

Spring MVC is a module of Spring for building web applications following the Model-View-Controller design pattern.

What is a DispatcherServlet in Spring MVC?

DispatcherServlet is the front controller in Spring MVC that handles all incoming HTTP requests and delegates them to appropriate handlers.

What is Spring Cloud?

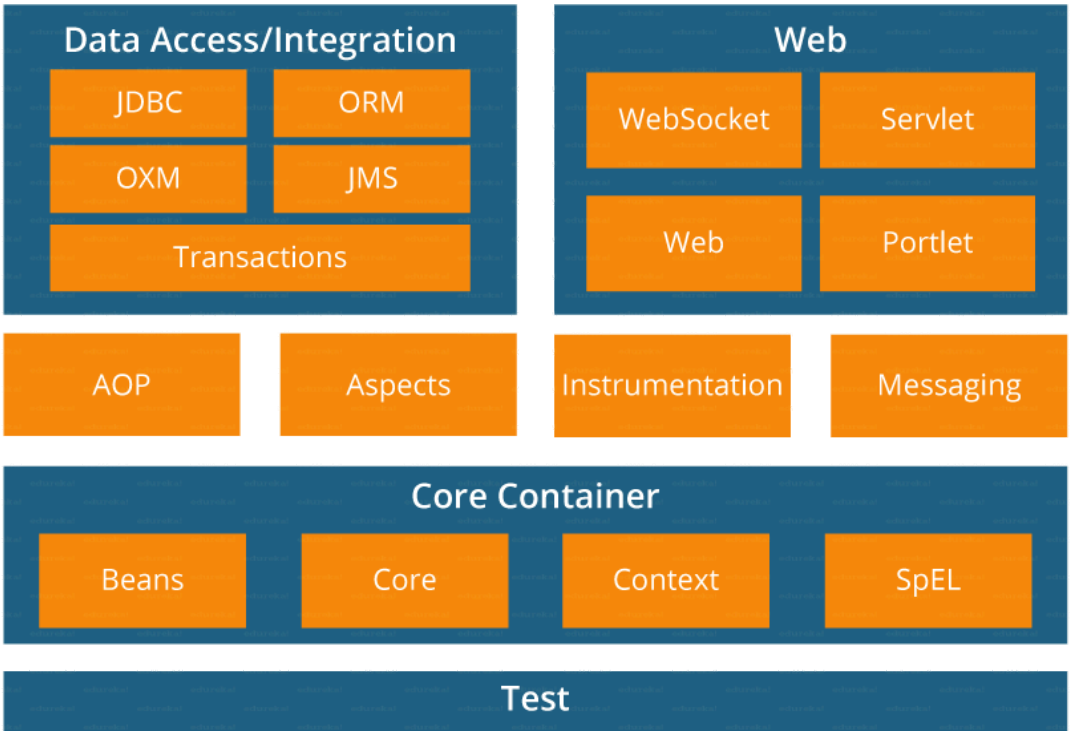
Spring Cloud provides tools for building and managing distributed systems, including microservice architectures.

What is a Spring Bean's lifecycle?

The lifecycle includes instantiation, dependency injection, initialization (post-construct), use, and destruction (pre-destroy).

How many modules are there in Spring Framework and what are they?

There are around 20 modules. Shown in the image.



What is a Spring configuration file?

A Spring configuration file is an XML file. This file mainly contains the class information. It explains how those classes are set up and connected.

What is ContextLoaderListener and what does it do?

The ContextLoaderListener is a Spring Framework component used in web applications to load the Spring application context when the web application starts. It initializes the IoC container, manages beans, and allows components to access Spring beans for their processing.

What is the importance of the web.xml in Spring MVC?

The web.xml file in Spring MVC is used for configuring the DispatcherServlet, defining context parameters, filters, and listeners, as well as handling error pages.

Spring Boot

What is Spring Boot?

