



Java Spring Boot Bootcamp

HAZIRLAYAN

MEHMET ALTAN

Framework Nedir?

- ▶ Bir framework, genel bir yapı, rehberlik veya temel sağlayan büyük çaplı bir yazılım yapısıdır. Framework'ler, bir uygulama veya proje geliştirmek için gerekli olan temel bileşenleri, işlevleri ve kalıpları içerirler. Framework'ler genellikle belirli bir amaca yönelik genel bir altyapı sağlarlar ve geliştiricilere bir temel çerçeve sunarak daha hızlı ve düzenli bir şekilde çalışmalarına yardımcı olurlar. Framework'ler, genellikle belirli bir programlama dili veya platforma özgüdürler.
- ▶ Örnek olarak, web geliştirme alanında popüler olan bir framework olan "Ruby on Rails", web uygulamalarını hızlı bir şekilde geliştirmek için gerekli temel bileşenleri ve kalıpları içeren bir çerçeve sağlar.

Kütüphane (Library)

- Bir kütüphane, genellikle belirli bir programlama dili veya platformda kullanılmak üzere önceden yazılmış işlevlerin (fonksiyonlar, sınıflar, modüller vb.) koleksiyonudur. Kütüphaneler, geliştiricilere belirli görevleri yerine getirmek için kullanabilecekleri araçlar sunar. Kütüphaneler, projelerdeki tekrarlayan görevleri kolaylaştırmak ve kod tekrarını azaltmak amacıyla kullanılır.

Farklar


- ▶ Framework'ler genellikle daha büyük ve kapsamlı bir yapı sunar, işlevler, kalıplar ve temel bileşenler içerirken, kütüphaneler daha spesifik işlevler ve araçlar sunar.
- ▶ Framework'ler genellikle belirli bir amacı veya kullanım senaryosunu hedeflerken, kütüphaneler daha geniş bir yelpazede kullanılabilir.
- ▶ Framework'ler genellikle uygulamanın temel yapısını ve akışını belirlerken, kütüphaneler daha özgün görevleri yerine getirmek için kullanılır.
- ▶ Genel olarak, bir framework bir yapı sunar ve uygulamanın genel tasarımını belirlemeye yardımcı olurken, bir kütüphane ise belirli görevleri daha kolay gerçekleştirmek için kullanılan araçlar sağlar.


Örnekler;

- ▶ Spring Boot, Java tabanlı bir framework'dür ve özellikle mikro hizmetler ve web uygulamaları gibi modern uygulamaların hızlı bir şekilde geliştirilmesini sağlar. Spring Boot, otomatik yapılandırma, hızlı prototipleme ve hazır bileşenler gibi özellikler sunarak geliştiricilere genel bir çerçeve sağlar.
- ▶ Apache Commons Lang, Java programcılarının sıkça karşılaştığı yaygın görevleri kolaylaştırmak için kullanılan bir kütüphanedir. Bu kütüphane, metin manipülasyonundan dizge işlemlerine, veri dönüşümlerinden nesne işlemlerine kadar birçok genel işlevi içerir. Örneğin, metin manipülasyonu işlemleri, dize boşluğu kontrolü, dize birleştirme ve bölme işlemleri gibi işlevleri bu kütüphane sağlar.

Spring Framework Nedir Neden Kullanılır?

- ▶ Spring Framework, Java tabanlı bir açık kaynak yazılım çerçevesidir.
- ▶ Spring Framework, özellikle büyük ve karmaşık uygulamaların geliştirilmesi sırasında yaşanan zorlukları hafifletmeyi amaçlar.
- ▶ Spring'in temel unsurlarından biri, uygulama düzeyinde altyapısal destektir: Spring, kurumsal uygulamaların "tesisatına" odaklanır, böylece ekipler, belirli dağıtım ortamlarıyla gereksiz bağlar olmadan uygulama düzeyinde iş mantığına odaklanabilir.

