I. Mertkan YILMAZ

1-- Spring dışında dependency injection için kullanabileceğimiz framework'ler / kütüphaneler nelerdir ?

Dependency injection için aslında direkt olarak bir framework kullanmak zorunda değiliz. Framework kullanmadan da yapılanlar yapılabilir ancak framework kullanmak işimizi kolaylaştırır. Mesela bir dependecyi direkt olarak sadece bir anotasyon ve bir constructor ile inject ederiz.

Bunun yanı sıra kullanılan farklı frameworkler mevcuttur. Spring.NET, Unity, Ninject, Weld.

2-- @SpringBootApplication anotasyonu hangi anotasyonları kapsamaktadır ? Görevlerini kısaca açıklayınız.

@SpringBootApplication anotasyonu @Configuration**,** @EnableAutoConfiguration ve @ComponentScan anotasyonlarını kapsar. Anatosyanlar sayesinde dependency injection konfigürasyon işleri sağlanır, beanlar bu anotasyonlar ile tanımlanır. Bunlar yerine XML file ile bu iş yapılabilir ama bu daha zordur ve uğraştırıcıdır.

@Configuration anotasyonu, kullanıldığı sınıfta @Bean methodları mevcut olduğunu belirtiriz ve Spring Containerın run-timeda bu beanlerden servis requesti yaratmasını sağlar. @ComponentScan ile spring'in controller, service ve diğer tanımlanan komponentler için bakmasını sağlarız. @EnableAutoConfiguration ile bizim tanımladığımız beanlere göre application context'I oluşturur. Bu anotasyonu application class'ına koymalıyız ki paketteki tüm classlara geçerli olsun.

3-- Primary, @Qualifier anotasyonlarının kullanım amaçlarını açıklayınız.

Baze durumlarda aynı türden sınıflar için farklı beanler oluştururuz bu gibi durumlarda @Primary anotasyonu ile beanlar arasında öncelik vermis oluruz. NoUniqueBeanDefinitionException Error'unu alırsak muhtemelen bu tarz bir öncelik verilmediği içindir, bu anotasyon ile bu sıkıntı giderilebilir. @Qualifier ise hangi bean'in inject edileceğine springin karar veremediği durumlarda karışıklığı gidermek için kullanılan bir diğer anotasyondur. Bu anotasyon ile aynı türler arasında hangi beani inject etmek istediğimizi direct olarak veririz. Spesifik olarak isim belirtiriz. @Qualifier ve @Primary arasındaki fark @Qualifier'in direct olarak inject edilecek olanı belirtirken, @Primary'nin öncelik belirlemesidir.

4-- Convention over configuration kavramını seçtiğiniz bir örnek üzerinden açıklayınız.

Konfigürasyon demek bir yazılım projesinde bazı gerekli temellerin bizim tarafımızdan genellikle XML, YAML, JSON dosyaları ile konfigüre edilerek oluşturulmasıdır. Bu konfigürasyon yaklaşımı ile hareket edilen projelerde yazılımcı herhangi birşeyi konfigürasyon dosyaları vs ile konfigüre eder. "Convention" yaklaşımı ile hazırlanan projelerde bir çok şey düşünülmeden belli bir pattern çerçevesinde gider. Herşeyin bir default'u vardır. Bu sayede yazılımcılar kodun temel işleyişine, uyhulama mantığına odaklanıp bunun geliştirmeye çalışırlar. Bunu birkaç örnek üzerinden açıklarsak;

Maven ile hazırlanan projelerde belirli bir dosya sistemi vardır ve her dosyanın nerede konumlanacağı, dosya ağacı bellidir. Framework uygulamayı buna göre çalıştıracağı için yazılımcı bu

düzeni kendi kurmaz. Mesela Pytest framework'ünde testler test klasörünün altına yazılır ve test caseler buradan okunur. Bunlar uygulamanın defaultundadır.

5-- Aspect Oriented Programlama nedir ? Avantajları ve dezavantajları nelerdir ?