Spring Boot is a framework that simplifies the development of Spring-based applications by providing defaults for code and annotation configuration.

How to create a Spring Boot application?

We can create a Spring Boot application using Spring Initializr, Spring Boot CLI, or by manually setting up a Maven/ Gradle project.

How does Spring Boot differ from the traditional Spring Framework?

Spring	Spring boot
Spring is an open-source lightweight framework widely used to develop enterprise applications.	Spring Boot is built on top of the Spring framework, widely used to develop REST APIs.
The most important feature of the Spring Framework is dependency injection.	The most important feature of the Spring Boot is Autoconfiguration.
It helps to create a loosely coupled application.	It helps to create a stand-alone application.
To run the Spring application, a deployment descriptor is required.	There is no requirement for a deployment descriptor(web.xml).
To create a Spring application, the developers write lots of code.	It reduces the lines of code.
It doesn't provide support for the in-memory database.	It provides support for the in-memory database such as H2.
Developers have to define dependencies manually in the pom.xml file.	pom.xml file internally handles the required dependencies.

What is auto-configuration in Spring Boot?
Auto-configuration automatically configures our Spring application based on the dependencies present in the classpath.

What are the main features of Spring Boot?

- Auto-configuration, starter dependencies, embedded servers, Spring Boot CLI, Actuator for monitoring, and Spring Initializr.

What is SpringBoot annotation?

- ➔ **@SpringBootApplication**: Indicates a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning.
- ➔ **@RestController**: This annotation is used to simplify the creation of RESTful web services. It combines @Controller and @ResponseBody.
- ➔ **@RequestMapping**: It maps certain web requests to their specific controller and methods. It can also be used to take query parameters from the URL.
    - ◆ **@GetMapping**: It is used for mapping HTTP GET requests onto specific handler methods.
    - ◆ **@PostMapping**: It is used to map HTTP POST requests onto specific handler methods.
    - ◆ **@PutMapping**: It is used for mapping HTTP PUT requests onto specific handler methods
    - ◆ **@DeleteMapping**: It is used to map HTTP DELETE requests onto specific handler methods
    - ◆ **@PatchMapping**: It is used to map HTTP PATCH requests onto specific handler methods
- ➔ **@Autowired**: It helps us to inject the object dependency implicitly. It internally uses a setter or constructor injection.
- ➔ **@Component**: It marks a class as a Spring component, allowing Spring to detect and register it as a bean.
- ➔ **@Service**: It is used to mark the class as a service provider. It is used to write business logic.
- ➔ **@ConfigurationProperties**: It binds the properties from configuration files to Java objects, allowing type-safe configuration.
- ➔ **@Controller**: The @Controller annotation indicates that a particular class serves the role of a controller.
- ➔ **@Qualifier:** This annotation is useful when we want to specify which bean of a certain type should be injected by default.

- ★ **@Repository**: It is used to manage data persistence and retrieval in a database. It indicates a DAO (Data Access Object) component.
- ★ **@Configuration**: Indicates a class that contains bean definitions.
- ★ **@Bean**: Indicates a method that produces a bean to be managed by Spring.
- ★ **@Entity**: It defines that a class can be mapped to a table.
- ★ **@Table**: Specifies the table name in the database for a JPA entity.
- ★ **@Id**: Specifies the primary key of an entity.
- ★ **@GeneratedValue**: Specifies the generation strategy for the primary key.
- ★ **@EnableAutoConfiguration**: Enables Spring Boot's auto-configuration mechanism.
- ★ **@Transactional**: Transactional annotation manages transactions in a Spring boot application and is used to define a scope of transaction.

- ➔ **@EnableScheduling**: Enables Spring's scheduled task execution capability.
- ➔ **@EnableAsync**: Enables Spring's asynchronous method execution capability.
- ➔ **@Value**: Injects property values into fields, methods, or constructor parameters.
- ➔ **@ConditionalOnProperty**: Configures beans to be loaded conditionally based on a specific property.
- ➔ **@SpringBootTest**: Used for writing integration tests, and bootstrapping the entire container.
- ➔ **@TestConfiguration**: Specifies additional test-specific configuration classes.