- 
- ▶ Hızlı ve Kolay Geliştirme: Spring Boot, birçok yaygın kullanım senaryosu için önceden yapılandırılmış ayarlar ve varsayılan değerler içerir. Bu, geliştiricilerin proje kurulumu ve yapılandırmasıyla uğraşmak yerine daha hızlı bir şekilde kod yazmalarına olanak tanır.
 - ▶ Mikro Hizmet Mimarisine Uygunluk: Spring Boot, mikro hizmet mimarileriyle uyumlu bir şekilde tasarlanmıştır. Bu, servisleri küçük ve bağımsız bileşenlere bölmeyi ve bu bileşenleri kolayca dağıtmayı kolaylaştırır.

- 
- ▶ Otomatik Yapılandırma: Spring Boot, uygulama yapılandırmasını otomatik olarak yapabilen geniş bir özellik yelpazesi sunar. Bu sayede geliştiriciler, XML veya yapılandırma dosyalarıyla uğraşmak yerine daha basit ve anlaşılır bir şekilde yapılandırma yapabilirler.
 - ▶ Entegrasyon Kolaylığı: Spring Boot, çeşitli veritabanları, mesaj sıralama sistemleri, güvenlik çerçeveleri ve diğer dışsal bileşenlerle kolay entegrasyon sağlar. Bu, geliştiricilerin farklı bileşenleri uygulamalarına entegre etmesini basit hale getirir.


- ▶ Ölçeklenebilirlik ve Performans: Spring Boot uygulamaları genellikle yüksek performans ve ölçeklenebilirlik sağlamak için tasarlanmıştır. Spring Framework'ün temel avantajları, Spring Boot'un bu özelliklerini de destekler.
- ▶ Topluluk Desteği: Spring Boot, geniş ve aktif bir topluluğa sahiptir. Bu topluluk, soruları yanıtlamak, rehberlik etmek ve kaynak sağlamak gibi konularda yardımcı olabilir.
- ▶ Üretkenlik Artışı: Spring Boot, geliştiricilerin tekrarlayan görevlerle uğraşmasını azaltarak üretkenliği artırır. Özellikle hızlı bir şekilde prototip oluşturmak veya yeni özellikler eklemek istediğinizde büyük bir avantaj sağlar.

Spring Core Teknolojilerini Tanıma

- ▶ "Dependency Injection" (Bağımlılık Enjeksiyonu), yazılım geliştirmede kullanılan bir tasarım deseni ve Spring Framework gibi çerçevelerde temel bir ilkedir.
- ▶ Bağımlılık Enjeksiyonu, bir nesnenin (veya bileşenin) başka bir nesneye olan bağımlılığının, dışarıdan enjekte edilerek sağlanmasını ifade eder. Bu bağımlılık, genellikle bir sınıfın başka bir sınıfa olan bağımlılığını içeren bir nesne referansını içerebilir.

Bağımlılık Enjeksiyonu'nun faydaları;

- ▶ Esneklik: Bağımlılıklar, bileşenin kendisi tarafından yönetilmediğinden, bileşenin başka bileşenlerle değiştirilmesi daha kolaydır. Böylece, uygulama davranışını değiştirmeden bileşenleri güncellemek veya değiştirmek mümkün olur.
- ▶ Test Edilebilirlik: Bağımlılıklar dışarıdan enjekte edildiğinde, bileşenler bağımlılık yerine taklit edilmiş (mock) nesnelerle test edilebilir. Bu, bileşenlerin yalnızca kendi işlevselliğini test etmenizi sağlar ve dışarıdaki bağımlılıkların testi için daha fazla karmaşıklık oluşturmaz.
- ▶ Tekrar Kullanılabilirlik: Bir bileşenin birden fazla yerde kullanılması gerektiğinde, aynı bağımlılığı birkaç kez yeniden oluşturmak yerine tek bir yerden enjekte edilebilir.
- ▶ Daha Az Bağımlılık: Bağımlılık Enjeksiyonu, sınıfların birbirleriyle daha az sıkı bağlı olmasını sağlar. Bu, bileşenlerin daha bağımsız ve değiştirilebilir hale gelmesine yardımcı olur.

- 
- ▶ Spring Framework'te Dependency Injection, bean'lerin (nesnelerin) yönetildiği IOC (Inversion of Control) konteyneri aracılığıyla gerçekleştirilir. Spring, farklı türlerde bağımlılık enjeksiyonu sağlayarak (örneğin, constructor enjeksiyonu, setter enjeksiyonu veya field enjeksiyonu) geliştiricilere esneklik sağlar.

