AKTU · 2ND YEAR · SEMESTER 4 · BCS403

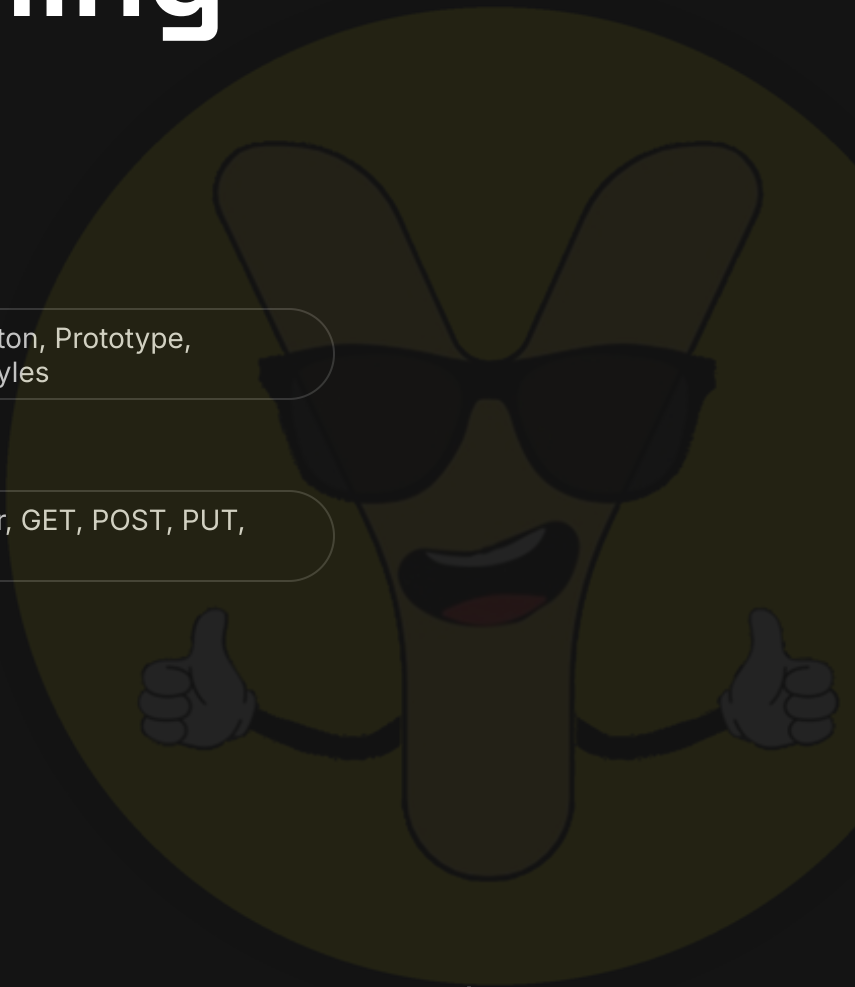
Object Oriented Programming with Java

Unit 5 — Spring Framework and Spring Boot

Spring Core Basics: Spring Dependency Injection concepts, Spring Inversion of Control, AOP, Bean Scopes (Singleton, Prototype, Request, Session, Application, Web Socket), Auto wiring, Annotations, Life Cycle Call backs, Bean Configuration styles

Spring Boot: Spring Boot Build Systems, Spring Boot Code Structure, Spring Boot Runners, Logger

Building RESTful Web Services: Rest Controller, Request Mapping, Request Body, Path Variable, Request Parameter, GET, POST, PUT, DELETE APIs, Build Web Applications



