Java ve Spring Boot ile Rest Api

Nurcan Şensoy

Sunum İçeriği

1.Spring nedir?

2.Spring Boot nedir?

3.Spring ve Spring Boot arasındaki farklılıklar nedir?

4.Spring Boot'un avantajları nelerdir?

5.Rest Api nedir?

    5.1.Rest Prensipleri

6.Katmanlı mimari nedir?

7.Bir Spring Boot projesi nasıl oluşturma örneği

Spring Nedir?

Java ile geliştirme yapmayı kolaylaştıran, açık kaynak kodlu Core Container, AOP, Data Access, Web gibi modüllerden oluşan bir framework-kütüphanedir.

Spring, OOP tabanlı bir tasarım sağlar.

Spring'deki en önemli kavram Dependency Injection’dır.


Sınıf A
A sınıfı, B sınıfının bazı metodlarını kullanıyor
Sınıf B
Bu bir bağımlılıktır

Spring Kütüphanesinin Yararları

- Spring kütüphanesi bir web uygulamasının tüm katmanlarına uygulanabilir.
- Gevşek bağlılık (Loose Coupling) ve kolay test edilebilirlik sağlar.
- XML ve Annotation konfigürasyonlarını destekler.
- Singleton ve Factory sınıflarının ortadan kaldırılması için gerekli özelliğe sahiptir.
- Bildirimsel (Declarative) programlamayı destekler.

Spring Boot Nedir?

Java ile veya **Spring Framework** ile bir uygulama geliştirildiğinde xml dosyaları üzerinden konfigürasyon işlemleri yapılmaktadır. Bu işlemler özellikle projeler büyüdükçe vakit alabilmekte ve karmaşıklığa yol açabilmektedir. **Spring Boot** ise bu vakit kaybını en aza indirmeye çalışan bir Spring framework'üdür.

Spring Boot'un Yararları

- Bağımsız (stand-alone) uygulamalar oluşturur.
- Gömülü olarak Tomcat, Jetty veya Undertow ile birlikte gelir.
- XML konfigürasyonuna ihtiyaç duymaz.
- LOC (Lines of Code) 'u azaltmayı hedefler.
- Başlatması kolaydır.
- Özelleştirme ve yönetim basittir.

Spring ve Spring Boot Arasındaki Fark

Spring kütüphanesi bize esneklik uygulamaya odaklanırken, Spring Boot kod uzunluğunu kısaltmayı ve bir web uygulaması geliştirmenin en kolay yolunu bize sunmayı amaçlamaktadır.

Rest Api Nedir?

Representational state transfer; İlgili isteğe karşılık gelen verinin JSON / XML gibi dosya formatlarında gönderilmesidir. REST API, REST mimarisinin prensiplerine taşıyan API'lardır. Tüm prensiplerin karşılanması durumunda RESTful API olarak da adlandırılır.

Rest Prensipleri

- İstemci – Sunucu: (Client – Server)
- Tek Tip Arayüz: (Uniform Interface)
- Durumsuzluk: (Statelessness)
- Önbelleklenebilir: (Cacheable)
- Katmanlı Sistem: (Layered System)
- İsteğe Bağlı Kod: (Code On Demand – Optional)

İstemci – Sunucu (Client – Server) Prensibi

İstemci isteği gönderen, sunucu da ilgili cevabı veren durumundadır. Birbirlerinin sorumluluk alanlarına girmezler. Birbirlerinden bağımsız programlama dilleri ve teknolojiler kullanabilirler.

Tek Tip Arayüz (Uniform Interface) Prensibi

Aynı kaynağa yönelik olan tüm istekler, isteğin nereden geldiğinden bağımsız olarak aynı şekilde görünmelidir. Bu aynı zamanda istemci – sunucu bağımsızlığını da destekler.