- ★ **@RestControllerAdvice**: Provides global exception handling, data binding, and validation advice for REST controllers.
- ★ **@PathVariable**: It is used to retrieve data from the URL path and bind them to method parameters.
- ★ **@RequestParam**: It is used to read the form data and bind it automatically to the parameter present in the provided method.

- ★ **@RequestBody**: Binds the body of an HTTP request to a method parameter.
- ★ **@ResponseBody**: Indicates that a method return value should be bound to the web response body.
- ★ **@ModelAttribute**: It binds method parameters or method return values to model attributes. It plays a crucial role in the Model-View-Controller (MVC) architecture, where it helps transfer data between the Controller and the View.

What is a Spring Boot Starter?

Starters are a simple way to describe the dependencies we need to include in our project to get started quickly with common frameworks and libraries.

How to configure properties in Spring Boot?

Using application.properties or application.yml files.

What is a Spring Boot Actuator?

Spring Boot Actuator is a module that provides production-ready features to monitor and manage our Spring Boot application. To enable it add the dependency spring-boot-starter-actuator and configure it in application.properties.

What is a Spring Boot actuator endpoint?

Actuator endpoints are URLs that show important details about how the application is running, such as health, metrics, and environment properties.

How to deploy a Spring Boot application to an external server?

Package the application as a WAR file and deploy it to the external server.

What is Spring Initializr?

Spring Initializr is a web-based tool to generate Spring Boot project structure with the necessary dependencies.

How to handle exceptions in Spring Boot?

Exceptions can be handled using @ControllerAdvice and @ExceptionHandler annotations or globally via ErrorController.

What is DevTools in Spring Boot?

DevTools is a set of tools that helps in development by providing features like automatic restarts, live reload, and configurations for better productivity.

What is a Spring Boot Profile?

It allows us to create different setups for different situations or uses.

How does Spring Boot handle logging?

Spring Boot uses Apache Commons Logging internally and provides support for Logback, Java Util Logging, Log4J2, and more.

How to manage dependencies in Spring Boot?

Dependencies are managed using Maven or Gradle, often through the use of Spring Boot starter dependencies.

What is the spring-boot-starter-parent?

It is a special starter project that provides default configurations for our application and a complete dependency tree to quickly build our Spring Boot project.

How to secure a Spring Boot application?

Security can be implemented using Spring Security, configured via dependencies, and annotations like @EnableWebSecurity.

What is Spring Boot CLI?

Spring Boot CLI is a Spring Boot software to run and test Spring Boot applications from the command prompt. When we run Spring Boot applications using CLI, then it internally uses Spring Boot Starter and Spring Boot @autoconfiguration components to resolve all dependencies and execute the application.

How to run a Spring Boot application on a different port?

By setting the server.port property in the application.properties file or as a command-line argument.

How to integrate Spring Boot with a database?

By including the appropriate starter, such as spring-boot-starter-data-jpa, configuring the data source properties, and using Spring Data repositories.

How to test a Spring Boot application?

Using Spring Boot's testing support with @SpringBootTest, @WebMvcTest, and other annotations for different testing scenarios.

What is the application.properties file?

It is a configuration file used to define application-level settings, such as database configuration, server port, and custom properties.

How does Spring Boot support microservices?

Spring Boot supports microservices through a variety of features and components like Simple Configuration, Embedded Servers, Spring Cloud Integration, API Gateway, Security, Docker and Kubernetes Support, testing, etc.

Hibernate

What is Hibernate?

Hibernate is an open-source Object-Relational Mapping (ORM) framework for Java. It simplifies database interactions by mapping Java objects to database tables.

What are the main advantages of using Hibernate?