Aspect Oriented Programming (AOP) aslında iyi kod yazma amaçlı bir uygulamadır. Uygulamamızda çok fazla kodu, aynı kodları aynı görevler için tekrar tekrar kullanacağımıza uygulamamızdan bunları ayırarak gerektiğinde kullanmamızı sağlayacak daha modüler bir yapı kurmamızı sağlar. OOP ile bir benzeşimini kuracak olursak; OOPde uygulamayı classlara bölerken burada "aspectlere" böleriz diyebiliriz.

Aspect yaklaşımı ile "Cross-cutting concerne"leri kodumuzun temel mantığını, core diyebileceğimiz mantığını etkilemeyen kısımları ayırırız. ("Cross-cutting concern" loglama, güvenlik, data aktarımı gibi konularda ihtiyaç duyduğumuz kısımlardır. Bunlar uygulamanın işleyişi etkiler ama asıl, temel mantığını etkiler diyemeyiz. Bunlar neredeyese tüm uygulamalarda ihtiyaç duyulan işlemlerdir. Mesela güvenlik doğrulaması gereken işlemler her uygulamada çoğu methodta vardır ve bunları uzun uzun tekrar yazmak yerine bir aspect olarak yazarız.)

Avantajları şu şekildedir ; İçi içe yazılmış ve sürekli tekrar eden kodlardan kurtulabiliyoruz, daha temiz ve anlaşılır kodlar yazabiliyoruz, yazmış olduğumuz kodları daha abstract hale getirerek modülerliğini arttırıyoruz, bakım ve geliştirme maliyetlerini azaltıyoruz, uygulamamızı daha yönetilebilir ve daha esnek hale getirebiliyoruz.

6-- SOLID prensiplerini kısaca açıklayınız. Sizce her koşulda bu prensipler çerçevesinde mi kod yazılmalıdır ? Neden ?

SOLID prensipleri kod yazarken, özellikle enterprise kod yazarken bir proje geliştirilirken izlenmesi fayda sağlayan kurallardır. Bu kurallar, uyulmazsa programımızın çalışmasını engelleyecek vesaire kurallar değildir. Kod bu kurallara göre yazıldığında daha maintain edilebilir, bakımı kolay, ileride olabilecek değişikliklere daha uyumlu, başkaları tarafından daha anlaşılabilir kurallar bütünüdür. Bu kurallar yazılım camiası tarafından temiz kod yazımı çerçevesinde kabul edilmiş, kabul görmüş kurallardır.

S : Single Responsibiltiy maddesi, uygulamada bir sınıfın sadece bir işten sorumlu olması gerektiğini söyler.

O : Open/Closed Principle maddesi, bir sınıfın extension yani eklentiye açık ama direkt olarak modifikasyona kapalı olması gerektiğini söyler.

L : Liskov Substitution Principle maddesi, bir sınıfın alt sınıfının bir objesinin, o üst sınıfın herhangi bir objesi ile yer değiştirdiğinde herhangi bir mantık hatasına veya büyük değişime yol açmadan yer değişebileceğini söyler.

I : Interface Segregation Principle, bir sınıfın implemente ettiği bir methodda kullanmayacağı bir method olmaması gerektiğini veya clientlerin asla kullanamayacakları sınıfları bağlı olmaması gerektiğini söyler.

D : Dependency Inversion Principle, bir high-level modülün low-level modüle bağımlı olmaması gerektiğini söyler. İkisinin de abstract bir sınıfa bağlı olması gerektiğini ifade eder.

7--Swagger nedir, ne amaçla kullanılmaktadır ?

Swagger aslında herhanfi bir RESTful api'yi bize tanıtan, nasıl kullanıldığını anlatan, arayüz sağlayarak kullanmamızı sağlayan bir API'dir. Mesela bir örnek uygulama geliştirdik ve endpointleri vs mevcut. CRUD operasyonları yapabildiğimiz bir uygulama olduğunu düşünelim. Bunların endpointlerine erişerek, bunlara nasıl istek atacağımızı dökümante eder. Bunları kullanabileceğimiz, hatta database'e kayıt yapıp işlemleri yapacağımız arayüz sunar. Swagger'ın kullanılması için Spring projemize gerekli dependencyleri eklemeyeli ve gerekli konfigürasyonları yapmalıyız.

8-- Richardson Maturity Model'i seviyeleriyle birlikte açıklayınız.