DECODED BY

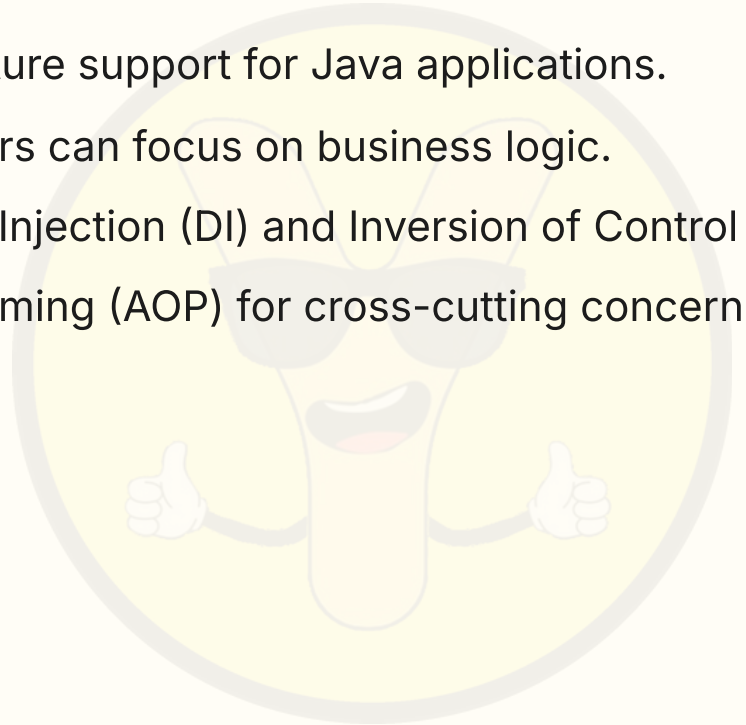
Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

Introduction to Spring Framework

Spring is a lightweight, open-source framework for building enterprise-grade Java applications.

- Provides comprehensive infrastructure support for Java applications.
- Focuses on 'plumbing' so developers can focus on business logic.
- Core features include Dependency Injection (DI) and Inversion of Control (IoC).
- Supports Aspect-Oriented Programming (AOP) for cross-cutting concerns.



IoC and Dependency Injection

INVERSION OF CONTROL (IOC)

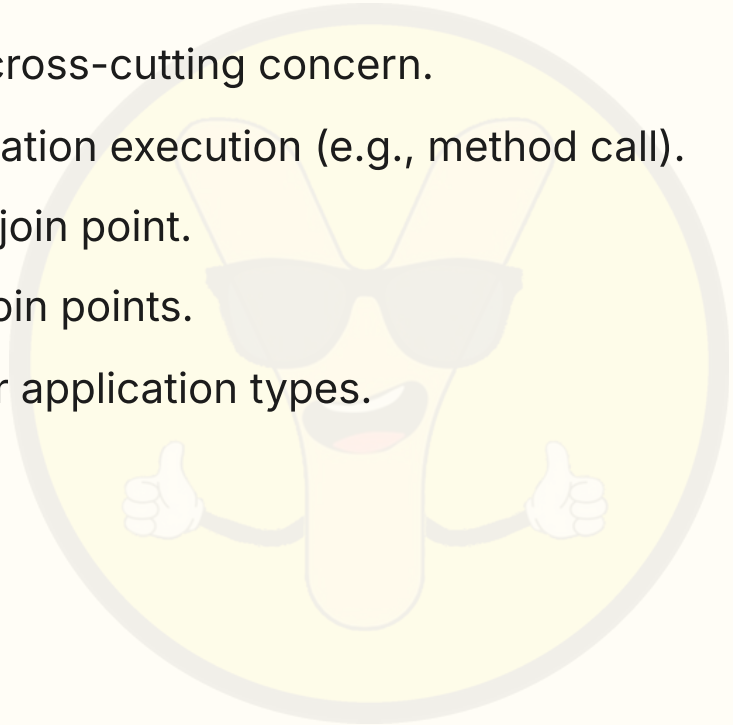
A design principle where the control of object creation and lifecycle is transferred from the application code to the Spring Container.

- Dependency Injection (DI) is the primary technique to implement IoC.
- Objects define their dependencies through constructors or setter methods.
- The Container injects these dependencies at runtime.
- Reduces tight coupling between classes.

Aspect-Oriented Programming (AOP)

AOP modularizes cross-cutting concerns like logging, security, and transaction management.

- Aspect: A module encapsulating a cross-cutting concern.
- Join Point: A specific point in application execution (e.g., method call).
- Advice: Action taken at a particular join point.
- Pointcut: Expression that matches join points.
- Weaving: Linking aspects with other application types.



Spring Bean Scopes

Scope	Description
Singleton	Single instance per Spring IoC container (Default).
Prototype	New instance created every time requested.
Request	Single instance per HTTP request (Web aware).
Session	Single instance per HTTP session.
Application	Single instance per ServletContext.

Bean Configuration and Lifecycle

Beans are configured via XML, Annotations, or Java-based configuration.

