2023-2024 EĞITIM-ÖĞRETIM YILI BAHAR DÖNEMI YAZILIM GELİŞTİRME VE ORTAM ARAÇLARI FİNAL PROJESI

MEDİPOL ÜNİVERSİTESİ

AKIF MARKET STOK TAKIP SISTEMI

PROJE BILGISI: BU PROJEDE .... <<HTTPS://GITHUB.COM/AKIFBERABASAR/FINALODEVI>>'INDE ÇALIŞMALAR PAYLAŞILMIŞTIR.

ADI SOYADI/NAME SURNAME: AKİF BERA BAŞAR
NUMARASI/NUMBER: H5220006

KONU BAŞLIKLARI

- MAVEN
- SPRING
- JUNIT
- POSTMAN
- JMETER
- UML
- GIT

Proje yazılırken JAVA dili kullanılmıştır.

Kodlar Eclipse platformunda yazılmıştır.

POM.XML

```
<groupId>tr.edu.medipol.yazilimgelistirme</groupId>
<artifactId>FinalOdeviServis</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>FinalOdeviServis</name>
<description>Final Odevi Servis </description>
```

POM.XML, APACHE MAVEN TARAFINDAN KULLANILAN BIR YAPILANDIRMA DOSYASIDIR. "POM" KISALTMASI, "PROJECT OBJECT MODEL"I TEMSIL EDER VE BIR MAVEN PROJESININ YAPILANDIRMA, BAĞIMLILIKLAR, DERLEME AYARLARI VE DIĞER PROJEYE ÖZGÜ BILGILERI IÇERIR. POM DOSYASI, MAVEN TARAFINDAN KULLANILARAK PROJE YÖNETIMI VE DERLEME IŞLEMLERININ GERÇEKLEŞTIRILMESINI SAĞLAR.

MAVEN

```
<properties>
    <java.version>17</java.version>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
</properties>
```

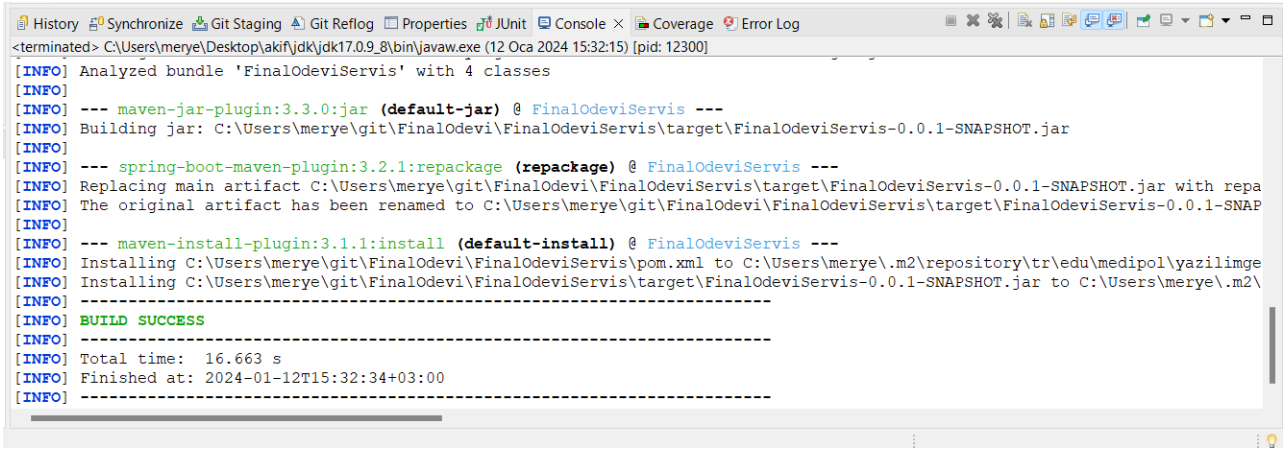
Maven projemizi oluşturduktan sonra pom.xml dosyamıza gerekli özelliklerle doldurmaya başlıyoruz.

<maven.compiler.source> etiketi, Maven projesinin Java kaynak kodunun hangi Java sürümü ile yazıldığını belirtir.