İstemci Durumsuzluk (Statelessness) Prensibi

STATE

- Söz konusu veriyi – durumu belirtir.
- **Stateful** (Durum bilgisi olan) vs **Stateless** (Durum bilgisi olmayan) İstemci tafından gerçekleştirilen her istek birbirinden bağımsızdır ve sunucu bu isteklerin her birini bağımsız olarak değerlendirir. Sunucu istemci tarafından kendisine gönderilen bilgileri tutmamalıdır.

Önbelleklenebilir (Cacheable) Prensibi
Sunucu gelen isteklere verilen cevapların önbelleklenebilir olup olmadığını belirtmelidir. Örneğin "**Cache-Control**", **"Expires"** gibi HTTP başlıkları önbellek ile ilgili bilgiler taşır.

Katmanlı Sistem (Layered System) Prensibi
İstemci – sunucu arasındaki ilişki katmanlara ayrılabilir ve bileşenler sadece ilişkili oldukları katmanlara karşı sorumlu olurlar.

İsteğe Bağlı Kod (Code On Demand – Optional) Prensibi
Sunucu, istemci tarafına istemcinin işlevini genişletecek ek kodlar gönderebilir. Bu özellik istemci tarafında yapılması gereken işlemleri hafifletir.

Katmanlı Mimari Nedir?

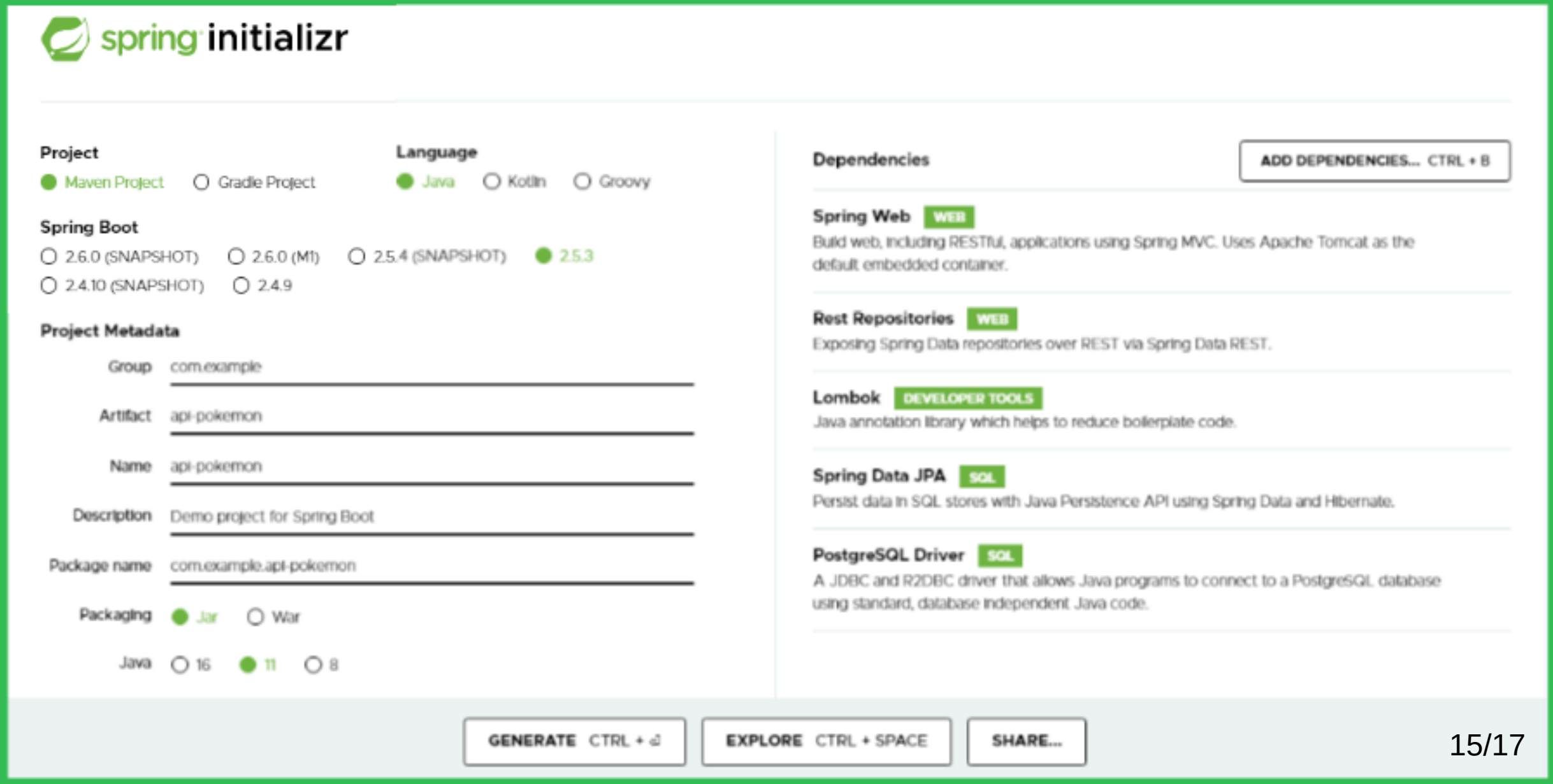
DataAccess: Bu katman veri tabanından veriye erişmek için yazılacak olan kodların bulunduğu katmandır. Örnek olarak **Java** da **Hibernate** kodlarının yazılacağı katman burasıdır. Yine bir başka deyişle **C#** da kullanılan ORM teknolojisi olan **Entity Framework** kodlarının yazılacağı katman burasıdır.

