Spring Tutorial

Last Updated : 27 Aug, 2025

Spring Framework is a powerful and flexible platform for enterprise Java development. It provides a wide range of features that simplify application development and make projects more scalable, secure, and maintainable.

Key Features:

- **Spring MVC** – A robust web framework for building dynamic web applications.
- **Spring Boot** – Simplifies application setup and configuration, enabling rapid development.
- **Spring Security** – Offers strong authentication and authorization features.
- **Spring Data** – Simplifies database operations and ORM integration.
- **Spring Cloud** – Provides tools for building microservices-based architectures.

Spring integrates seamlessly with other popular frameworks like Hibernate, Struts, EJB, JSF, and more, which is why it's often called a **"framework of frameworks."** It also comes with multiple modules such as IoC, AOP, DAO, Context, and Web MVC.

Why Use Spring?

- **POJO-Based Development:** You can build enterprise-level applications using simple Plain Old Java Objects (POJOs). This avoids the need for heavy EJB containers; a servlet container like Tomcat is often enough.
- **Modular Design:** Spring is highly modular, so you only use the parts you need without unnecessary overhead.

- **Framework Integration:** Works well with existing technologies such as Hibernate, Quartz, JDK timers, and other view frameworks.
- **Easy Testing:** Its DI and POJO-based approach make testing straightforward, allowing easy injection of mock data for test cases.
- **Web MVC Framework:** Provides a clean, flexible, and well-structured MVC framework, often preferred over heavier alternatives like Struts.
- **Centralized Exception Handling:** Converts framework-specific exceptions (JDBC, Hibernate, etc.) into consistent unchecked exceptions, simplifying error handling.
- **Lightweight Containers:** The IoC containers are lightweight compared to EJB containers, making Spring ideal for applications that need efficiency in memory and CPU usage.

Basics

- Introduction
- Architecture
- Reasons to Use
- Spring Initializr
- Spring vs. Struts in Java

Software Setup and Configuration (STS/Eclipse/IntelliJ)

Getting started with Spring development requires setting up an integrated development environment (IDE). This section guides you through installing and configuring popular IDEs like Spring Tool Suite (STS), Eclipse, and IntelliJ IDEA, ensuring you have the right tools for building Spring applications.

- Download and Install
- Create and Setup Spring Boot Project
- Project with IntelliJ IDEA
- Setup in Eclipse IDE
- Run First Spring Boot Application

Core Spring

Core Spring focuses on the fundamental principles of Inversion of Control (IoC) and Dependency Injection (DI). These concepts are crucial for decoupling components and managing object lifecycles. This section explores how Spring's IoC container and DI mechanisms work, providing examples and explanations.

- Create a Simple Spring Application
- Inversion of Control
- BeanFactory
- ApplicationContext
- Dependency Injection
- Inversion of Control vs Dependency Injection
- Injecting Objects By Constructor Injection
- Setter Injection with Map
- Dependency Injection by Setter Method
- Setter Injection with Non-String Map
- Constructor Injection with Non-String Map
- Bean life cycle in Java Spring
- Custom Bean Scope in Spring
- IoC Container
- Configure Dispatcher Servlet in web.xml File
- Expression Language(SpEL)
- Difference Between RowMapper and ResultSetExtractor

Spring Annotations

Annotations in Spring simplify configuration and setup by providing metadata about our components. This section deep dives into the various annotations available in Spring, such as @Component, @Service, @Repository, and more, explaining how they help streamline development.

- Framework
- Core
- Stereotype

- @Bean
- @Controller
- @Value
- @Configuration
- @ComponentScan
- @Qualifier
- @Service
- @Repository
- @Required
- @Component
- @Autowired
- @Scope
- @Required

Spring Boot

Spring Boot is an extension of the Spring Framework that simplifies the development of new applications. It provides a range of out-of-the-box configurations and defaults, embedded servers, and streamlined deployment options, making it easier and faster to build production-ready applications.