DI Örneği

- Senaryo: Diyelim ki bir "Logger" (Günlükleyici) sınıfınız var ve bu sınıf, bir loglama hizmeti sağlar. Bir "UserService" (Kullanıcı Hizmeti) sınıfınız da var ve kullanıcı işlemlerini gerçekleştirirken loglama yapmak istiyorsunuz. Bu durumda, Logger sınıfını UserService'e enjekte ederek Dependency Injection kullanabiliriz.

JAVA

```
public class Logger {  
    public void log(String message) {  
        System.out.println("Log: " + message);  
    }  
}
```

```
public class UserService {  
    private Logger logger;  
  
    // Constructor Injection ile Logger enjekte ediliyor  
    public UserService(Logger logger) {  
        this.logger = logger;  
    }  
  
    public void addUser(String username) {  
        // Kullanıcı ekleme işlemi  
        logger.log("User added: " + username);  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        // Logger'ı oluştur  
        Logger logger = new Logger();  
  
        // UserService'e Logger'ı enjekte et  
        UserService userService = new UserService(logger);  
  
        // Kullanıcı ekleme işlemini gerçekleştir  
        userService.addUser("john_doe");  
    }  
}
```

Spring

```
public class Logger {  
    public void log(String message) {  
        System.out.println("Log: " + message);  
    }  
}
```

```
@Service  
public class UserService {  
    private Logger logger;  
  
    @Autowired  
    public UserService(Logger logger) {  
        this.logger = logger;  
    }  
  
    public void addUser(String username) {  
        // Kullanıcı ekleme işlemi  
        logger.log("User added: " + username);  
    }  
}
```

```
@Component  
public class AppRunner implements CommandLineRunner {  
  
    private final UserService userService;  
  
    @Autowired  
    public AppRunner(UserService userService) {  
        this.userService = userService;  
    }  
  
    @Override  
    public void run(String... args) {  
        userService.addUser("john_doe");  
    }  
}
```

Spring Boot Enjeksiyon türleri

- ▶ Spring Boot, Dependency Injection (DI) işlemlerini gerçekleştirmek için farklı enjeksiyon türleri sunar. Bu enjeksiyon türleri, Spring Boot uygulamalarını farklı şekillerde yapılandırmak ve bağımlılıkları enjekte etmek için kullanılır. İşte Spring Boot'ta kullanabileceğiniz bazı DI türleri:

Constructor Injection

- Bu enjeksiyon türünde, bağımlılıklar bean'in kurucusu (constructor) aracılığıyla enjekte edilir. Spring Boot, otomatik olarak uygun constructor'ı tanımlar ve enjeksiyonu gerçekleştirir.

```
@Service
public class UserService {
    private final Logger logger;

    @Autowired
    public UserService(Logger logger) {
        this.logger = logger;
    }
}
```

Setter Injection

- Bu enjeksiyon türünde, bağımlılıklar sınıfın setter yöntemleri aracılığıyla enjekte edilir. Bu yöntem daha esnek bir enjeksiyon sağlar

```
@Service
public class UserService {
    private Logger logger;

    @Autowired
    public void setLogger(Logger logger) {
        this.logger = logger;
    }
}
```

Field Injection

- Bu enjeksiyon türünde, bağımlılıklar doğrudan sınıfın özelliklerine (field) enjekte edilir. Bu yöntemde setter veya constructor kullanmadan enjeksiyon yapabilirsiniz.


```
@Service
public class UserService {
    @Autowired
    private Logger logger;
}
```

Method Injection

- Bu enjeksiyon türünde, bir sınıfın yöntemleri aracılığıyla bağımlılıklar enjekte edilir.


```
@Service
public class UserService {
    private Logger logger;

    @Autowired
    public void injectLogger(Logger logger) {
        this.logger = logger;
    }
}
```