- It's fast
- It's lightweight and open-source
- Reduces boilerplate code
- Supports transparent persistence
- Database independence

- Provides HQL (Hibernate Query Language)
- Automatic table creation

What is an ORM tool?

ORM stands for Object-Relational Mapping. This tool maps objects to database tables, handling CRUD operations and transactions, reducing the need for manual SQL queries.

What is a SessionFactory in Hibernate?

SessionFactory is a thread-safe, heavyweight object that creates Session instances for database operations. It is created once per application.

What is a Hibernate Session?

A Session is a lightweight, non-thread-safe object used to interact with the database. It handles CRUD operations and acts as a cache for persistent objects.

What is lazy loading in Hibernate?

Lazy loading is a performance optimization where related data is loaded only when it is accessed for the first time, rather than at the initial database query.

What is HQL?

HQL stands for Hibernate Query Language. It is an object-oriented query language similar to SQL but operates on Java objects instead of database tables.

What is the difference between get() and load() methods in Hibernate?

- get(): Returns null if the object is not found; hits the database immediately.
- load(): Throws an exception if the object is not found; returns a proxy and does not hit the database until the object is accessed.

What is the purpose of the Hibernate configuration file?

The configuration file (hibernate.cfg.xml) defines database connection settings, mapping files, and other configuration properties.

What is a Hibernate mapping file?

mapping file is an XML file used to define the mapping between Java classes and database tables in Hibernate. It specifies how the properties of a Java class correspond to the columns of a database table.

What is caching in Hibernate?

Caching improves application performance by storing frequently accessed data in memory. Hibernate uses two levels of caching to optimize performance:

- First-level cache: This is Session-specific, enabled by default.
- Second-level cache: This is SessionFactory-specific, and requires configuration. But this is optional.

What is JPA?

Java Persistence API (JPA) is a Java specification that provides specific functionality and is a standard for ORM tools.

What are the common annotations used in Hibernate?

- @Entity: Defines the class as an entity.
- @Table: Specifies the table name.
- @Id: Specifies the primary key.
- @GeneratedValue: Specifies how the primary key should be generated.
- @Column: Defines the column details.
- @OneToOne: One entity instance is associated with one instance of another entity.
- @OneToMany: One entity instance is associated with many instances of another entity.
- @ManyToOne: Many entity instances are associated with one instance of another entity.
- @ManyToMany: Many entity instances are associated with many instances of another entity.

How to handle transactions in Hibernate?

Transactions are managed using the Transaction interface, typically within a Session context. Follow the below step-by-step.

1. begin(): It starts a new transaction.
2. commit(): It ends the transaction and flushes the associated session.
3. rollback(): It rolls back the current transaction.
4. setTimeout(int seconds): It specifies the transaction timeout for any subsequent calls to begin() on this instance.
5. isActive(): It determines whether or not the transaction is still active.
6. registerSynchronization(Synchronization s): registers a user synchronization callback for this transaction.
7. wasCommited(): checks if the transaction is committed successfully.
8. wasRolledBack(): checks if the transaction is rolled-back successfully.

Why is Hibernate better than Java Database Connectivity (JDBC)?

1. Hibernate code is cleaner and more readable thanks to the elimination of boiler-plate code, something found in JDBC
2. Unlike JDBC API, Hibernate supports associations, collections, and inheritances
3. HQL (Hibernate Query Language) is closer to Java and is more object-oriented
4. Developers don't need to write code to store and load data into the database
5. Hibernate enables faster application development

Name the four ORM levels in Hibernate.

- Full Object Mapping
- Light Object Mapping
- Medium Object Mapping
- Pure Relational

What is "dirty checking"?

The dirty checking feature helps developers and users avoid time-consuming write actions, thereby reducing database write times.

What is Hibernate's default cache service?

Hibernate's default cache service is EHCache, though the framework additionally supports OSCache, SWARMCache, and TreeCache.

What are the concurrency strategies?