Richardson Maturity Model (RMM) bir web servisin REST servis olabilmesi için belli bir oldunluk seviyesine göre sınıflandırılmasıdır. RMM'e göre 4 kademe vardır.

Kademe 0da servisler sadece tek bir URL işlem yapar. Yani karşı tarafta yapılacak her işlem için URI aynıdır. Tek bir end-point vardır. Hullanılan herhangi bir HTTP method yoktur.

Kademe 1de birden fazla end-point vardır, birden fazla URI mevcuttur ve artık HTTP method olan POST kullanılır. Yani aslında seviye 0'dan farklı olarak farklı resourcelara erişim var.

Kademe 2, POST methoduna ek olarak PUT, DELETE, GET methodlarının da olaya dahil olduğu seviyedir. Ayrıca bu seviyede HTTP Error cevaplarıda döndürülür.

Kademe 3'te artık bir request yapıldığında karşı taraftan yapabileceğimiz farklı işlemler döner. Kademe 3'te HATEOAS (Hypertext As The Engine Of Application State) diye bir yapı devreye girer. HATEOAS sayesinde cliente farklı cevaplar dinamik olarak verilebilir. Aslında bir nevi server tarafı client'in nasıl ilerleyeceğini kontrol eder.

9-- URL, URI, URN kavramlarını bir örnek üzerinden açıklayınız.

URI (Uniform Resource Identifier) web teknolojileri tarafından kullanılan, kendine has,tek bir soyut veya somut bir karakter setidir. Bazı URI'ler aynı zamanda bir kaynağın yerini de belirlemek için kullanılırlar yani aynı zamanda adresini de bu karakter seti ile belirtirler. Bu özelliğe sahip olanlara URL(Uniform Resource Locator) denir. Diğer URI'ler ise sadece isim olarak unique yapılardır ve herhangi bir kaynak adresini belirtmezler sadece isim belirtirler. Bunlar URN (Uniform Resource Names) olarak adlandırılır.

Mesela ISBN (International Standart Book System) bir URN belirtir. "urn:isbn:0-486-27557-4" karakter seti aslında Shakespeare's play Romeo and Juliet'nin bir özel yayının URNsidir. İsim olarak tektir ama bunun dışında yerine, lokasyonuna dair bir bilgi değildir. Bir başka örnek ise http://example.org/wiki/Main_Page adresidir. Bu ise bir URLdir ve bir kaynağın adresini belirtir. example.org adresinden sunulan wiki/Main_Page kaynağının adresidir.

10— Idempotency nedir ? Hangi HTTP metotları idempotent' tir ?

Idempotency bilgisayar bilimlerinde bir sistemde bir çağrının, işlemin yapılması sonucu üzerinde işlem yapılan kaynağın, bu işlem aynı şekilde defalarca da yapılsada değişmeden kalması durumudur. Idempotency'i sağlayan HTTP methodları GET, PUT, DELETE idempotent HTTP methodlarıdır. Neden? GET methodu ile zaten kaynakta herhangi bir değişiklik yapmadan sadece sonuç görme işlemleri yapılır. PUT ve DELETE ise ilk işlemde, evet kaynakta bir değişikliğe sebep olabilirler fakat bu

methodları 2., 3. Ve daha sonraki kullanışlarımızda kaynakta herhangi bir değişiklik meydana gelmez. POST methodunda ise kaynakta direkt olarak bir değişiklik olur ve POST ettiğimiz eklentiler, kaynağı değiştirir.

11-- RFC (Request For Comment) neyi ifade etmektedir ? HTTP hangi RFC dokümanında açıklanmıştır ? Bu dokümanda HTTP hakkında ne tür bilgiler yer almaktadır ?

RFC numaralndırılmış, internet hakkında spesifikasyonlar ve notlar içeren dökümanlardır. Bu dökümanlar IETF (Internet Engineering Task Force), IAB (Internet Architecture Board), IRTF (Internet Research Task Force), ve bağımzsız yazarlar tarfından oluşturulmuşlardır. Internet networkleri hakkında bir çok detaylı yazılardır.

HTTP, RFC2616da ve RFC2068de detaylandırılmıştır. HTTP notasyonları, protokol parametreleri (URL, URI, URN), HTTP mesajları, bileşenleri (body, header, request) methodları gibi bir çok detayı içerir.