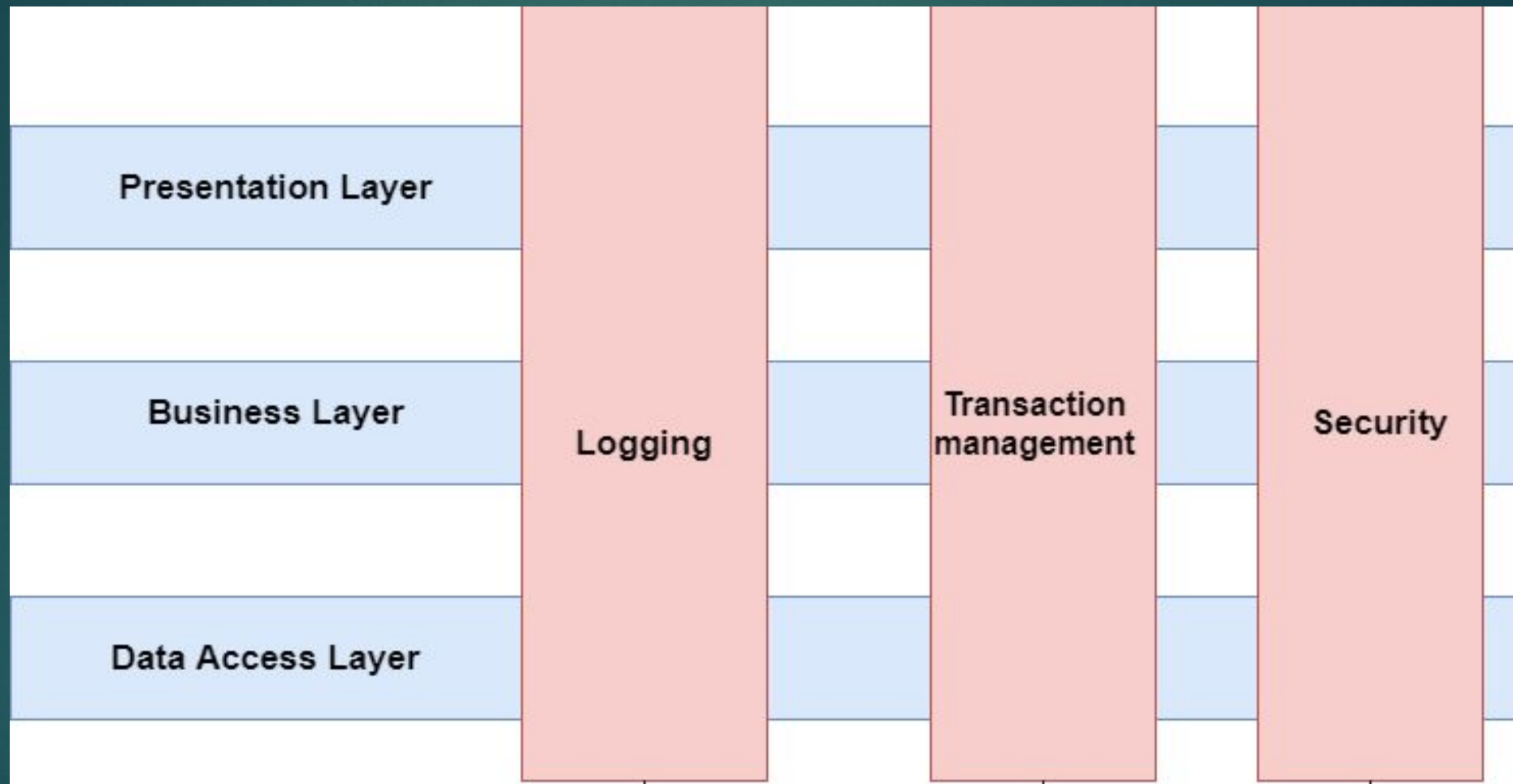

- 
- ▶ Spring Boot, bağımlılıkları otomatik olarak enjekte etmek için @Autowired anotasyonunu kullanır. Bu sayede bağımlılıkları manuel olarak belirtmek zorunda kalmazsınız ve Spring Framework, uygun enjeksiyonu otomatik olarak gerçekleştirir.

Araştırınız

- ▶ Spring Boot, genellikle Constructor Injection'u teşvik eder, bunun nedeni nedir?