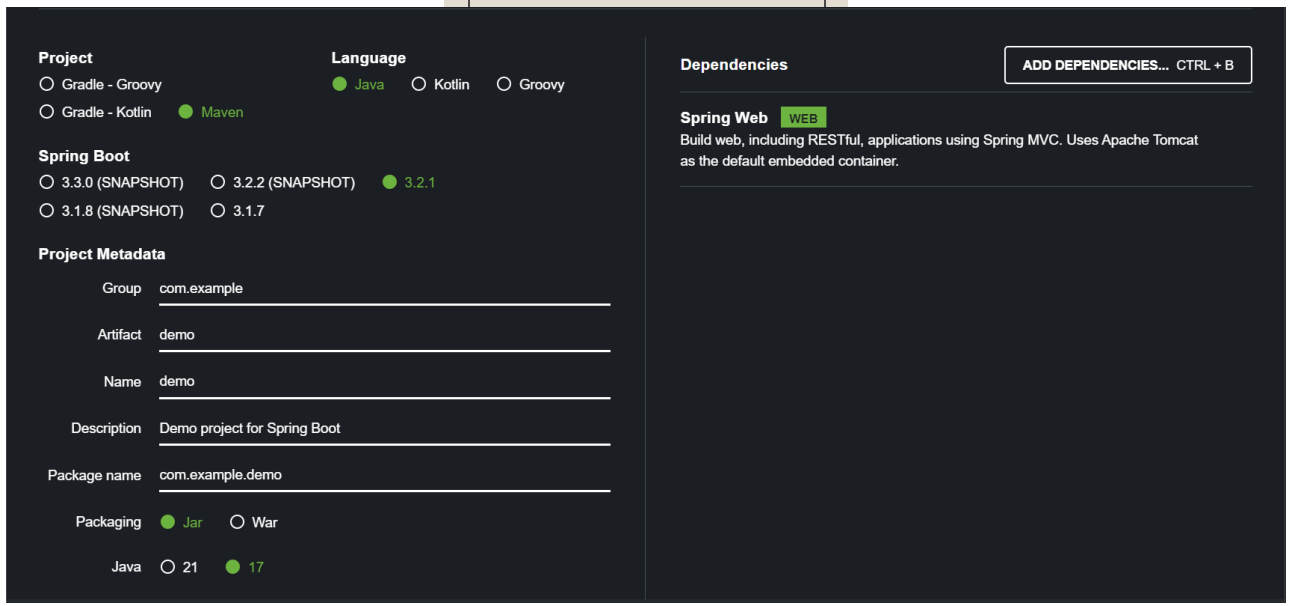
<maven.compiler.target> etiketi, Maven projesinin derlendikten sonra hangi Java sürümü hedeflendiğini belirtir.

MAVEN

Maven projeye başarılı şekilde eklenmiştir.



SPRING



https://start.spring.io/ adresinden kendimize web servisi oluşturuyoruz.

BİR ÖNCEKİ SLAYTTA YER ALAN GÖRSELDEKİ VERİLERİ PROJEMİZİN POM.XML KISMINDA YAZAN VERİLERE GÖRE DOLDURACAĞIZ.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
```

SPRİNG'İ PROJEMİZE EKLİYORUZ.

JUNIT

JUnit, Java dilinde yazılmış yazılım projelerinde birim testlerin otomatik olarak yapılmasını sağlayan bir test çerçevesidir (testing framework). JUnit, yazılım geliştiricilere birim testlerini yazmak, yönetmek ve çalıştırmak için bir dizi araç ve sınıf sağlar.

```
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version><!--$NO-MVN-MAN-VER$-->
        <scope>test</scope>
    </dependency>
</dependencies>
```

PROJEMİZE JUNIT'İ EKLİYORUZ.

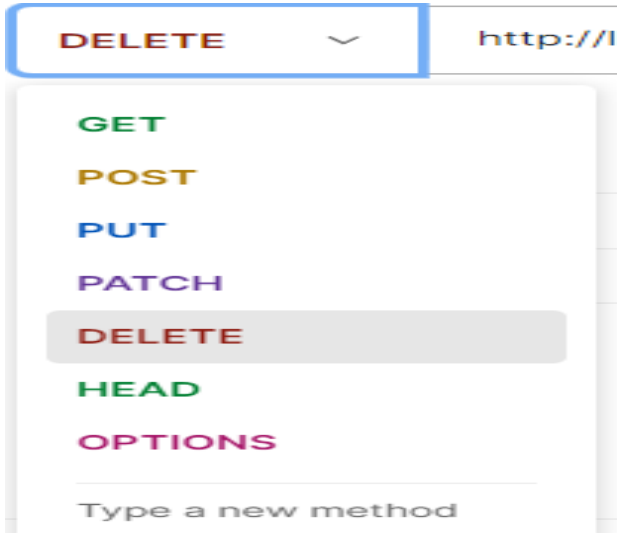
JUNIT

Aşağıda JUNIT test örneği verilmiştir.

