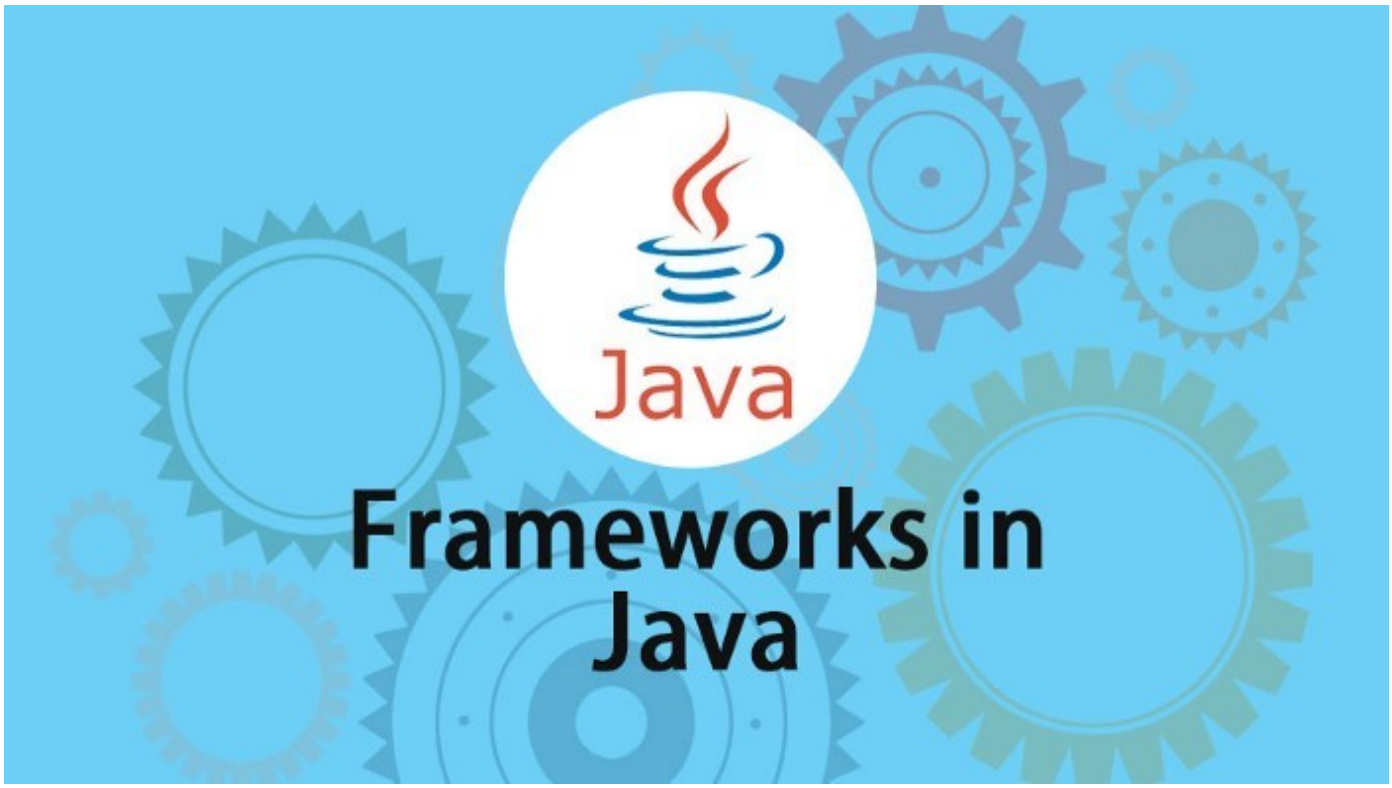
Java Frameworkleri ve Çözmüş olduğu problemler



Framework nedir? İsterseniz ilk önce bu ne anlama geliyor onu öğrenelim.

