

Temel Spring Boot Anotasyonları

- **@Bean** - Bir metodun Spring tarafından yönetilen bir Bean ürettiğini belirtir
- **@Service** - Belirtilen sınıfın bir servis sınıfı olduğunu belirtir.
- yazılımın bussines logic kısmını gerçekleştiren sınıflar için kullanılır.
- **@Repository** - Veritabanı işlemlerini gerçekleştirmeye yeteneği olan yapıldığı repository sınıfını belirtir.
- **@Controller** - Requestleri yakalayabilme yeteneği olan bir web controller sınıfını belirtir. frontend veya api isteklerinin karşılayan sınıfları tanımlamak için kullanılır.
- **@RestController** tanımı **@Controller** tanımının özelleşmiş halidir.
- **@RequestMapping** - controller sınıfının handle ettiği HTTP Requestlerin path eşlestirmesini yapar. Genel (ana) mapping anotasyonudur
- **@Autowired** - Constructor, Değişken yada setter metodlar için dependency injection işlemi gerçekleştirir. Her run da çalışmaz, çalışması gerekiğinde çalıştırır
- **@SpringBootApplication** - Spring Boot autoconfiguration ve component taramasını aktif eder.

@RequestBody

- HTTP isteğin gövdesini deserialize ederek, tanımlanan sınıfın objesine dönüştürür.
- Gelen istediği başlığını kontrol edip, istek gövdesini json'dan domain objesine deserialize eder.
- **@ResponseBody** varsayılan olarak tanımlanır.

@Entity

- Tanımlanan sınıfın bir JPA varlığı olduğunu belirtir.
- Uygulama çalıştırıldıktan sonra veritabanı işlemlerinin buradaki verilere göre yapılacağı tanımlanır.
- Eğer **@Entity** veya **@Table** içinde ek bir bir belirtim yapılmadıysa, sınıf adı tablo adı olur.
- **@Table** belirtimi yalnızca veritabanı ile ilgili yapmak istediğimiz özel belirtimler için kullanılır.
- Entity sınıfı bir POJO sınıfıdır. Final yada Inner class olarak tanımlanamaz.
- Veritabanında oluşan tablodaki her satır bir entity set olarak tanımlanır.

@PathVariable: *{id}* değişkeni tarafından temsil edilen URI'nin şablonlanmış kısmını çıkarmak için **@PathVariable** ek açıklamasını kullanıyoruz .

@RequestParam ve **@PathVariable** , istek URI'sinden değerleri çıkarmak için kullanılabilir, ancak bunlar biraz farklıdır.

@RequestParam s değerleri sorgu dizesinden alırken, **@PathVariable** s URI yolundan değerleri alır :