- Introduction
- Create Project
- Run First Spring Boot Application
- Architecture
- Application Properties
- Dependency Management
- Starters
- Hello World Example
- REST Example
- Starter Test
- Spring JDBC vs Spring Data JDBC
- Exception Handling
- Actuator

To know more about Spring Boot, refer to - Spring Boot Tutorial

Spring MVC

Spring MVC is a framework within Spring that helps you build robust and maintainable web applications using the Model-View-Controller design pattern. This section covers everything from setting up a simple controller to advanced topics like custom validation and integration with databases.

- Introduction
- Java-based configuration
- Create and RunFirst MVC Controller
- Create First Model
- Create First View
- ViewResolver
- RequestParam
- Spring @RequestMapping
- @Controller vs @RestController
- Exception Handling
- Capture Data using @RequestParam
- Project For Finding Doctors Online with MySQL
- Custom Validation
- File Upload
- Integration with MySQL
- CRUD

To know more about Spring MVC, refer to - Spring MVC Tutorial

Spring with REST API

Spring's support for RESTful web services allows developers to create robust APIs that can be consumed by various clients. This section introduces the basics of REST, how to create controllers, and how to handle different types of data formats like JSON and XML.

- REST Controller

- RESTful Web Services
- Simple RestController
- REST API using Spring Boot
- REST JSON Response
- REST XML Response
- JSON using Jackson in REST API Implementation with Spring Boot
- RestTemplate

Spring Data JPA

Spring Data JPA makes it easy to implement JPA-based repositories with minimal boilerplate code. This section explores how to perform **CRUD operations**, use custom queries, and manage database transactions using Spring Data JPA.

- Introduction
- Find Records From MySQL
- Delete Records From MySQL
- @Table Annotation
- Insert Data in MySQL Table
- Attributes of @Column Annotation with Example
- @Column Annotation
- @Id Annotation
- Access database
- Project

Spring JDBC

Spring JDBC provides a straightforward way to interact with databases, offering a simpler alternative to ORM frameworks like Hibernate. This section covers the basics of using JdbcTemplate, named parameters, and handling SQL exceptions.

- Template
- Example
- SimpleJDBCTemplate

- Prepared Statement
- NamedParameterJdbcTemplate
- Using SQL Scripts with Spring JDBC + JPA + HSQLDB
- ResultSetExtractor

Spring ORM or Spring Hibernate

Spring ORM integrates with Hibernate to provide a robust solution for **object-relational mapping** in Java applications. This section covers the setup, configuration, and use of Hibernate with Spring, including mappings, lifecycle management, and query handling.

- Configuration and Creating a Table in the Database
- JPA vs Hibernate
- Spring ORM Example using Hibernate
- Lifecycle
- Validation using Hibernate Validator
- CRUD
- One-to-One Mapping
- Many-to-One Mapping
- One-to-Many Mapping
- Many-to-Many Mapping
- Eager/Lazy Loading
- @Embeddable and @Embedded Annotation
- Example using JPA and MySQL
- Automatic Table Creation
- get() and load() Method
- Pagination
- Batch Processing

To know more about Hibernate, refer to - *Hibernate Tutorial*

Spring AOP

Aspect-Oriented Programming (AOP) in Spring helps in separating cross-cutting concerns, such as logging, security, and transaction management, from the business logic. This section introduces AOP concepts, usage of aspects, and how to implement them in Spring applications.

- Introduction
- AOP Around Advice
- Implement AOP
- Example (Spring1.2 Old Style AOP)
- AspectJ Xml Configuration
- AspectJ Annotation
- Cache Provider
- Around Advice
- After Advice
- Before Advice

Spring Security

Spring Security is a powerful and customizable authentication and access control framework for Java applications. This section covers the core features, including setting up basic security, customizing authentication, and managing user sessions.

- Introduction
- Important Terms
- OAuth2 Authentication
- Method Level
- JSP Tag Library
- Form-Based Authentication
- Remember Me
- Authentication and Authorization in Spring Boot 3.0

To know more about Spring Security, refer to - *Spring Security Tutorial*

Interview Questions on Spring Framework

- Spring Interview Questions and Answers
- Java Spring MCQ & Quiz

Comment | More info | Advertise with us

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305