Framework: Uygulamalarımızı-projelerimizi daha düzenli, daha güvenli ve daha hızlı geliştirmek için oluşturulmuş ortamlardır. Uygulamamızın işlevine göre bir yazılım dili seçip o dile ait ve projeye yönelik bir framework kullanabiliriz. Piyasada yüzlerce framework var ve hangisi daha iyi tartışmaları yer almaktadır. En iyi framework diye bir şey yoktur aslında yarın daha iyi bir framework ile çalımayacağınızı kimse garanti edemez. Sizin isteklerinize cevap veren her framework benim gözümde iyidir.Tabiki de artıları ve eksileri vardır,olacaktır da...

Bir framework kullanmanın avantajları aşağıdaki şekildedir:

- Daha iyi programlama uygulamaları oluşturmaya yardımcı olur.
- Kod daha güvenlidir.
- Yinelenen ve gereksiz kod önlenebilir.
- Daha az hatayla tutarlı kod geliştirmeye yardımcı olur.
- Gelişmiş teknolojiler üzerinde çalışmayı kolaylaştırır.
- Yazılım framework'lerini oluşturabilir veya açık kaynaklı olanlara katkıda bulunabilirsiniz. Dolayısıyla, işlevsellikte sürekli bir gelişme sağlar.
- Çeşitli kod segmentleri ve işlevleri önceden oluşturulmuş ve önceden test edilmiştir. Bu, uygulamaları daha güvenilir hale getirir.
- Kodu test etmek ve hata ayıklamak çok daha kolaydır. Koda sahip olmayan geliştiriciler tarafından bile yapılabilir.
- Bir uygulama geliştirmek için gereken süre önemli ölçüde azalır.

1-Spring

Spring framework muhtemelen bilinen en popüler frameworklerden birisidir. Büyük bir ekosistemi ve topluluğu bulunmaktadır. Kurumsal düzeyde java uygulamaları,web uygulamaları ve microservisler oluşturmamızı sağlar. Spring aslında bir dependency injection toolu(bağımlılık enjeksiyon aracı) olarak başladı fakat zamanla tam ölçekli bir uygulama frameworku haline geldi. Gel gelelim dependency injection nedir? DI aslında projede geçen bağımlılıkların(nesnelerin, sınıf içerisinde kullanılacak olan yapıların) sınıflara geçilmesi işlemidir.

Spring framework; veritabanı bağlantıları, istisnaları yakalamak gibi genel görevler de dahil yapılandırma ve programlama modeli sunmaktadır.

Spring, IoC (inversion of control) prensibine göre hareket etmektedir.IoC; Sınıfların ya da servislerin; bağımlılıklarının, yaşam döngüsünün (oluşturulmasının ve yok edilmesinin) kontrollerinin framework ya da bir container içerisine verilmesi işlemidir. Bu prensip esasta yaşam döngüsü efektif bir biçimde yönetmek işlemini baz alır ve bunu başarılı bir şekilde gerçekleştirir.

- **Lightweight:** Spring hafiftir.Temel sürümü 2MB'tir fakat lightweight kavramı uygulamada belirli bir değişiklik yapıldığında tüm uygulama bu değişiklikten minumum düzeyde etkilenir düşüncesini de içermektedir.
- **Dependency Injection/Inversion of Control (IoC):** Bağımlılıkların kontrolünü louse coupling(gevşek bağımlı) olarak gerçekleştirmektedir.
- **Spring Container:** Uygulama nesnelerinin yaşam döngüsünü içerir ve onları yönetir.
- **Transaction Management:** JDBC,dosya yükleme,hata yakalama olaylarını ve bean konfigurasyonlarını yönetir.
- **Spring MVC:** SWeb uygulamalarının oluşturulmasında ve XML,JSON gibi yanıtların döndürülmesinde kullanılır.
- **Exception Handling:** JDBC,Hibernate vs. tarafından gelecek özel hataların çevrilmesinde onların anlaşılır kılabilmek için bir API sağlar.
- **Easy Test:** Kolay test edilebilirlik sağlar.

