Spring Boot ile Rest Service Geliştirme

Mert Alptekin
Lead Software Consultant

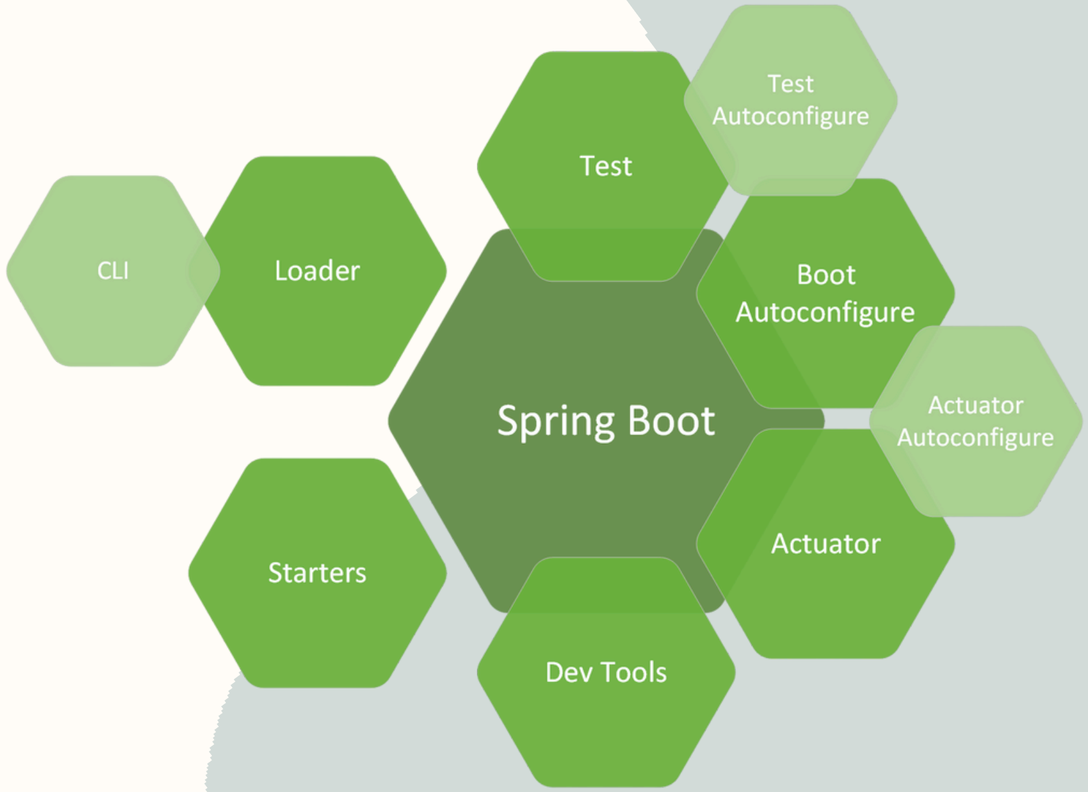
Eğitim Kataloğu

1. Clean Code Principles
2. Spring Boot Ecosystem
3. Spring Web MVC
4. Spring IOC & Annotations
5. Swagger/OpenAPI
6. Spring AOP (Aspect-Oriented Programming)
7. Spring Validation API
8. Spring Data JPA
9. Spring Boot Error Handling & Logging & Ops (Actuator)
10. Spring Event Driven Devopment
11. Spring Security
12. Spring Test Driven Development with Junit

Spring Boot Ekosistemi Nedir?

Java dünyasında hızlı, konfigürasyon yükü az, üretime hazır uygulamalar geliştirmeyi sağlar.

- Spring Framework'ün kolaylaştırılmış versiyonu
- "Convention over Configuration" prensibi
- Embedded Tomcat, Jetty veya Undertow desteği
- Auto Configuration özelliği
- Starter bağımlılıklar (spring-boot-starter-web, spring-boot-starter-data-jpa, vs.)



Spring Boot ile Ne Tür Projeler Yapılabilir?

