PATİKA TÖDEB
JAVA SPRİNG BOOTCAMP

ÖDEV 3

BATUHAN AYYILDIZ

1-

IOC (Inversion of Control)

IOC, programdaki objelerin ya da programın bir kısmının kontrolünün container'a ya da framework'e transfer edilmesidir. Yönetilebilirliği arttırır. Daha kolay unit test yazımı ve modüler programlama yapımını sağlar.

DI (Dependency Incjection)

DI, IOC'nin uygulanmasında kullanılır. Objelerin birbiriyle bağlantısını objelerin kendisi yerine bir assembler idare eder. Bu sayede objeler loosely coupled kriterine daha uygun olur.

2-

Bean Scope, bean'in yaşam döngüsünü ve görünürlüğünü bizim kullanımımıza göre belirler. Güncel Spring'te 6 tip scope vardır. Son dördü Web Aware Scopes'tur. Sadece web-aware uygulamalrı içerisinde kullanılabilirler

- Singleton --> Bean'in kapsama alanını her Spring IOC container için tek instance olarak ayarlar.
- Prototype --> Bean, container'dan her çağrıldığında farklı bir instance ile dönüş yapılmasını sağlar.

Web Aware Scopes

- Request --> HTTP request için olan bir scope'tur
- Session --> HTTP session için olan bir scope'tur
- Application --> ServletContext yaşam döngüsü için bean instance oluşturur.
- Websocket --> Websocket session için olan bir scope'tur

3-

3 tane özelliği aktive etmek için bu anotasyon kullanılır. Bu anotasyonlar aşağıda belirtilmiştir.

- @EnableAutoConfiguration: Spring Bootun otomatik konfigürasyon mekanizmasını aktive eder. Otomatik konfigürasyon, classpath taranırken bean'lerin otomatik olarak oluşturlmasını sağlar.
- @ComponentScan : uygulamanın bulunduğu pakette @Component taramasını aktive eder. Bulunduğu paket ve alt paketleri tarayıp annoted sınıfları belirler ve onları Spring beans olarak yapılandırır.
- @Configuration: Fazladan bean kayıt ettirilmesini ya da ek konfigürasyon sınıflarını import edilmesini sağlar

4-

Spring framework'üne ek olarak gömülü HTTP Serverları(Tomcat,Jetty) barındırır. İlerleyen aşamalarda son derece karmaşık hale gelebilen XML yazımlarını ortadan kaldırır. Daha kısa kodlar yazılmasına ve daha kolay web uygulaması geliştirilmesine olanak sağlar. Otomatik konfigürasyon özelliği vardır. Böylelikle sınıfları istenmelerine, talep edilmelerine göre yapılandırır.

5-

Singleton, bir yazılım design pattern'idir. Bir sınıfın bir objeye olan instantiation'larını kısıtlar.Yani sadece üretilen objeye doğrudan bir ulaşım sağlar. Birden fazla instantiation'ı olmaz. Tek instance'ı olur.

Bir kaynağın değişik partilerle paylaşıldığı durumlarda o kaynağa erişimi kontrol etmek için kullanılabilir. Böylelikle thread safe bir uygulama sağlanır. Aynı kaynağı isteme durumundaki çakışmalar önlenir.

6-

@RestController anotasyonu RESTful web servisleri yapmak için kullanılır. @Controller'ın ve @ResponseBody 'nin kombinasyonundan oluşur. Kullanıldığında @Controller'ın aksine @ResponseBody oluşturulmasına gerek kalmaz. Controller sınıfının her istek işleme methodu dönen objeleri HttpResponse'a otomatik olarak serialize eder. Normalde yani @Controller anotasyonunda, bu iş @ResponseBody kullanılarak yapılır.

7-

Ana farklardan biri, Spring'te olan Dependency Injection'ı geliştirerek Spring Boot Autoconfiguration haline çevirmiştir. Bu özellik sayesinde sınıfları isteklere göre otomatik olarak yapılandırır. Diğer farklar ise Spring Boot'ta Tommycat, Jetty gömülü olarak gelir ve Spring Boot karmaşık XML yazımlarını ortadan kaldırır.

8-

Birden fazla kişinin aynı dosya üzerinde çalışmasını sağlar. O dosya üzerinde başka kişiler tarafından yapılan değişikliklerin sıkıntısız ve hatasız bir şekilde dosyaya entegrasyonunu sağlar. İşe yaramayacak dosyaların karşı tarafa aktarılırken ayıklanmasında kolaylık sağlar. Uygulamaların farklı versiyonlarının daha rahat bir şekilde depolanmasını sağlar. Yedekleme konusunda da avantajı vardır. Eğer çalışılan dosya bilgisayardan ya da storage'tan silinirse rahatlıkla o dosyaya ulaşılmasını sağlar

9-

Solid prensipleri yazılım tasarımındaki 5 prensibi temsil ederken. Prensiplerin ingilizce olan hallerinin baş harflerinin birleşimi SOLID kelimesini oluşturur.

S --> The Single Responsiblity Principle

Bir sınıfın tek bir işi olmalıdır. Bu yüzden değişmek için tek sebebi olmalıdır. Mesela kitap adı altında bir sınıf çeşitli field'lar içerebilir. Bu sınıf sadece veri modeli değişince değişmelidir. Her şeyi yapabilen "God classes" sınıfları oluşturulmamalıdır.