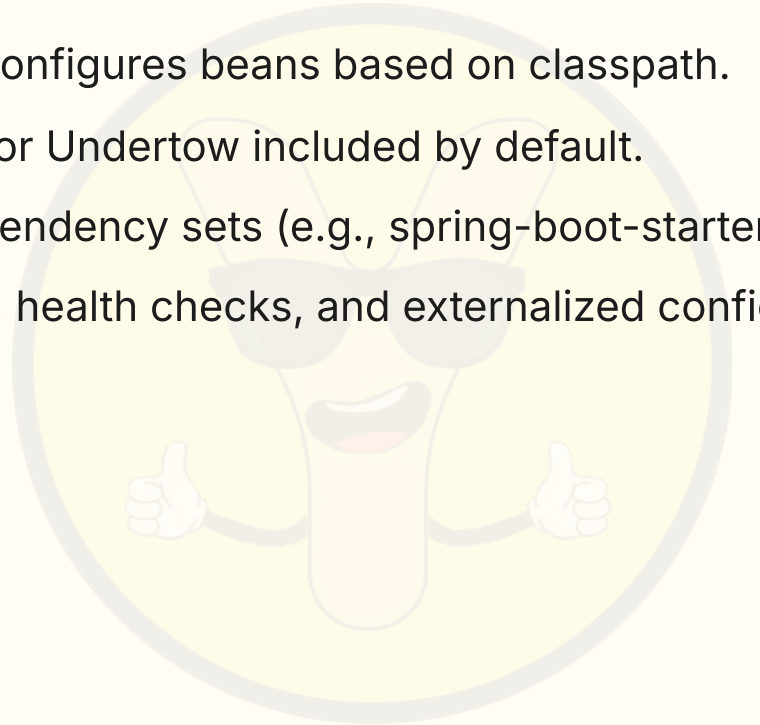
- Auto-wiring: Spring automatically resolves dependencies using @Autowired.
- Lifecycle Callbacks: @PostConstruct (initialization) and @PreDestroy (destruction).
- XML Config: Defined in applicationContext.xml.
- Java Config: Uses @Configuration and @Bean annotations.
- Annotation Config: Uses @Component, @Service, @Repository.