```
@Test
public void testStart() {
    System.out.println("JUnit test başlatıldı.");
    fail("Test başlatılamadı.");
}
```

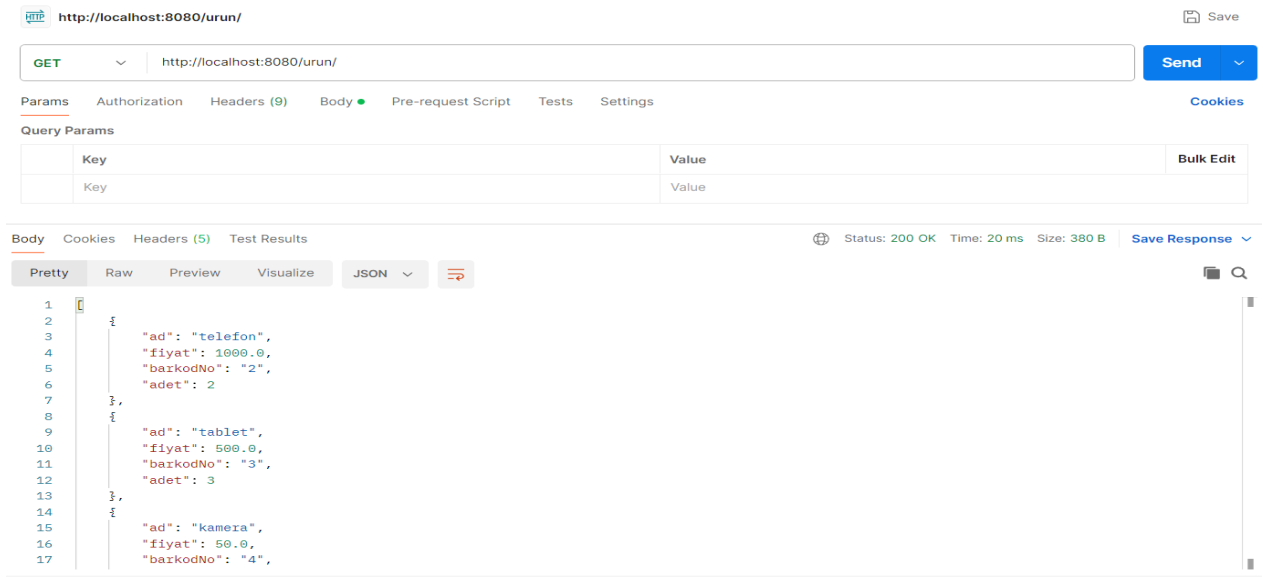
POSTMAN

Postman, özellikle geliştiricilerin API'larını test etmeleri, hata ayıklamaları, belgelemeleri ve işbirliği yapmaları için yaygın olarak kullanılan bir araçtır.



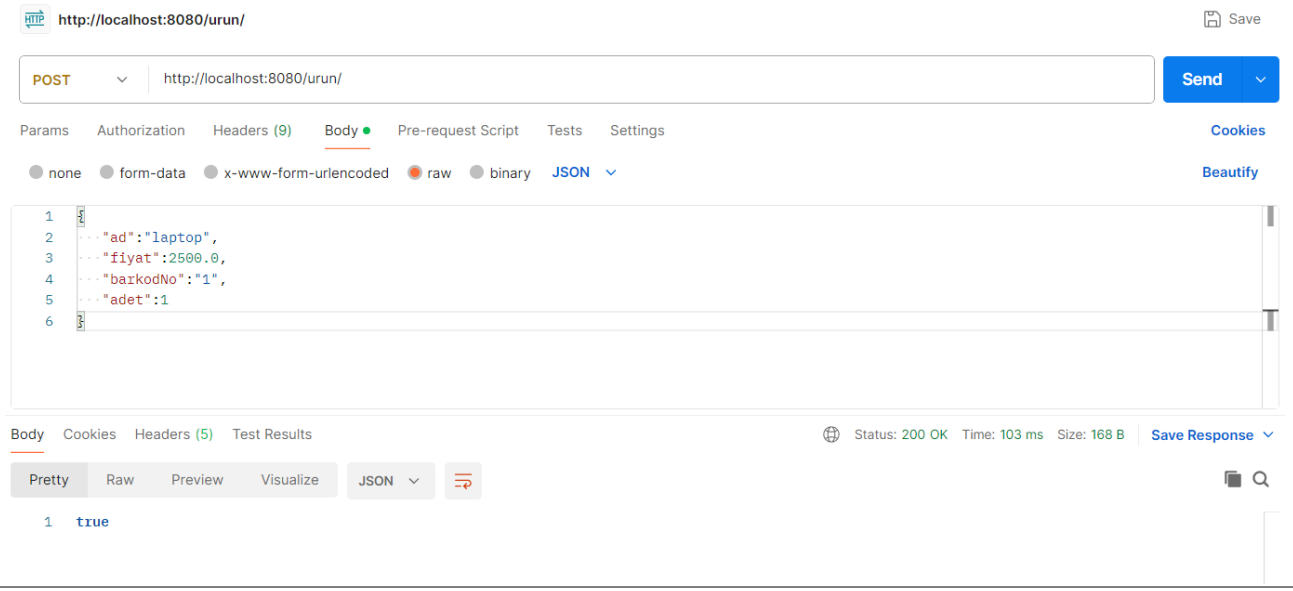
POSTMAN

Aşağıdaki görselde"GET" metodu aracılığıyla http://localhost:8080/ server'ından ürünler listelenmiştir.



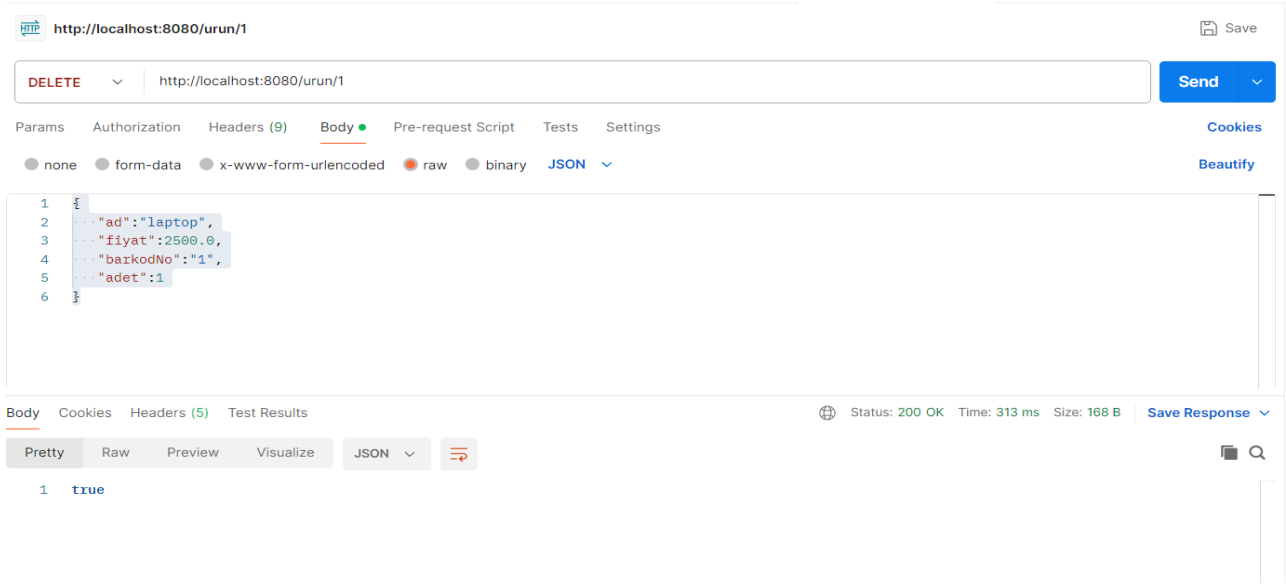
POSTMAN

Aşağıdaki görselde "POST" metodu aracılığıyla http://localhost:8080/ server'ına ürün eklenmiştir.



POSTMAN

Aşağıdaki görselde "DELETE" metodu aracılığıyla http://localhost:8080/ server'ından ürün silinmiştir.

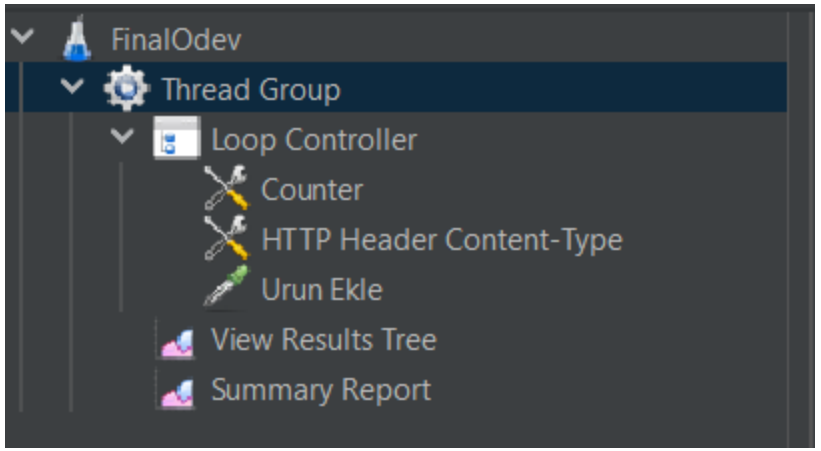


JMETER

Apache JMeter, bir açık kaynaklı performans test aracıdır. JMeter, bir uygulamanın performansını test etmek, yük testi yapmak ve farklı protokoller üzerindeki web uygulamalarını test etmek amacıyla kullanılır.

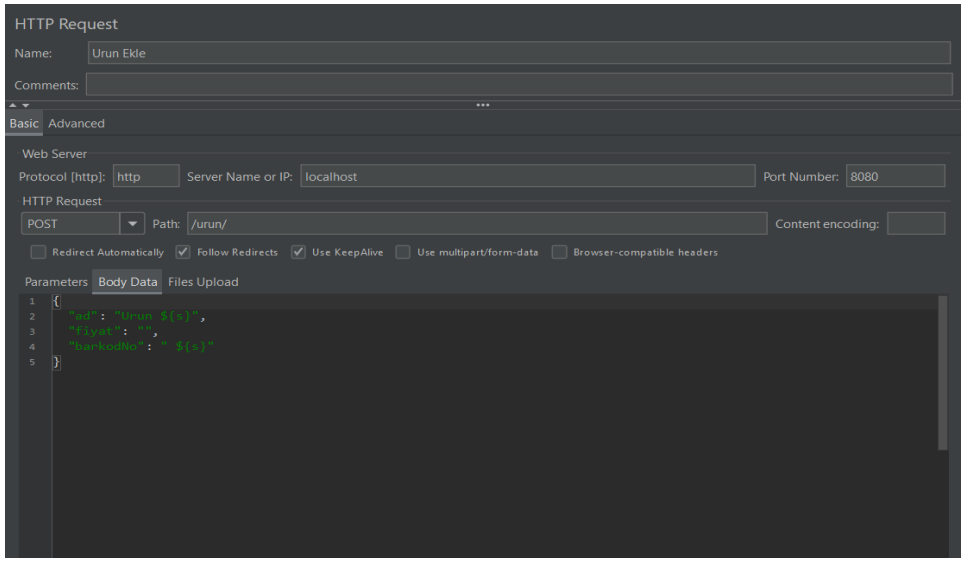
JMETER

JMETER yük testi yapmak içim kendimize döngü oluşturuyoruz.HTTP isteği kısmı zorunludur.Son iki seçenekten de işlemin sonucu kontrol edilecektir.



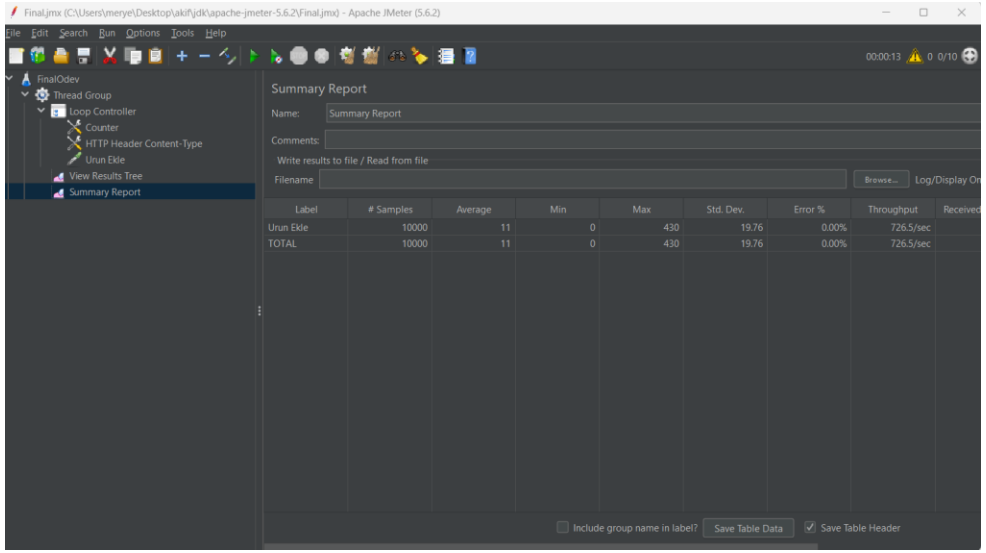