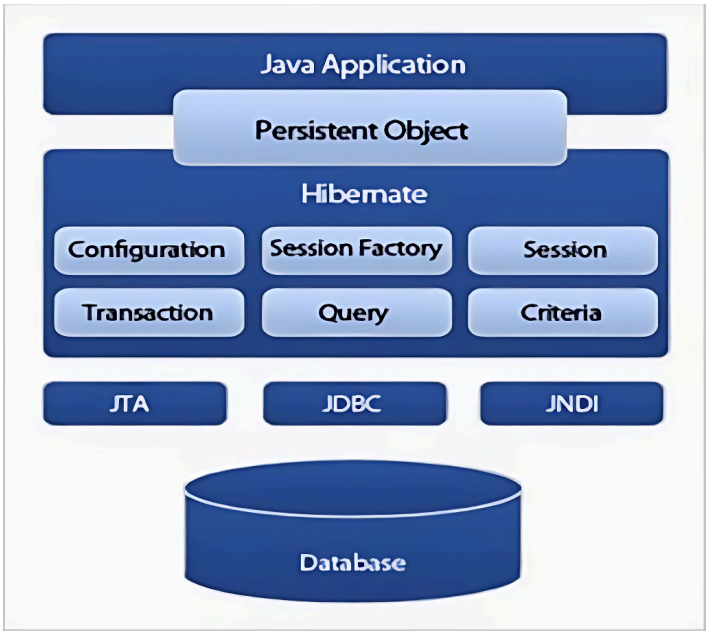
Concurrency strategies are mediators responsible for storing and retrieving cached items. When enabling a second-level cache, the developer must decide which cache concurrency to implement for each persistent class and collection.

1. Non-strict-Read-Write: This strategy works with data that can be altered and can tolerate a small chance of stale data. This strategy offers no guarantee of consistency between the database and the cache.
2. Read-Only: This strategy works best with data that can't be changed, and consequently, is only used to reference data.
3. Transactional: This strategy is used primarily for read-mostly data in cases where it's essential to prevent stale data in concurrent transactions, in those rare instances of an update.
4. Read-Write: This strategy is like the transactional strategy.

What is Hibernate Validator framework?

Hibernate Validators offer field-level validation for every attribute of a bean class like null/not null, empty/not empty, min/max valid email, and valid credit card.

Hibernate architecture



What does Session.lock() method in hibernate do?

The SessionSession.lock() method is used to acquire a lock on an object in the current Session.

What is the criteria API in hibernate?

The criteria API is a programmatic way of creating and executing queries. It allows developers to build complex queries using a fluent interface rather than writing raw SQL.

Does Hibernate support Native SQL Queries?

Yes, Hibernate supports Native SQL Queries, which allow us to use SQL statements directly to interact with the database. This can be useful when we want to perform complex queries that are impossible with HQL or Criteria API.

Explain states that a persistent entity exists in.

1. Transient: When an object is first instantiated using the new operator, it is in the transient state.
2. Persistent: Once a transient object is associated with a Hibernate session (via session.save() or session.persist()), it enters the persistent state.
3. Detached: An object transitions to the detached state when its associated Hibernate session is closed, or the object is explicitly detached using session.evict() or session.clear().
4. Removed State: When an object is marked for deletion from the database using session.delete(), it enters the removed state.

What is Named SQL Query?

Named SQL Query is a feature in Hibernate that allows us to define a named query and reuse it throughout our application. It is defined in the mapping file or by using the @NamedQuery annotation.

What is N+1 SELECT problem in Hibernate? How to solve?

Problem: This arises when Hibernate executes additional database queries to fetch associated entities or collections for each entity fetched initially. This can lead to a significant performance overhead, especially when dealing with large datasets.

Solution: The N+1 SELECT problem can be solved using fetching strategies such as eager loading/ fetch join, lazy loading/explicit fetching, or batch fetching. Another solution is to use the JOIN FETCH clause in HQL or JPQL Query.