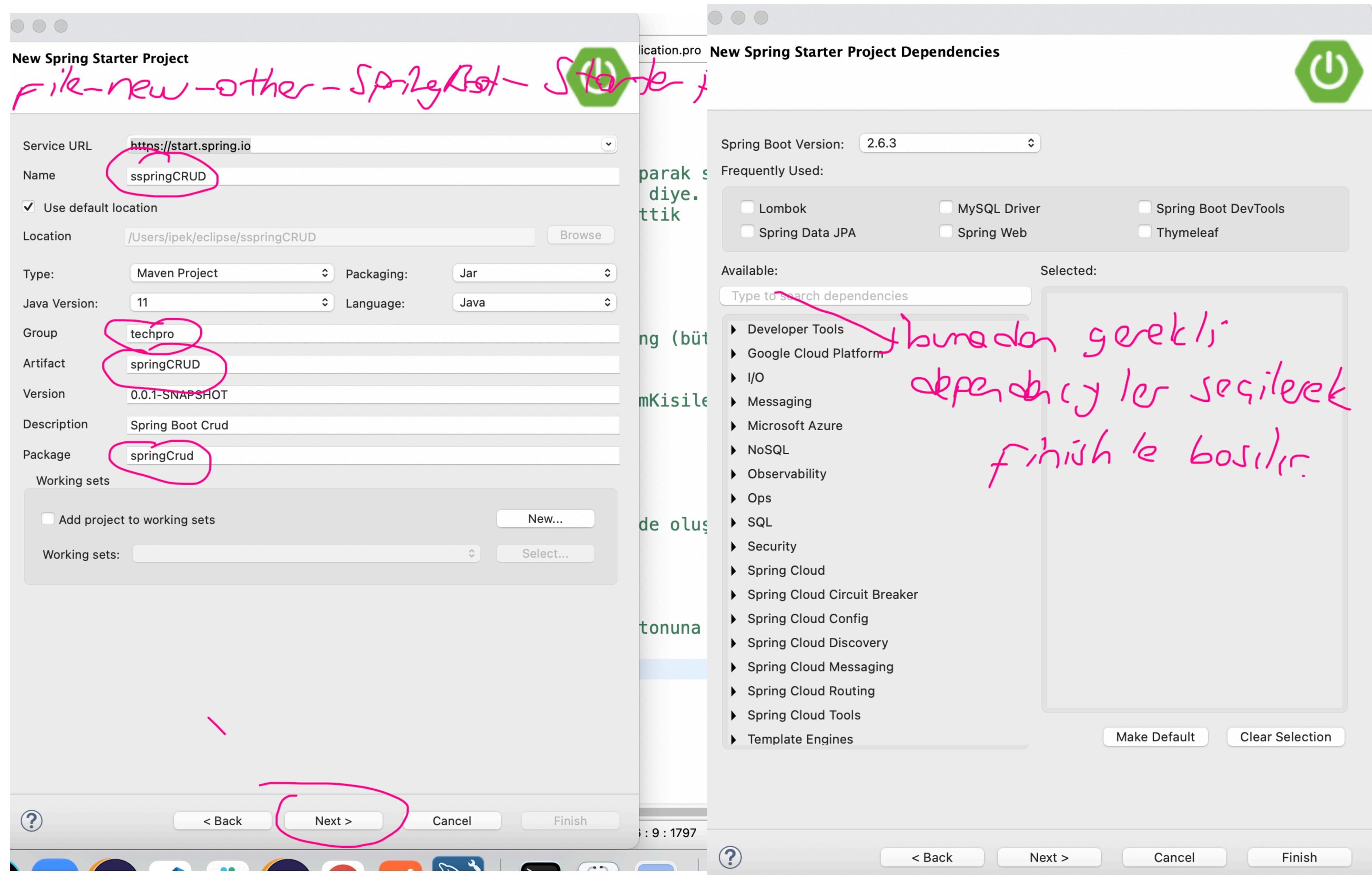
```
@GetMapping("/foos/{id}")
@ResponseBody
public String getFooById(@PathVariable String id) {
    return "ID: " + id;
```

@RequestParam için şöyle olacaktır:

```
@GetMapping("/foos")
@ResponseBody
public String getFooByIdUsingQueryParam(@RequestParam String id) {
    return "ID: " + id;
}
```

Springboot projesi oluşturma

1.yol eclipse e marketplace den indirilen ses sayesinde, eclipse içinde oluşturulur



2. Yol

start.spring.io

Uygulamalar IT google translate -... Winter Turkish 2021 Developer My Meetings - Zo... Slack | live_chann... Google Translate

ingilizce Türkçe

spring® initializer

yazdır spring initializer sık sık
aynı şekilde kullanılarak
dependency'ler sağlanır

Project

Maven Project
 Gradle Project

Language

Java Kotlin
 Groovy

Dependencies

No dependency selected

ADD DEPENDENCIES... ⌘ + B

Spring Boot

3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT)
 2.7.0 (M1) 2.6.4 (SNAPSHOT) 2.6.3
 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group com.example