- 
- ▶ Inversion of Control (IoC - Kontrolün Tersine Çevrilmesi): Spring Core'un temel prensiplerinden biridir. Bu prensibe göre, nesnelerin yaşam döngüsü ve yönetimi Spring Framework'e devredilir. Geliştirici, bu bileşenleri oluşturma ve yönetme sorumluluğundan kurtulur.
 - ▶ Bean Yönetimi: Spring Core, bean'leri (nesneleri) yönetmek için gelişmiş bir mekanizma sunar. Bean'lerin oluşturulması, yaşam döngüsü yönetimi, imha edilmesi gibi işlemleri otomatik olarak gerçekleştirir.

- ▶ Spring Boot, Aspect-Oriented Programming (AOP) desteği sağlar. AOP, bir yazılım tasarım yaklaşımıdır ve uygulamanın çapraz kesen endişelerini (cross-cutting concerns) ele almayı amaçlar. Çapraz kesen endişeler, uygulamanın farklı yerlerinde (sınıflar, metotlar, paketler vb.) tekrar eden ve genellikle işlevselliğin dışında olan kod parçacıklarıdır. Örnek olarak, loglama, güvenlik, hata yönetimi gibi çapraz kesen endişelere sahip olabilirsiniz.
- ▶ Spring Boot AOP, bu çapraz kesen endişeleri ayrı bir katman olarak ele almanızı ve kodunuzu daha temiz ve düzenli hale getirmenizi sağlar. AOP, uygulama kodunu etkilemeden belirli noktalarda (örneğin, metot çağrıları öncesinde veya sonrasında) kodu otomatik olarak enjekte ederek çapraz kesen endişeleri ele alır.



Örnek Senaryo: Bir ticaret uygulamasında ürün stok yönetimi yapılıyor. Ürünlerin stok seviyeleri düşük olduğunda bir uyarı logu oluşturulması gerekiyor.

```
import org.springframework.stereotype.Service;

@Service
public class ProductService {

    private int productStock = 15;


    public int updateStock(int quantity) {
        // Stok güncelleme işlemi
        productStock -= quantity;
        return productStock;
    }
}
```


```
@Aspect
@Component
public class StockAlertAspect {


    @AfterReturning(
        pointcut = "execution(* com.example.service.ProductService.updateStock(..))",
        returning = "result")
    public void afterStockUpdate(JoinPoint joinPoint, Object result) {
        String methodName = joinPoint.getSignature().getName();
        System.out.println("Stock updated by method: " + methodName);

        int updatedStock = (int) result;

        if (updatedStock < 10) {
            System.out.println(x:"Low stock alert! Product stock is below 10.");
        }
    }
}
```

- 
- ▶ Bu Aspect sınıfı, ProductService sınıfındaki updateStock metodu çalıştırıldığında, stok güncellemesi sonrasında stok seviyesini kontrol eder. Eğer stok seviyesi 10'un altında ise uyarı logu oluşturur.
 - ▶ Uygulamayı çalıştırdığınızda, ProductService'deki updateStock metodu çağrıldığında Aspect tarafından yakalanacak ve gerektiğinde uyarı logu üretilecektir.

- 
- ▶ Internationalization (Çokdillilik) ve Localization (Yerelleştirme): Spring, çokdillilik ve yerelleştirme desteği sağlar. Bu sayede uygulamanın farklı dillerdeki metinleri yönetilebilir ve farklı bölgelere özgü formatlamalar gerçekleştirilebilir.
 - ▶ Event Handling (Olay İşleme): Spring Core, olay tabanlı programlamayı destekler ve uygulama içindeki olayların üretilmesi, dinlenmesi ve işlenmesi için bir mekanizma sunar.

