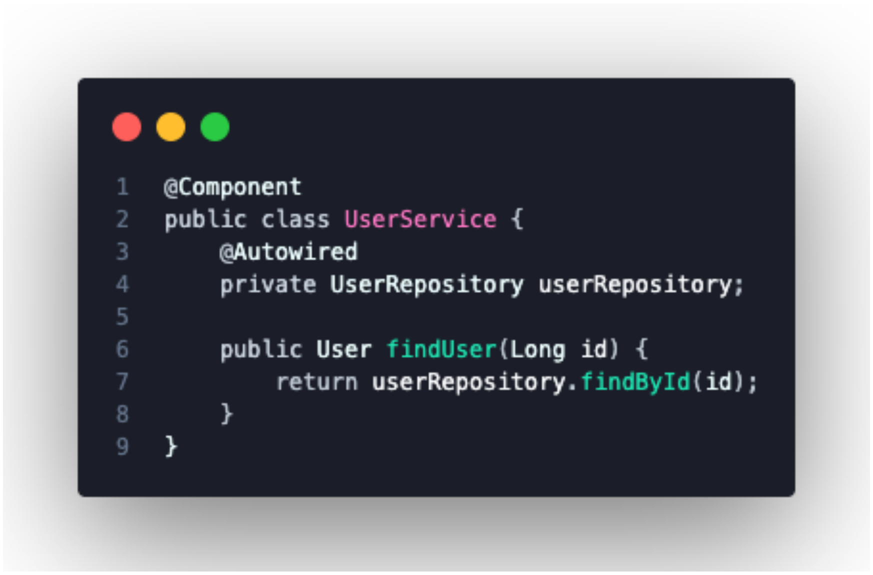
Spring Platform Questions & Answers

What is Spring?

Spring is a comprehensive, open-source application framework for Java that provides infrastructure support for developing enterprise applications. It simplifies Java development by offering features like dependency injection, aspect-oriented programming, and integration with various technologies.

Example:



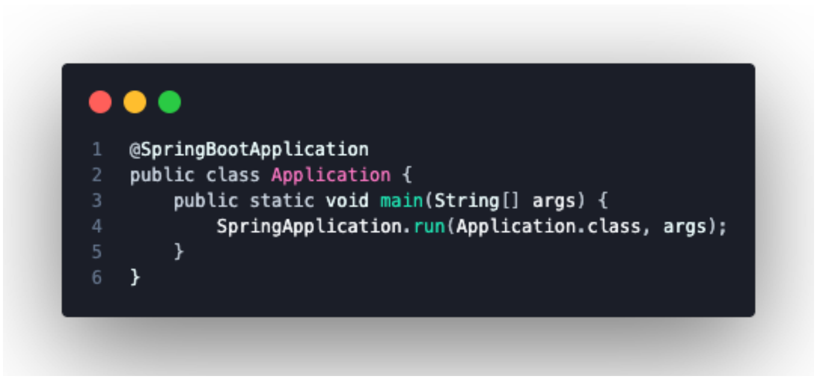
```java
@Component
public class UserService {
    @Autowired
    private UserRepository userRepository;

    public User findUser(Long id) {
        return userRepository.findById(id);
    }
}
```

What is Spring Boot?

Spring Boot is an extension of the Spring framework that simplifies the setup and development of Spring applications. It provides auto-configuration, embedded servers, and starter dependencies to reduce boilerplate code and configuration.