- RESTful API'ler
- Web uygulamaları (Thymeleaf, React, Angular backend)
- Mikroservis mimarileri (Spring Cloud)
- IoT servisleri
- Batch işlemler (Spring Batch)
- Messaging sistemleri (Kafka, RabbitMQ)

Spring Boot Projelerinde Kullanılan Altyapılar

Yaygın Katmanlar ve Kütüphaneler:

- Veritabanı Katmanı: Spring Data JPA, Hibernate
- Web Katmanı: Spring MVC
- Security Katmanı: Spring Security
- Mesajlaşma: Kafka, RabbitMQ
- Monitoring: Actuator, Micrometer, Prometheus
- Test: JUnit 5, Mockito, Spring Boot Test

Spring Context Kavramı Nedir?

Spring uygulamasının “beyni”. Tüm bean’leri yönetir (IOC Container).
Uygulama başlarken oluşturulur, yaşam döngüsü boyunca nesneleri yönetir.

```
ApplicationContext context = SpringApplication.run(App.class, args);
MyService service = context.getBean(MyService.class);
```

Inversion of Control (IoC) → Bağımlılıkları developer değil Spring yönetir.

Bean Kavramı ve Bean Tipleri

- Spring Context tarafından yönetilen nesnelerdir.

Tanımlama Yolları:

- @Component, @Service, @Repository, @Controller
- @Bean (metod seviyesinde, manuel tanımlama)

Bean Scope'ları:

- singleton (varsayılan)
- prototype
- request, session, application (web context için)

Bean Kavramı ve Bean Tipleri

- Spring Context tarafından yönetilen nesnelerdir.

Tanımlama Yolları:

- @Component, @Service, @Repository, @Controller
- @Bean (metod seviyesinde, manuel tanımlama)

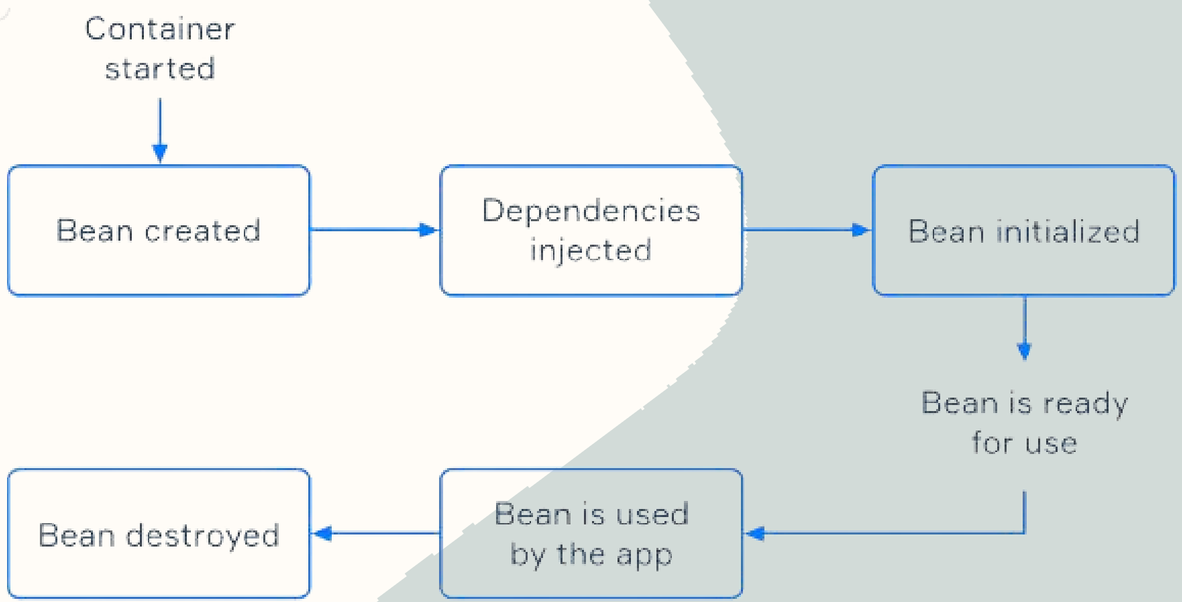
Bean Scope'ları:

- singleton (varsayılan)
- prototype
- request, session, application (web context için)

Bean Lifecycle Yönetimi

Bean Yaşam Döngüsü Adımları:

1. Instantiate → Nesne oluşturma
2. Populate Bean Properties → Bağımlılık enjeksiyonu
3. @PostConstruct → init aşaması
4. Kullanım
5. @PreDestroy → cleanup aşaması



Bean Lifecycle Scopes

- **Request**: (Web uygulamaları için) Her HTTP request başına yeni bir instance oluşturulur.
- **Singleton: (default):** Spring konteynerinde tek örnek oluşturulur, her yerde aynı instance kullanılır.
- **Prototype:** Her injection'da **yeni bir instance** oluşturulur.
- **Session:** (Web uygulamaları için) Her HTTP session başına yeni instance oluşturulur.
- **Application:** ServletContext başına tek instance.