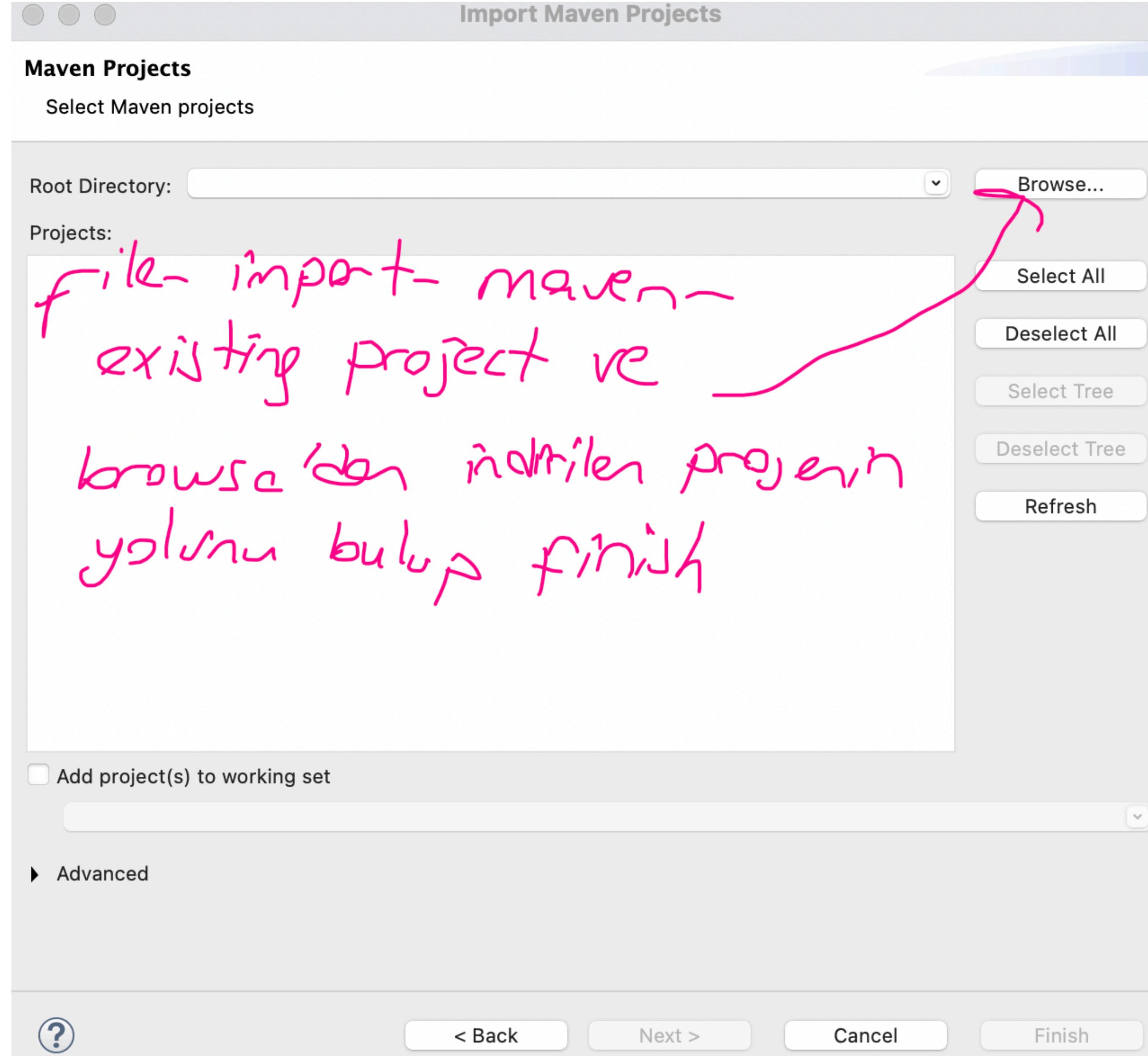
Artifact demo

Name demo

generate edilir. Daha sonra zip dosyasına
kopyalanır.

GENERATE ⌘ + ↩ **EXPLORE CTRL + SPACE** **SHARE...**

Sonra alttaki işlemlerle eclipse import edilir



```
> M S CRUD (in CRUD2) [boot] [devtool]
> M J hibernate1
> M J kdemo (in hibernate1)
> M S springboothymeleaf [boot] [devt]
> M S springCRUD [boot] [devtools]
    > Spring Elements
    > src/main/java
        > springCrud
            > SpringCrudApplication.java
        > springCrud.controller
            > HomeController.java
            > KisiController.java
        > springCrud.model
            > Kisi.java
        > springCrud.repository
            > KisiRepository.java
        > springCrud.service
            > KisiService.java
    > src/main/resources
        > static
        > templates
        > application.properties
        > src/test/java
> M J JRE System Library [JavaSE-11]
> M J Maven Dependencies
    > src
        > main
            > java
            > resources
            > webapp
                > index.html
        > test
    > target
        > HELP.md
        > mvnw
        > mvnw.cmd
        > pom.xml
> M S SpringMVC [boot] [devtools]
> M S thymeleaf [boot]
```

proje adı : *spring crud* (run class'ını içeriyor)

run class'ı : *src/main/java/springCrud/SpringCrudApplication.java*

run package : *src/main/java/springCrud* içindeki paketler

Frontend ve backend arası iletişimini sağlar
veritabanı kullanılır
service层i metodu: *findById()*
yada başka frontend sağlıyor.

Projemizin tasarım olur, table yada sınıf
JPA sayesinde veritabanı ile bağlantısı
ve, JPA'de altyapı metodları kullanarak backend
için yapacak fonksiyonları oluşturur.
Database bağlantılı, bulur

dependency'le ekler:

**Bütün package ler (controller, service, model...) ana run class in package sinin altına oluşturulmazsa,
alttaki anotasyonlarla hiyerarşî belirlenir**

@SpringBootApplication

**//@ComponentScan({"controller","service"})//aradığını bulabilsin diye
package leri belirtiyorum.
///temel olanlar bu şekilde (controller,service). model gibiler alttaki gibi**

//@EntityScan("model")

//@EnableJpaRepositories("repository")