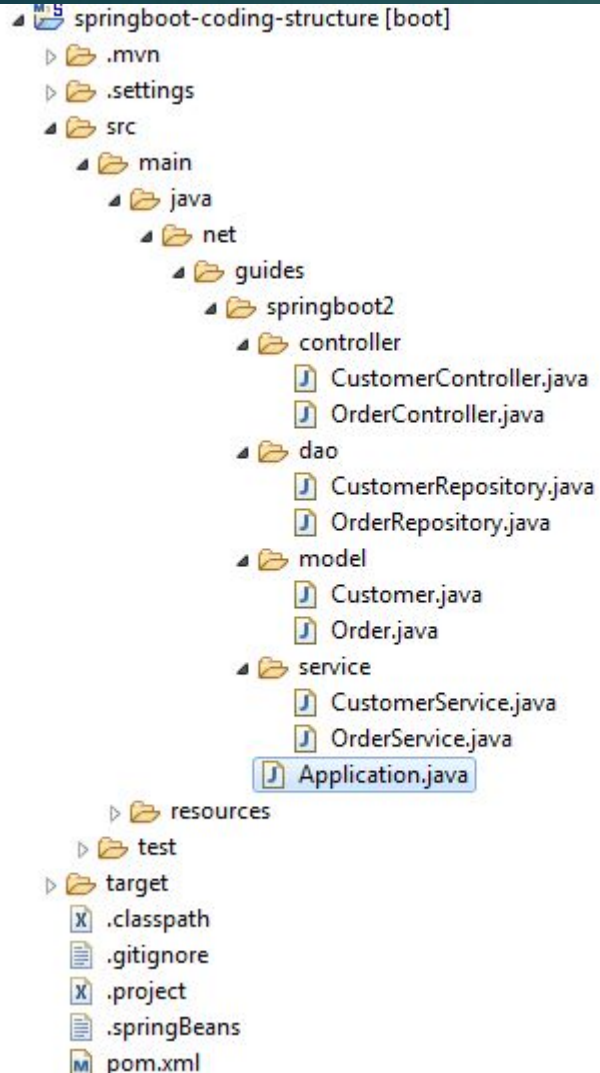
- 
- ▶ Resource Management (Kaynak Yönetimi): Spring, veritabanları, dosyalar, ağ bağlantıları gibi kaynakların etkin ve güvenli bir şekilde yönetilmesini kolaylaştıran bir dizi araç ve sınıf sağlar.
 - ▶ Validation ve Data Binding (Doğrulama ve Veri Bağlama): Spring, giriş doğrulamasını kolaylaştıran ve giriş verilerini nesnelere bağlamayı sağlayan bir dizi araç ve sınıf sunar.

Spring Boot projesi oluşturma

- ▶ Spring Initializr
- ▶ Spring Initializr, Spring Boot projelerini hızlı bir şekilde oluşturmanıza yardımcı olan bir web tabanlı araçtır. Initializr, projenizin temel yapılandırmasını oluşturmanıza, bağımlılıkları seçmenize ve projenizi indirip kullanmaya başlamanıza olanak tanır.

- ▶ Proje Ayarlarını Belirleme: Initializr'ı açtığınızda, projenizin temel ayarlarını belirleyebilirsiniz. Bu, proje adı, dil (Java veya Kotlin), proje paket adı, proje tipi (Maven veya Gradle) gibi bilgileri içerir.
- ▶ Bağımlılıkları Seçme: Projenize hangi bağımlılıkların eklenmesini istediğinizi seçebilirsiniz. Örneğin, veritabanı erişimi, güvenlik, web uygulama, test çerçeveleri gibi çeşitli bağımlılıkları seçebilirsiniz.
- ▶ Proje Oluşturma: Ayarları yapılandırdıktan sonra, "Generate" veya benzeri bir düğmeye tıkladığınızda Initializr proje paketinizi oluşturur ve bir ZIP dosyası olarak indirmenize izin verir.
- ▶ Proje Kullanmaya Başlama: İndirdiğiniz ZIP dosyasını açarak projeyi kendi geliştirme ortamınıza (IDE) veya metin düzenleyicinize aktarabilirsiniz. Ardından projenizi çalıştırarak Spring Boot uygulamanızı başlatabilir ve geliştirmeye başlayabilirsiniz.

Spring Boot Proje Yapısı





- ▶ `src/main/java`: Bu dizin, uygulamanızın Java kaynak kodlarını içerir. Bu klasör altında, Spring bileşenlerinizi (Controller'lar, Service'ler, Repository'ler vb.) ve diğer Java sınıflarınızı oluşturabilirsiniz.
- ▶ `src/main/resources`: Bu dizin, uygulamanızın kaynaklarını (konfigürasyon dosyaları, statik içerik, şablonlar vb.) içerir. Örneğin, `application.properties` veya `application.yml` gibi yapılandırma dosyalarını burada bulabilirsiniz.
- ▶ `src/test/java`: Bu dizin, test sınıflarınızı içerir. Unit testleri veya entegrasyon testleri için kullanabilirsiniz.
- ▶ `src/test/resources`: Bu dizin, test kaynaklarını içerir. Testlerinizde kullanacağınız örnek veriler, konfigürasyon dosyaları veya diğer test kaynakları burada yer alabilir.