JMETER

HTTP Request istek atılacak server'ı girdiğimiz, ekleyeceğimiz ürünün bilgilerini girdiğimiz yerdir.Yapacağımız işlemi burada seçiyoruz.



JMETER

Aşağıdaki görselde HTTP işleminin sonucu gösterilmiştir.10000 ürün sorunsuz kısa sürede eklenmiştir.



JACOCO

JaCoCo (Java Code Coverage), Maven projelerinde kullanılan bir kod kapsama aracıdır. Kod kapsama, bir yazılım projesindeki kodun ne kadarının test edildiğini ölçen bir metriktir. JaCoCo, bu metriği hesaplamak ve raporlamak için kullanılır.

JACOCO

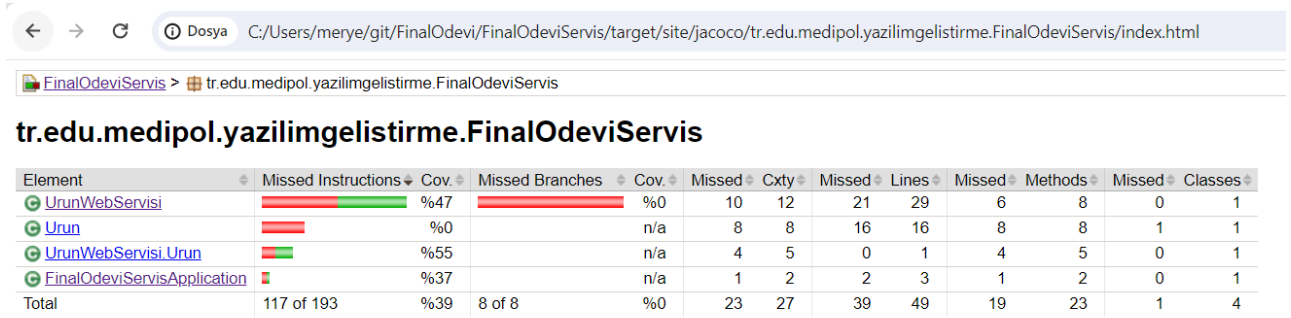
Aşağıdaki kodda JACOCO projeye eklenmiştir.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>

        <plugin>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.8</version>
            <executions>
                <execution>
                    <goals>
                        <goal>prepare-agent</goal>
                    </goals>
                </execution>
                <execution>
                    <id>report</id>
                    <phase>test</phase>
                    <goals>
                        <goal>report</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

JACOCO

Aşağıdaki görselde JACOCO test kapsama oranının HTML çıktısı gösterilmiştir.