Example:



```java
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

What is the relation between Spring platform and Spring Boot?

Spring Boot is built on top of the Spring platform. It uses Spring framework as its foundation but adds:

- Auto-configuration capabilities
- Embedded server support
- Production-ready features (metrics, health checks)
- Simplified dependency management

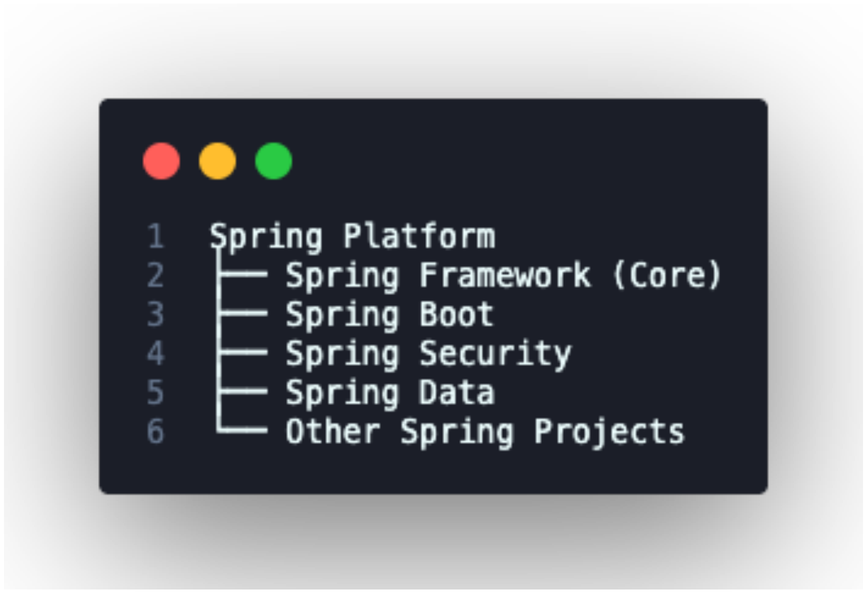
Example: Spring is the engine, Spring Boot is the complete car with all accessories pre-installed.

What is the relation between Spring platform and Spring framework?

The Spring framework is the core of the Spring platform. The Spring platform encompasses:

- Spring Framework (core container, AOP, data access)
- Spring Boot
- Spring Security
- Spring Data
- Spring Cloud
- Other Spring projects

Structure:

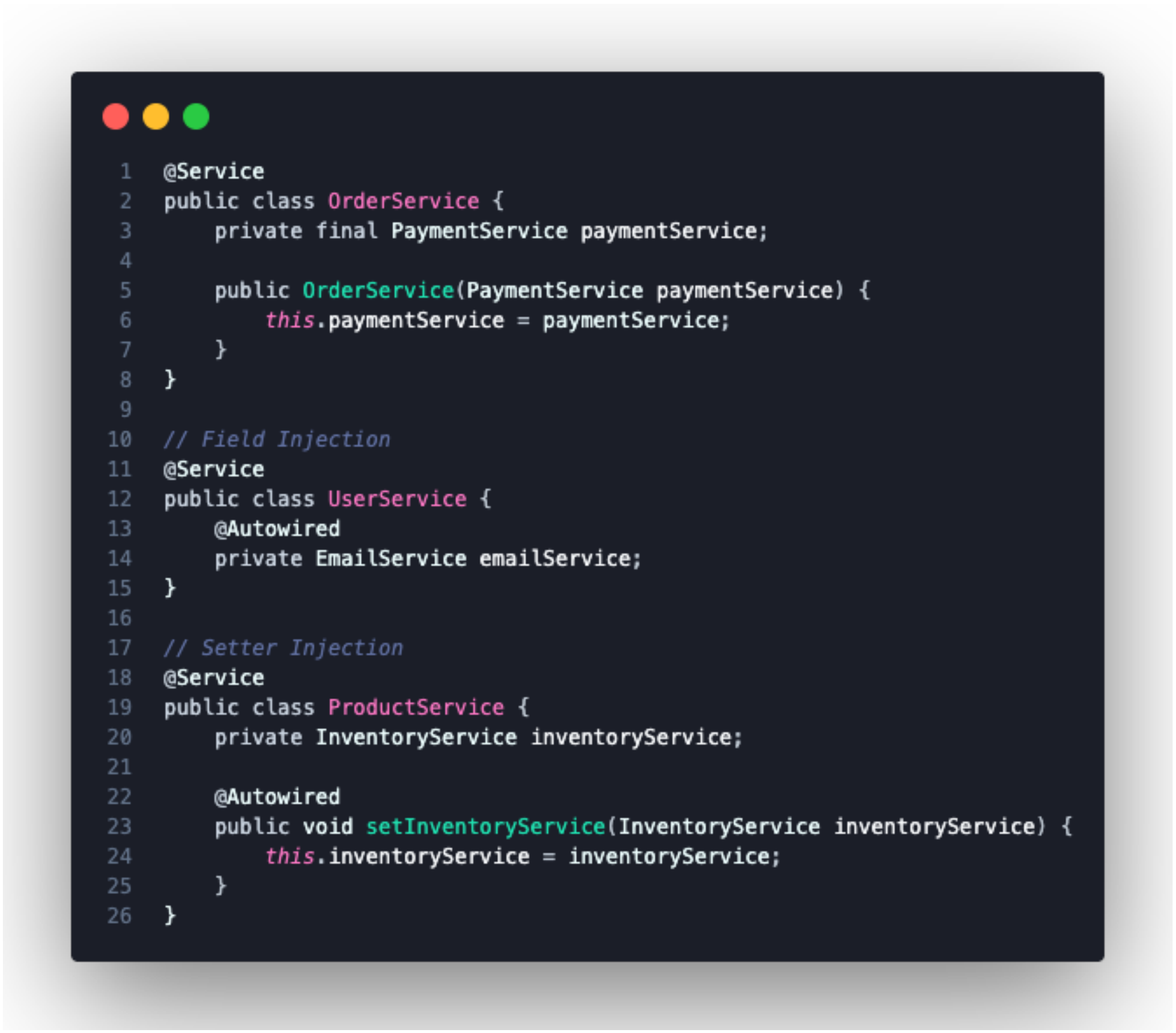