Entities : Bu katman veri tabanı nesnelerinin tutulduğu katmandır. Yani kısaca veri tabanında bulunan tabloların yazılım dilinde karşılığının tutulduğu katmandır. Veri tabanında tablo bu katmanda class a eşit gelmektedir.

Business: Bu katman ise işlerin yapıldığı katmandır. Belirlenen algoritmalara göre verilerin şekillenmesini sağlayan logic dediğimiz işlemlerin yapıldığı katmandır. Örnek olarak veri tabanından belirli bir kurala göre verileri çekmek istiyorsak o kuralı bu iş katmanında belirlediğimiz metot içerisinde uygulayarak istediğimiz veriyi elde etmiş oluruz.

User Interface(UI): Bu katman isminden de anlaşılacağı gibi projemizin kullanıcılarla iletişime ve etkileşime geçtiği katmandır. Bu katmanda kullanıcılara bir ara yüz sunulur ve kullanıcılar bu ara yüz ile beraber oluşturmuş olduğumuz proje ile etkileşime geçer.

Spring Boot Projesi Oluşturma



Maven: Maven build işlemlerini sadeleştirmek için kullanılan bir inşa aracıdır ve ‘build’ , ‘reporting’, ‘dependency management’ , ‘documentation’ işlemlerini yapar. Konfigürasyon için **pom.xml** dosyasını kullanır. Bu dosya projeyi build etmek için gereken bütün bilgileri içerir. Maven bir görevi yerine getirmeden önce çalışma dizininde POM'u bulur ve gerekli bilgileri okur.

Gradle: Derleme otomasyonuna ve çoklu dil geliştirmesine destek veren bir **build tool**’dur. **Android** başta olmak üzere **Java**, **Scala**, **Groovy**, **C**, **C++** gibi birçok dilde kullanılmaktadır. **Open Source**(Açık Kaynak) kodlu bir build tool’dur.

“Dependency”lerden seçilen **Lombok**, Java projesi geliştirirken IDE’ye entegre edilebilen, anotasyon ve kod üretme kütüphanesidir. Daha temiz ve az kod yazmayı sağlar.

Bir diğeri SQL sorguları için seçilen **“Spring Data JPA”** dir. JPA tarafından belirlenen şartları yerine getiren Hibernate, EclipseLink gibi kütüphaneleri kullanarak ilişkisel veritabanları üzerinde işlem yapmayı sağlayan Spring Data projesidir.
Bir diğer "dependency" veritabanına bağlamak için gerekli olan **PostgreSQL Driver'dır.**

Beni dinlediğiniz için teşekkür ederim.