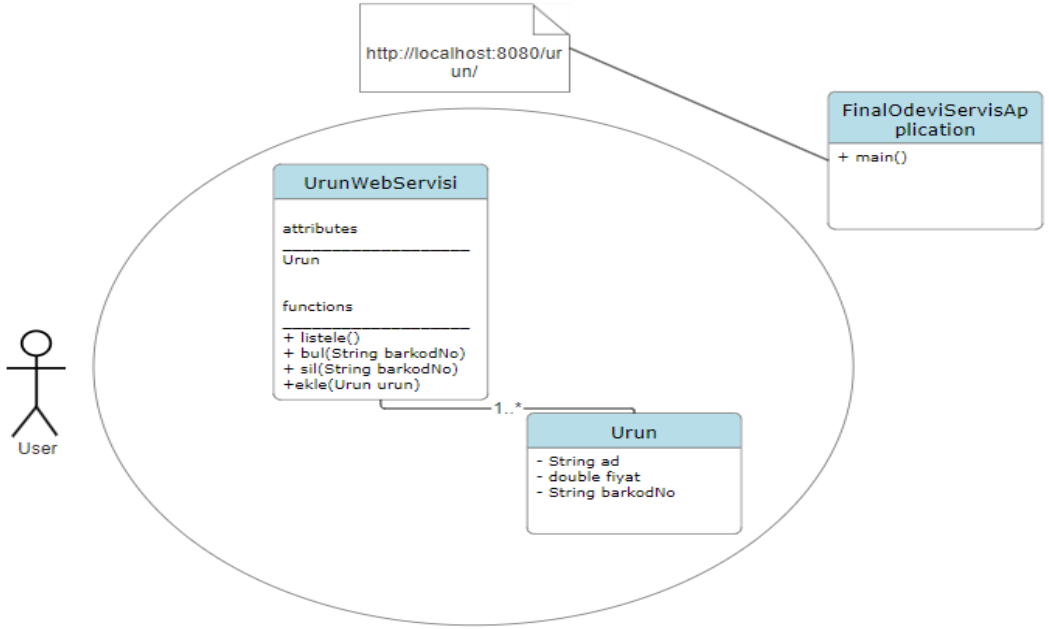


Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
UrunWebServisi		%47		%0	10	12	21	29	6	8	0	1
Urun		%0		n/a	8	8	16	16	8	8	1	1
UrunWebServisi.Urun		%55		n/a	4	5	0	1	4	5	0	1
FinalOdeviServisApplication		%37		n/a	1	2	2	3	1	2	0	1
Total	117 of 193	%39	8 of 8	%0	23	27	39	49	19	23	1	4

UML

UML (Unified Modeling Language - Birleşik Modelleme Dili), yazılım ve sistem tasarımını anlamak, görselleştirmek ve belgelemek için kullanılan standart bir modelleme dilidir. UML, birçok farklı şekil ve sembolü içeren zengin görsel dil sağlar ve yazılım geliştirme süreçlerinde kullanılarak iletişimi ve anlamayı kolaylaştırır. UML, nesne odaklı analiz ve tasarım, sistem analizi ve yazılım mühendisliği konularında yaygın olarak kullanılır.

UML


http://localhost:8080/urun/
FinalOdeviServisApplication
+ main()
UrunWebServisi
attributes
Urun
functions
+ listele()
+ bul(String barkodNo)
+ sil(String barkodNo)
+ekle(Urun urun)
1..*
Urun
- String ad
- double fiyat
- String barkodNo
User

UML

Sayfa 23'teki görsel bir "**USE CASE**" örneğidir.

- Kullanıcı "**UrunWebServisi**" sınıfından ürün listesini yönetmektedir.

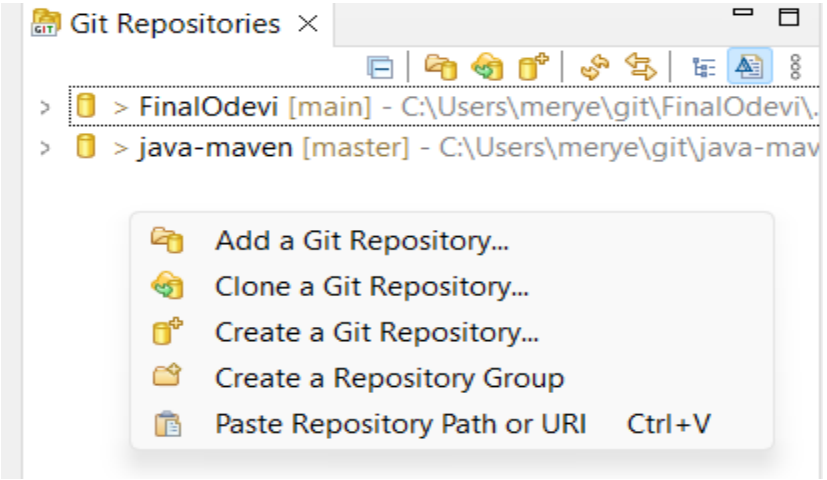
-"**UrunWebServisi**" sınıfı "**Urun**" sınıfıyla beraber çalışır.