```
Spring Platform
├── Spring Framework (Core)
├── Spring Boot
├── Spring Security
├── Spring Data
└── Other Spring Projects
```

What is Dependency Injection and how is it done in Spring?

Dependency Injection (DI) is a design pattern where objects receive their dependencies from external sources rather than creating them internally.

Example:



```java
@Service
public class OrderService {
    private final PaymentService paymentService;

    public OrderService(PaymentService paymentService) {
        this.paymentService = paymentService;
    }
}

// Field Injection
@Service
public class UserService {
    @Autowired
    private EmailService emailService;
}

// Setter Injection
@Service
public class ProductService {
    private InventoryService inventoryService;

    @Autowired
    public void setInventoryService(InventoryService inventoryService) {
        this.inventoryService = inventoryService;
    }
}
```

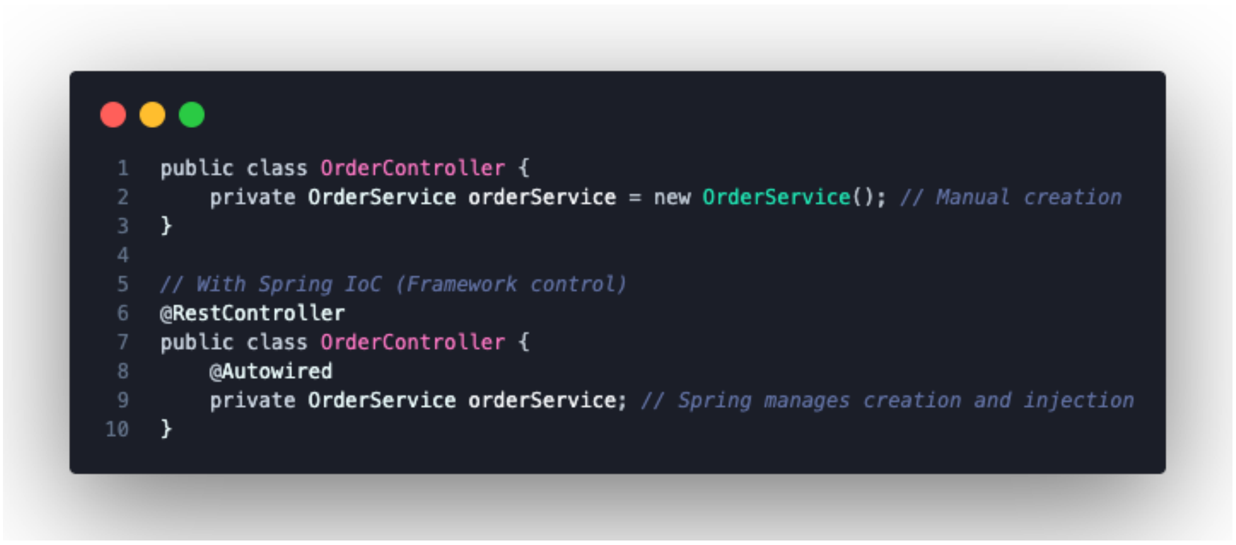
What is Inversion of Control (IoC) and how is it related to Spring?

IoC is a principle where the control of object creation and lifecycle is transferred from the application code to a framework or container.

Relationship to Spring:

- Spring's IoC container manages object creation, configuration, and lifecycle
- The ApplicationContext is Spring's IoC container
- Objects are defined as beans and managed by Spring

Example:



```java
public class OrderController {
    private OrderService orderService = new OrderService(); // Manual creation
}

// With Spring IoC (Framework control)
@RestController
public class OrderController {
    @Autowired
    private OrderService orderService; // Spring manages creation and injection
}
```