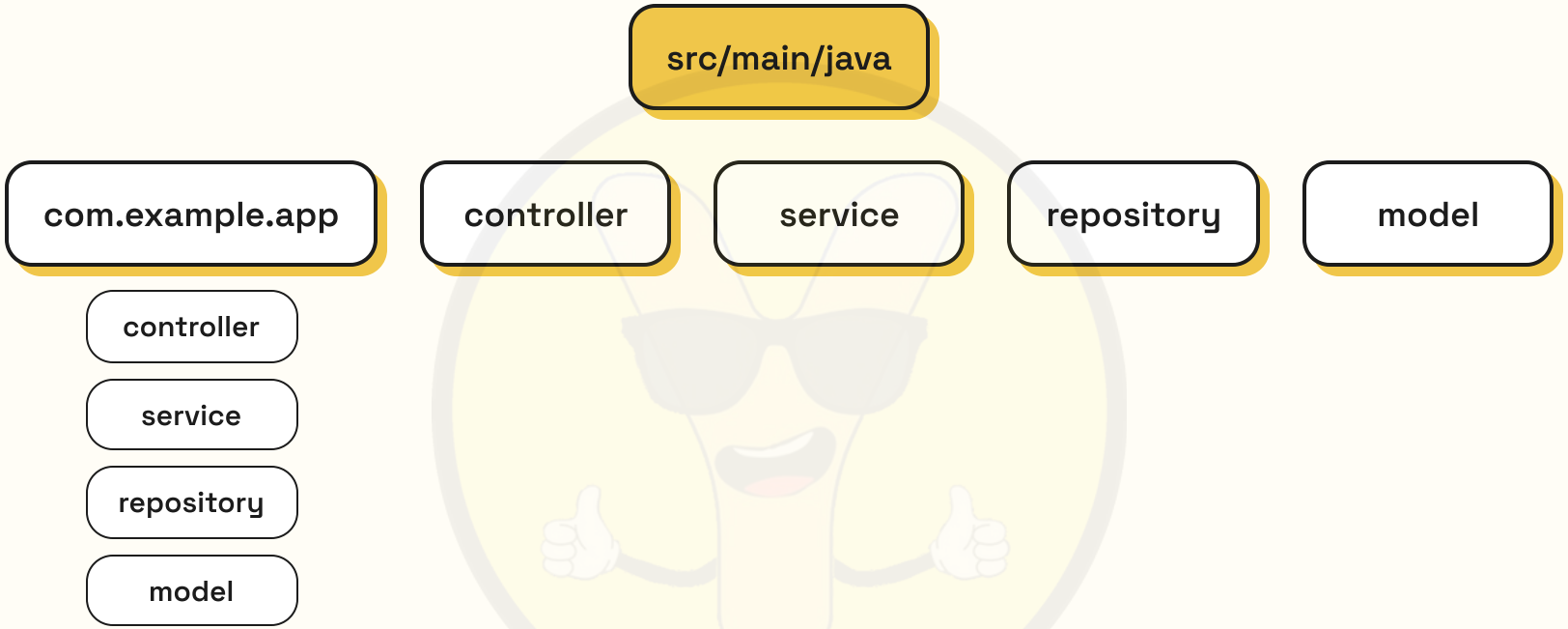
Introduction to Spring Boot

Spring Boot simplifies the development of production-ready Spring applications.

- Auto-configuration: Automatically configures beans based on classpath.
- Embedded Servers: Tomcat, Jetty, or Undertow included by default.
- Starter Dependencies: Curated dependency sets (e.g., spring-boot-starter-web).
- Production-ready features: Metrics, health checks, and externalized configuration.



Spring Boot Code Structure

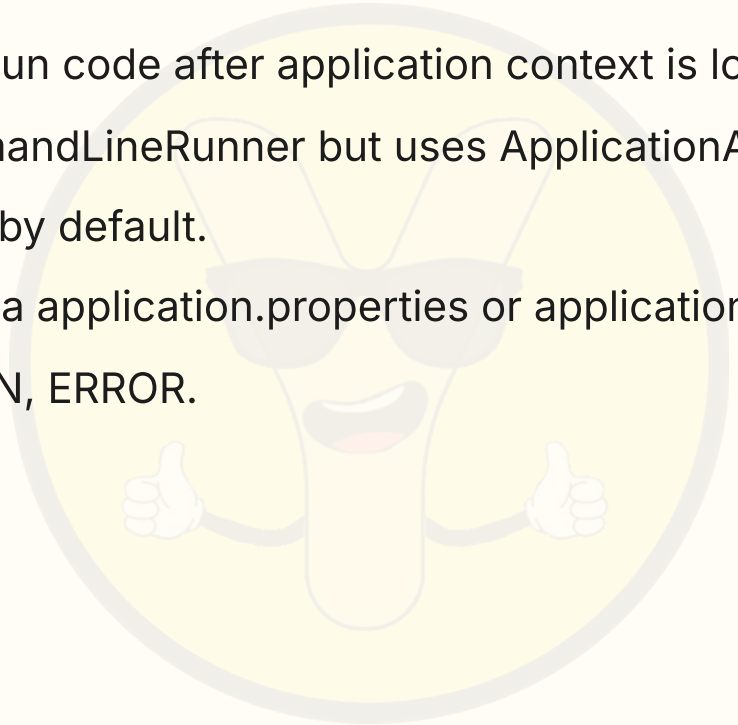


Standard Maven/Gradle project structure for Spring Boot applications.

Runners and Logging

Spring Boot provides mechanisms to execute code on startup and manage logs.

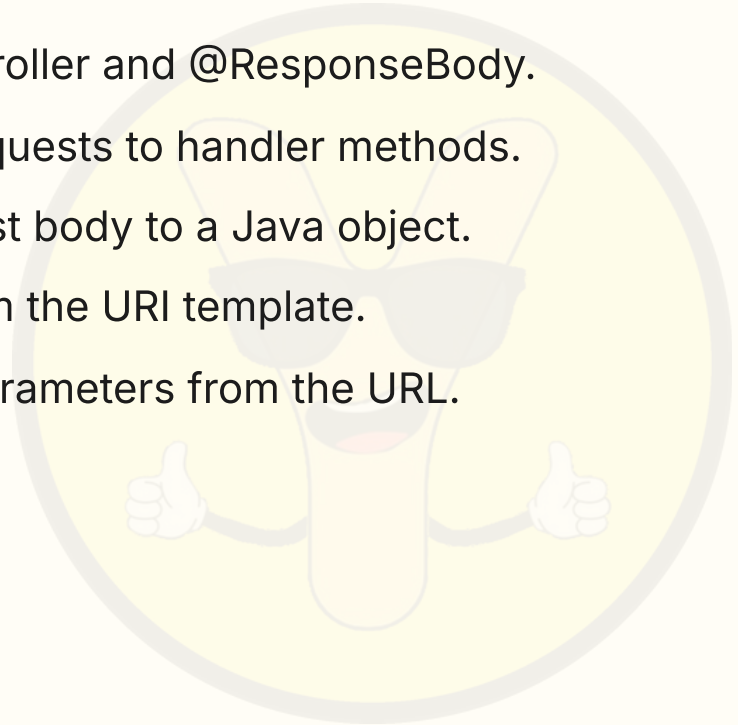
- CommandLineRunner: Interface to run code after application context is loaded.
- ApplicationRunner: Similar to CommandLineRunner but uses ApplicationArguments.
- Logging: Uses SLF4J and Logback by default.
- Configuration: Logs are managed via application.properties or application.yml.
- Levels: TRACE, DEBUG, INFO, WARN, ERROR.