O --> Open-Closed Principle

Sınıflar genişlemeye açık ama modifikasyona kapalı olmalılardır. Var olan sınıfa yeni özellikler eklemek için sınıfı baştan yazmaya gerek olmamalıdır. Mesela kitap sınıfı için yazma özelliği eklenmek istendiğinde kitap sınıfının için değişiklik yapılıp eklenmemelidir. Çünkü bu değişiklik kod içinde hatalara sebep olabilir. Bu özelliğin olduğu bir sınıf (BookPrint) kitap sınıfına extend edilmelidir.

L --> Liskov Substitution

Alt sınıflar üst sınıfların özelliklerini taşımalıdır. Mesela Araba sınıfı üst sınıf ve Toyota sınıfı alt sınıf olsun. Araba sınıfında benzin doldurma methodu olduğu düşünelim. Toyota alt sınıf olarak elektrikli araç ürettiğinde benzin doldurma methodu kullanılmayacak yerine elektrik yükleme methodu gerekecektir. Bu da bu prensibe uyulmadığı anlamına gelir.

I --> Interface Segregation

Büyük ve kapsamlı interface daha küçük interface'lere bölünmelidir. Bu özellik sınıfları implement ettiğimizdee sınıfların sadece onları ilgilendiren methodlara ulaşmasını sağlar. Mesela Printer diye bir interface oluşturup ayrı BookPrinter, FilePrinter, PaperPrinter vb methodlar eklediğimizi düşünelim. Sadece kitap sınıfı için Printer interface'ini kullanmamız diğer gereksiz methodlara da ulaşmamıza neden olur. Ama biz Printer interface'ini BookPrinter vb şekilde farklı interface'lere bölersek sadece ihtiyacımız olan interface'i kullanabiliriz.

D --> Dependency Inversion

Üretilen sınıflar, temel sınıflar veya fonksiyonlar yerine interface'lere ya da abstract sınıflara bağlı olmalıdırlar. Eğer yukarıdaki örnekte belirtilen Book sınıfında BookPrinter sınıfına bağlı bir obje new yaparak oluşturursak birbirine yüksek derecede bağlı kılmış oluruz. Bu yüzde BookPrinter sınıfında yaptığımız bir değişiklik Book sınıfında hata oluşmasına sebep olabilir. Bu yüzden interface kullanıp dependency azaltılmalıdır.

10-

RAD, yazılım geliştirme modellerinden biridir. Bir uygulamanın minimum seviyede bir planlamayla, hızlı bir şekilde prototipinin üretilmesi ve deneme yanılmayla uygulamanın geliştirilmesidir. Düşük seviyeli teknik riskler olduğunda, yüksek performans istenmeyip kullanıcının istekleri detaylı ve iyi bir şekilde biliniyorsa bu modelin kullanılması daha uygundur.

11-

Spring boot starters bağımlılık tanımlayıcılarıdırlar. Bu starterlar sayesinde Spring ile ilgili gerekli teknolojileri teker teker araştırma yapıp dependency'leri yazma yükünden kurtulunur. Avantajları:

- Konfigürasyon zamanını azaltarak developerlar için zamandan tasarruf sağlar
- Eklenmesi gereken dependency'ler azaldığı için POM'un daha rahat idare edilmesini sağlar.
- Dependency isimlerini ve versiyonlarını hatırlama gereksinimini veya onları bulmak için araştırma yapma durumunu ortadan kaldırır.

12-

Spring anotasyonları bir program hakkında temel bilgileri veya verileri sağlayan metadata formlarıdır. Bir öğeyi tanımlar ve ne yapması gerektiğini açıklarlar. Bu ifadeler run-time anında framework tarafından ya da geliştirme anında IDE veya complier tarafından yorumlanırlar. Spring Boot'taki temel anotasyonlara @Bean, @Service,@Repository,@Configuration,@Controller,@RequestMapping,@Autowired, @Component,@Qualifer,@SpringBootApplication,@ComponentScan ,@Required ... vb anotasyonlar örnek olarak gösterilebilir.

13-

Spring boot'ta dependency management otomatik olarak gerçekleşir. Spring boot'un her versiyonu desteklediği dependency'lerin listelerini barındırır ve bunları otomatik bir şekilde idare eder. Bu sayede kendi konfigürasyonlarımızı oluştururken dependency'lerin versiyonlarını belirtmemize gerek kalmaz. Spring Boot sürümünü tek bir yerde belirterek dependency bilgileri merkezileştirir. Bu durum, Spring Boot'ta bir sürümden diğer sürüme geçişi kolaylaştırır. Ayrıca otomatik dependency yönetimi, farklı Spring boot kütüphanelerindeki versiyon uyuşmaması hatalarının da önüne geçer.

14-

Spring Boot Actuator, Spring Boot Framework'ünün alt projesidir. HTTP endpointler kullanarak çalışan bir uygulamadaki operasyonel bilgileri görülmesini sağlar. Hazır uygulamaların health ve monitoring metriklerini, ölçülerini elde etmek için kullanılır.