- ▶ `pom.xml` (veya `build.gradle`): Projenizin yapılandırması ve bağımlılıklarının belirtildiği dosyadır. Maven veya Gradle gibi yapılandırma araçları kullanarak projenizin bağımlılıklarını ve yapılandırmasını yönetebilirsiniz.
- ▶ `target` (veya `build`): Bu dizin, projenin derlenmiş ve paketlenmiş sürümünü içerir. Derlenmiş Java sınıfları, paketlenmiş JAR veya WAR dosyası burada bulunabilir.
- ▶ `mvnw` (veya `mvnw.cmd`): Maven Wrapper veya Gradle Wrapper dosyalarıdır. Bu dosyalar, projenin herhangi bir Maven veya Gradle yüklemeden derlenmesini ve çalıştırılmasını sağlar. Proje bağımlılıkları otomatik olarak indirilir.
- ▶ `mvnw` (veya `mvnw.cmd`): Maven veya Gradle komut dosyasıdır. Projeyi komut satırından derlemek ve çalıştırmak için kullanılır.
- ▶ `.gitignore`: Git depolama alanına eklenmemesi gereken dosya veya dizinleri belirtmek için kullanılır. Örneğin, derleme çıktıları veya IDE özellikleri burada belirtilebilir.
- ▶ `README.md`: Proje hakkında temel bilgileri içeren dokümantasyon dosyasıdır. Genellikle proje yazarı, amacı, kullanımı ve diğer önemli bilgileri burada açıklar.

Resmi Dökümantasyonu Anlamak

- ▶ Temel Bilgileri Edinin: İlk adım, Spring'in temel konseptlerini ve terminolojisini öğrenmektir. Spring'in temel kavramlarına aşina olmadan dökümantasyonu anlamak zor olabilir. Spring Framework ve Spring Boot hakkında temel bilgi sahibi olun.
- ▶ Resmi Dökümantasyonu İnceleyin: Spring Framework ve Spring Boot resmi web sitelerinde geniş bir dökümantasyon koleksiyonu bulunmaktadır. Bu dökümantasyonu inceleyerek, temel yapı ve bileşenler hakkında bilgi edinebilirsiniz.
- ▶ Başlangıç Rehberleri ve Örnekler: Spring Framework ve Spring Boot'un resmi dökümantasyonunda, başlangıç rehberleri ve örnek uygulamalar yer alır. Bu rehberler, temel konseptleri adım adım açıklar ve pratik örnekler sunar. Bu rehberleri izleyerek, uygulamalı olarak nasıl yapılandırma ve kodlama yapmanız gerektiğini öğrenebilirsiniz.

- 
- ▶ Dökümantasyonu Parçalayın: Tüm dökümantasyonu tek seferde anlamaya çalışmak yerine, ihtiyacınız olan belirli konulara odaklanın. Hangi konuların sizin için daha öncelikli olduğunu belirleyin ve bu konuları öncelikli olarak inceleyin.
 - ▶ Örnek Kodları İnceleyin: Spring dökümantasyonu genellikle örnek kodlar içerir. Bu örnekleri inceleyerek, konseptleri gerçek dünya örnekleriyle nasıl uygulayabileceğinizi daha iyi anlayabilirsiniz.
 - ▶ Topluluk ve Forumları Kullanın: Spring topluluğu oldukça aktiftir ve çeşitli forumlar ve tartışma platformları mevcuttur. Sorularınızı sormak ve zorlandığınız konularda yardım almak için bu platformları kullanabilirsiniz.

- 
- Uygulama ve Deneyim: Dökümantasyonu anlamak için teorik bilgilerin yanı sıra pratik deneyime de ihtiyacınız vardır. Öğrendiklerinizi gerçek projelerde uygulayarak ve deneyim kazanarak konseptleri daha iyi anlayabilirsiniz.