Coupled ve Decoupled adına bir örnek gerçekleştirelim.

```java
package week01.model.test;

public class Car {

    private Tire tire;

    public Car() {

        tire=new Tire();

    }

}
```

Burada birbirine katı bağımlı iki sınıf görüyoruz. Yapıcı metot içerisinde new anahtar sözcüğüyle bir nesne üretilmekte ve Tire sınıfında bir değişiklik yaptığımızda Car sınıfının etkilenmemesi mümkün değildir. Peki bunu nasıl decoupled hale getireceğiz?

Buradaysa interfaceler üzerinde kodumuz gevşek bağımlı hale getirmiş olduk.Wheel yani tekerlek birçok çeşidi olabilir ve buraya göbekten dogrudan bağımlı hale gelmemiş olduk.

2-Hibernate

Java programlama dili ile RDBMS(ilişkisel veritabanı yönetim sistemeleri) arasında iletişim kurmamızı sağlayan ORM (Object-Relation Mapping) framework'üdür.

-Taşınabilir, üretken ve sürdürülebilirdir.

-Açık kaynak bir frameworktur.

– Veritabanınız ile modelleriniz arasındaki bağlantıyı sağlamak için geliştirilmiş en iyi sistemdir.
– Veri ekleme, silme, güncelleme, okuma gibi işlemleri kolaylaştırmaktadır.
– Veri aktarma sürecini yönetmenizi sağlayarak, alabileceğiniz veya çıkabilecek hatalara karşı yönetimi de sağlamaktadır.
– SQL yazmaktan kurtarmaktadır.

JDBC ile veritabanına kayıt eklerken kullandığımız yapı;

```
stmt.executeUpdate( "INSERT INTO ogrenci VALUES ('Beytullah', 'Bozkurt')");
```

Hibernate ile kayıt eklerken kullanacağımız yapı ise;

```
session.save(ogrenci);
```

3-JSF Java Server Faces

Java tabanlı uygulamalar için kullanıcı arayüzleri oluşturmak için kullanılan ve Oracle tarafından geliştirilen bir UI frameworktur.MVC modeli baz alarak hareket eder.

```
1.// index.xhtml
2. <?xml version=’1.0' encoding=’UTF-8' ?>
3. <!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN””http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4. <html xmlns=”http://www.w3.org/1999/xhtml"
5. xmlns:h=”http://xmlns.jcp.org/jsf/html">
6. <h:head>
7. <title>User Form</title>
8. </h:head>
9. <h:body>
```

10. **<h:form>**
11. **<h:outputLabel** for=”username”**>**User Name**</h:outputLabel>**
12. **<h:inputText** id=”username” value=”#{user.name}” required=”true” requiredMessage=”User Name is required” **/>
**
13. **<h:commandButton** id=”submit-button” value=”Submit” action=”response.xhtml”**/>**
14. **</h:form>**
15. **</h:body>**
16. **</html>**

Görüldüğü üzere belli tagler içerisine yazılararak kullanıcı arayüzleri geliştirmemizi sağlamaktadır.

4-SPARK

Spark Framework, Java ve Kotlin programlama dilleri için geliştirilmiştir. Kotlin, Java Sanal Makinesi üzerinde çalışır ve Java ile %100 birlikte çalışabilir. Spark, Java tabanlı web uygulamaları, mikroservice’ler ve REST API’leri geliştirmenize yardımcı olur.

Minimal seviyede kod yazımını amaçlamıştır.

```
post("/users", (request, response) -> {     response.type("application/json");



User user = new Gson().fromJson(request.body(), User.class);
userService.addUser(user);



return new Gson() .toJson(new
StandardResponse(StatusResponse.SUCCESS)); });
```

Ekleme işlemi yapan bir kod örnegi görmekteyiz.Okunurluk artıran bir yapısı bulunmaktadır.