Building RESTful Web Services

REST (Representational State Transfer) uses standard HTTP methods for resource manipulation.

- @RestController: Combines @Controller and @ResponseBody.
- @RequestMapping: Maps HTTP requests to handler methods.
- @RequestBody: Binds HTTP request body to a Java object.
- @PathVariable: Extracts values from the URI template.
- @RequestParam: Extracts query parameters from the URL.

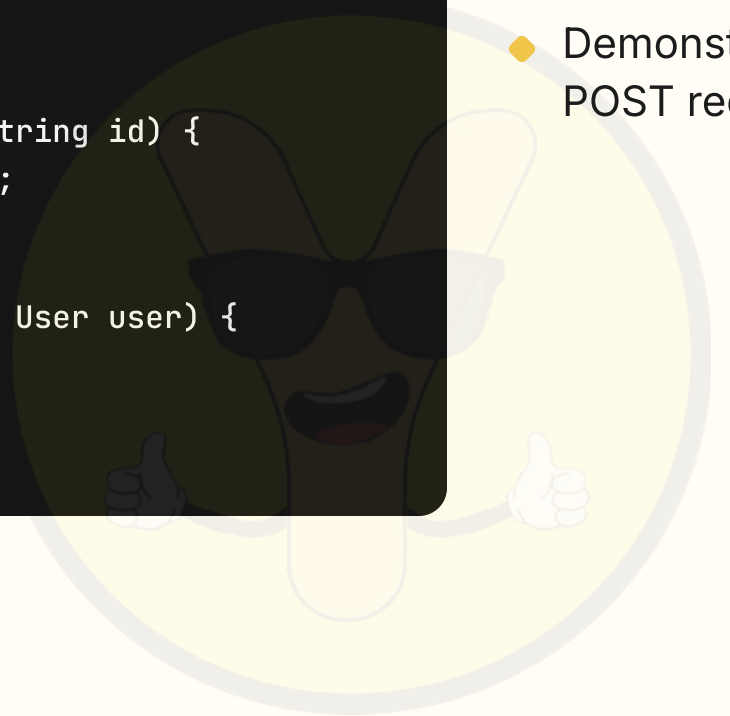


HTTP Methods in REST

Method	Purpose	Idempotent
GET	Retrieve resource	Yes
POST	Create resource	No
PUT	Update resource	Yes
DELETE	Remove resource	Yes

REST Controller Example

```
@RestController
public class UserController {
    @GetMapping("/users/{id}")
    public User getUser(@PathVariable String id) {
        return userService.findById(id);
    }
    @PostMapping("/users")
    public void createUser(@RequestBody User user) {
        userService.save(user);
    }
}
```



- Demonstrates path variable extraction.
- Demonstrates request body mapping for POST requests.

Exam corner

- **Q1** Explain the concept of Inversion of Control and Dependency Injection in Spring.
- **Q2** Differentiate between Singleton and Prototype bean scopes with examples.
- **Q3** What is AOP? Explain its key components like Advice, Pointcut, and Join Point.
- **Q4** Describe the significance of @RestController and @RequestMapping in Spring Boot.
- **Q5** Explain the role of Spring Boot Starters and Auto-configuration.
- **Q6** How do CommandLineRunner and ApplicationRunner differ?
- **Q7** Write a short note on the Spring Bean Lifecycle callbacks.

Unit 5 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 1 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