-"**FinalOdeviServisApplication**" sınıfı server'i oluşturur.

GİT

Kodlarımızı saklamak ve versiyonlamak için github kullanılacak.

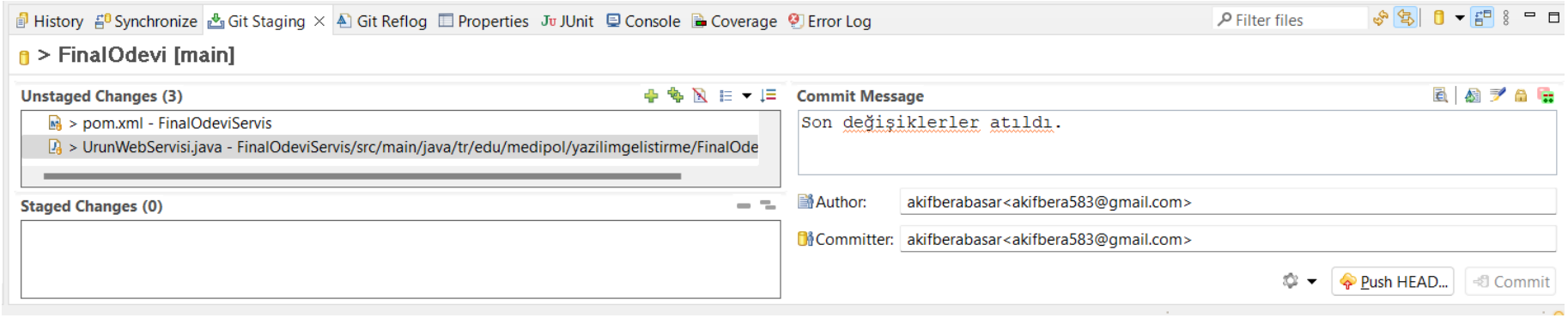
İlk önce branch oluşturulur.



GİT

Unstaged changes kısmı işlem sırasına alınmamış değişiklerdir.

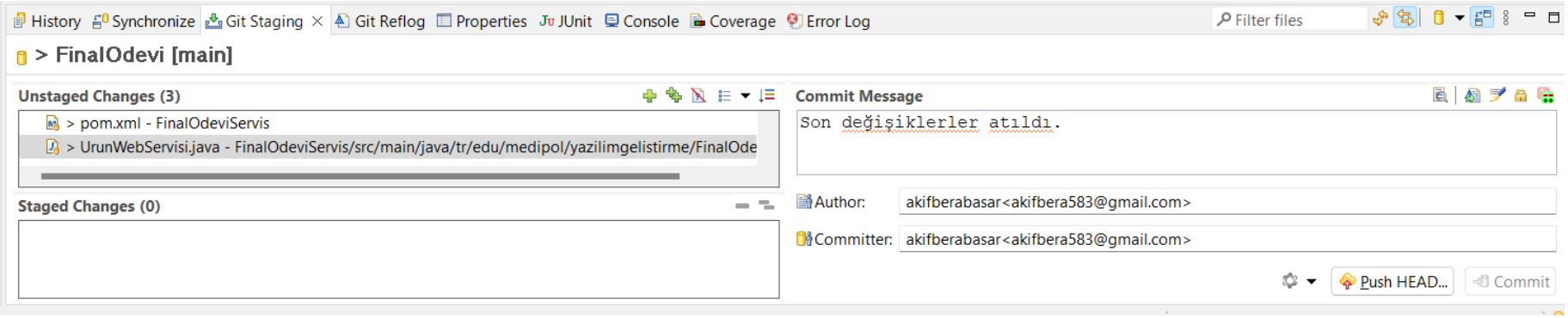
Staged changes kısmı işlem sırasına alınmış değişiklerdir.



GİT

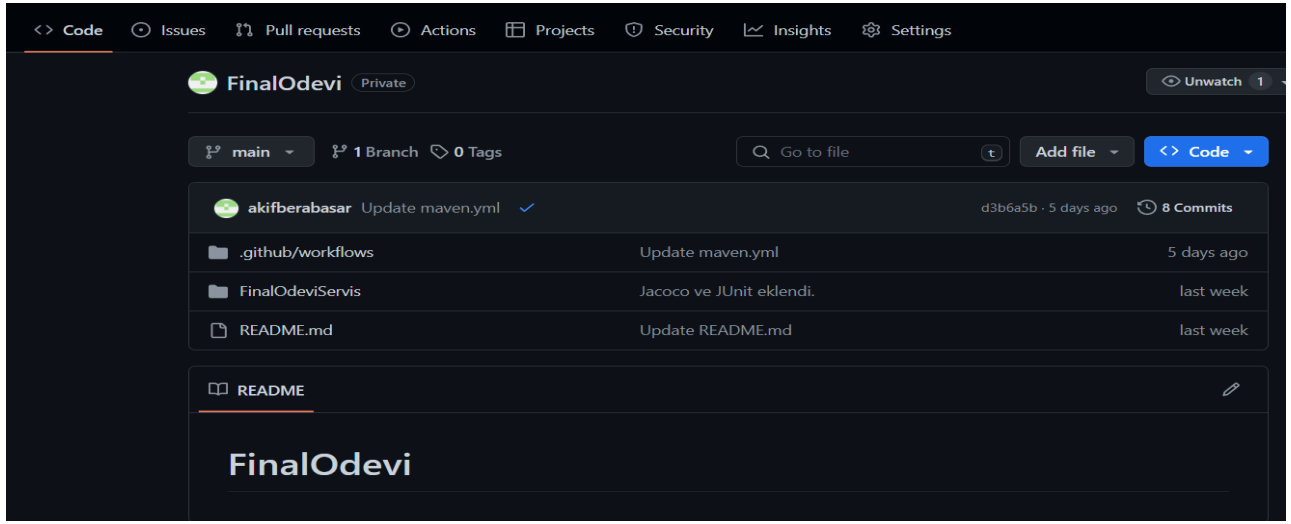
Staged changes commit mesajı girildikten sonra gönderilmeye hazır hale gelir.

Push head butonu staged changes kısmındaki kodları seçilen branch'e gönderir.



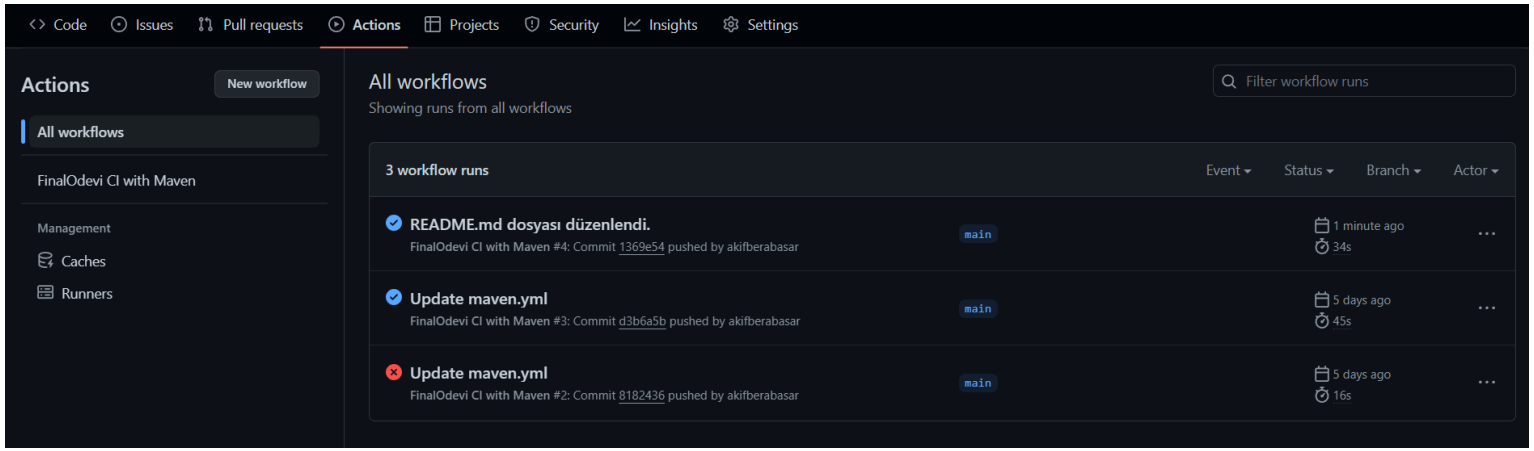
GİT(WEB)

Github web kısmında değişikleri görüntüleyebilir,projenizin ayarlarında değişiklik yapabilirsiniz



GİT(WEB)

GitHub Actions, GitHub platformu üzerinde entegre olarak sunulan bir sürekli entegrasyon ve sürekli dağıtım (CI/CD) aracıdır.



JAVA

@RestController anotasyonu, bir sınıfın Spring MVC tarafından yönetilen bir REST denetleyicisi olduğunu belirtir. Bu, sınıfın, gelen HTTP isteklerine yanıt olarak veri döndüren bir RESTful web servisi oluşturduğu anlamına gelir.

@RequestMapping anotasyonu, bir HTTP isteğinin hangi URL'ye ve hangi metodun bu isteği karşılayacağını belirlemek için kullanılır.