Spring Boot Projelerinin Çalışma Yapısı

Spring Boot projeleri bu 3 anatasyonu içerir.

@EnableAutoConfiguration: Spring, projedeki bağımlılıklara (classpath'e) bakarak otomatik olarak gerekli bean'leri oluşturur.

- spring-boot-starter-web varsa → Tomcat, DispatcherServlet, MVC ayarlarını otomatik kurar.
- spring-boot-starter-data-jpa varsa → EntityManager, TransactionManager vs. otomatik kurar.

@Configuration: Bir sınıfın Spring konfigürasyon sınıfı olduğunu belirtir.
Bu sınıfta tanımlanan @Bean metotları, Spring konteynerine manuel bean olarak eklenir.

- XML konfigürasyonun Java tabanlı versiyonudur.
- @Bean metodları singleton olarak context'e eklenir.
- @Configuration sınıfları da birer @Component gibi davranır.

Spring Boot Projelerinin Çalışma Yapısı

@ComponentScan: Spring'e, hangi paketlerdeki sınıfların **Bean** olarak taranacağını söyler.

- @Component, @Service, @Repository, @Controller anotasyonlarını taşıyan sınıfları bulur.
- Bunları Spring Context'e bean olarak kaydeder.

Spring Boot projeleri bu 3 anatasyonu içerir.

Spring AOP

Amaç:

Tekrarlayan çapraz kesen işlemleri (cross-cutting concerns) merkezi yönetmek.

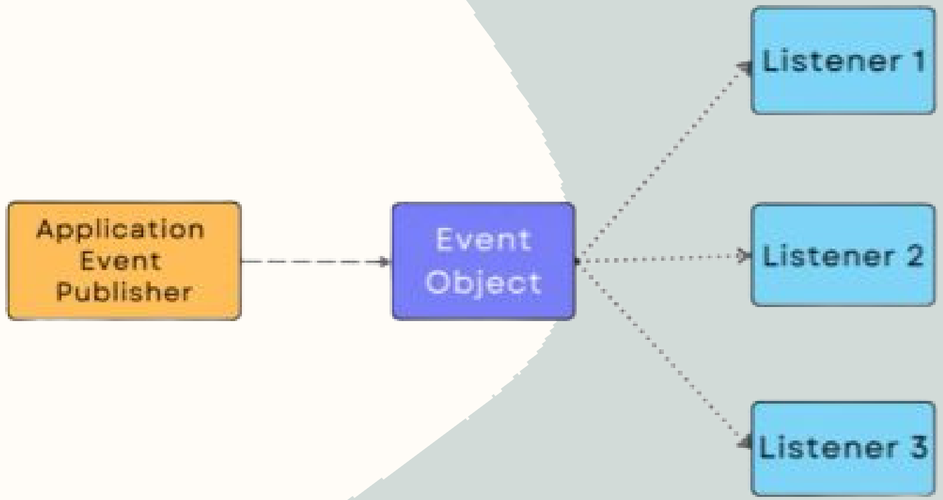
- Aspect: Ortak davranışı tanımlar (örn: loglama, güvenlik)
- Advice: Aspect'in uygulanacağı davranış tipi (Before, After, Around)
- Join Point: Hedef metodun çağrıldığı nokta
- Pointcut: Aspect'in hangi metotlara uygulanacağını belirler

Spring Event Driven Development

Amaç:

Bileşenler arası gevşek bağlı (loosely coupled) iletişim.

- **ApplicationEventPublisher** → Event yayar
- **@EventListener** veya **ApplicationListener** → Event'i dinler



Teşekkürler!

Teknoloji dolu günler geçirmeniz dileği ile hoşçakalın !