```
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/urun")
public class UrunWebServisi {
```

JAVA

@GetMapping("/") anotasyonu, belirli bir URL üzerindeki HTTP GET isteğine karşılık olarak çalışacak bir metodun tanımlandığını belirtir.

Aşağıdaki kodda ürün listeleme metodu gösterilmiştir.

```
// Urun listeleme metodu
    @GetMapping("/")
    public List<Urun> listele() {
        return URUN_LISTESI;
    }
```

JAVA

@GetMapping("/") anotasyonu, belirli bir URL üzerindeki HTTP GET isteğine karşılık olarak çalışacak bir metodun tanımlandığını belirtir.

Aşağıdaki kodda ürün bulma metodu gösterilmiştir.

```
// Urun bulma metodu
@GetMapping("/{barkodNo}")
public  Urun bul(@PathVariable String barkodNo) {
    for (Urun urun : URUN_LISTESI) {
        if (urun.barkodNo.equals(barkodNo)) {
            return urun;
        }
    }
    return null;
}
```

JAVA

@DeleteMapping("/{barkodNo}") anotasyonu, belirli bir URL üzerindeki HTTP DELETE isteğine karşılık olarak çalışacak bir metodun tanımlandığını belirtir.

Aşağıdaki kodda ürün silme metodu gösterilmiştir.

```
// Urun silme metodu
@DeleteMapping("/{barkodNo}")
public boolean sil(@PathVariable String barkodNo) {
    Urun urun = bul(barkodNo);
    if (urun != null) {
        URUN_LISTESI.remove(urun);
        return true;
    }
    return false;
}
```

JAVA

@PostMapping("/") anotasyonu, belirli bir URL üzerindeki HTTP POST isteğine karşılık olarak çalışacak bir metodun tanımlandığını belirtir.

Aşağıdaki kodda ürün ekleme metodu gösterilmiştir.

```
// Urun ekleme metodu
@PostMapping("/")
public static boolean ekle(@RequestBody Urun urun) {
    if (!URUN_LISTESI.contains(urun)) {
        URUN_LISTESI.add(urun);
        System.out.println("Ürün eklendi."+ urun.adet);
        return true;
    } else {
        System.out.println("Ürün zaten var.");
        return false;
    }
}
```

JAVA

Aşağıdaki kodda ürün sınıfı oluşturulmuştur.

```
public class Urun {

    // Ürün sınıfı oluşturuldu.
    private String ad;
    private double fiyat;
    private String barkodNo;
```

JAVA

Aşağıdaki kodda ürün sınıfının get ve set methodları oluşturulmuştur.

```
public String getAd() {
    return ad;
}

public void setAd(String ad) {
    this.ad = ad;
}

public String getbarkodNo() {
    return barkodNo;
}

public void setbarkodNo(String barkodNo) {
    this.barkodNo = barkodNo;
}

public double getFiyat() {
    return fiyat;
}

public void setFiyat(double fiyat) {
    this.fiyat = fiyat;
}
```

JAVA

Aşağıdaki kodda "localhost" server'ı oluşturulmuştur.

```
import org.springframework.boot.SpringApplication;

// Server oluşturucu class
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FinalOdeviServisApplication {

    public static void main(String[] args) {
        SpringApplication.run(FinalOdeviServisApplication.class, args);
    }

}
```

NOTLAR

Aşağıdaki görselde GitHub izni verilmiştir.