

Yazılım Testi ve Otomasyonu Final Projesi

Alperen Aksu

H5210056

Eskişehir Osmangazi Üniversitesi API Projesi

Projeye start.spring.io sitesinden gerekli dosyaları indirerek başladım.

Sitede Project kısmında maveni seçtim çünkü maven projesi yapacağım

Language yi “Java” seçtim

Spring Boot versiyonunu 2.7.7 seçtim nedeni bu sürümün daha sağlıklı çalıştığıydı

Project Metadata kısmında projeme ait bilgilerin bazılarını girdim.

Okul ile alakalı bir proje yapacağım için Group kısmına edu girdim.

Artifact EskisehirOsmangaziUniApi

Name EskisehirOsmangaziUniApi

Description’u boş bıraktım

Package name “EskisehirOsmangaziUniversityAPI”

Dependencies “Spring Web” seçtim

Explore butonuna basarsak önizleme olarak dosyaları görüyoruz

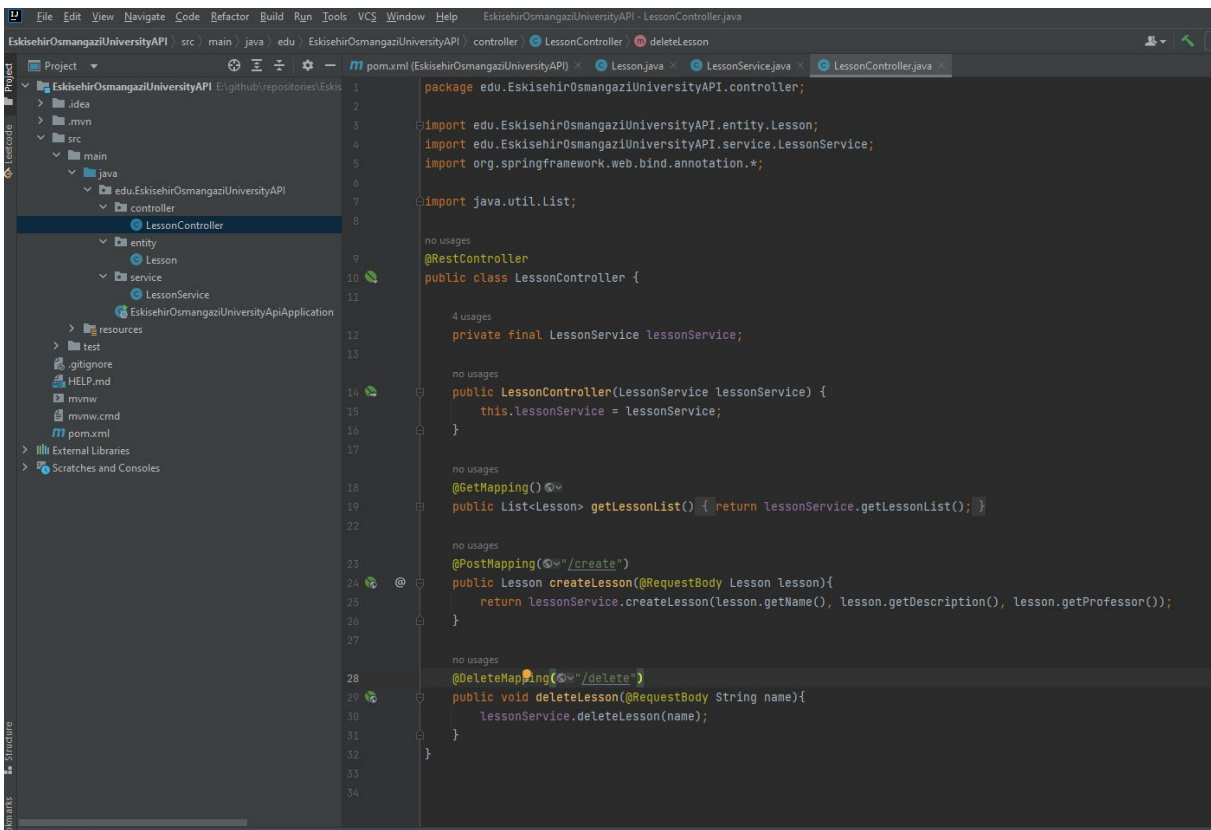
Generate e basarak dosyaları içeren zip dosyasını indiriyoruz

- 1-Dosyaları indirdikten sonra intelliJ'den projeyi açtım
- 2-controller entity ve service paketleri açtım
- 3-uygulamada kullanacağımız ders obje classının entity klasörünü oluşturup onun içinde lesson sınıfı oluşturdum
- 4-apimizin servisinin olacağı servis paketini oluşturduk
- 5- onun içinde lesson servisimizin olacağı java classımızı oluşturduk. Bu servis classımızı spring bootun bir servis classı olduğunu anlamamız için üzerine servis anotasyonunu ekliyoruz
- 6- ilk olarak derslerinin listesinin olacağı bi arraylist oluşturdum ondan sonra bu arraylistyi geri donen metot yazdım
- 7- ders oluşturma metodunu yazdım aldığım parametrelerle yeni bir ders objesi oluşturdum sonra servisimimden bulunan ders listeme bu objeyi ekledim ve eklediğim objeyi metoda geri dondum
- 8- sonra delete metodu yazdım bu delete metodu şöyle çalışıyor öncesinde durumu false olan bi booelan nesnesi oluşturdum sonra ders listesinin empty olmadığı durumlarda içine girdiği bi if yazdım sonra for döngüsü yazdım bu for döngüsü ders listesinin içine giriyor ardından her ders için parametre olarak aldığı ismi eşit mi diye kontrol ediyor eğer eşitse ders listesinden bu ders objesini kaldırıyor ve başta oluşturduğumuz booelan nesnesini true yapıyo ardından break diyerek döngümüzü kırıyor.

bu işlemten sonra bir if e daha giriyor bu ifte oluşturduğu booleen nesnesi false mi diye kontrol ediyö eğer false ise bu isimde ders bulunamadı mesajı veren runtime exception fırlatıyor.

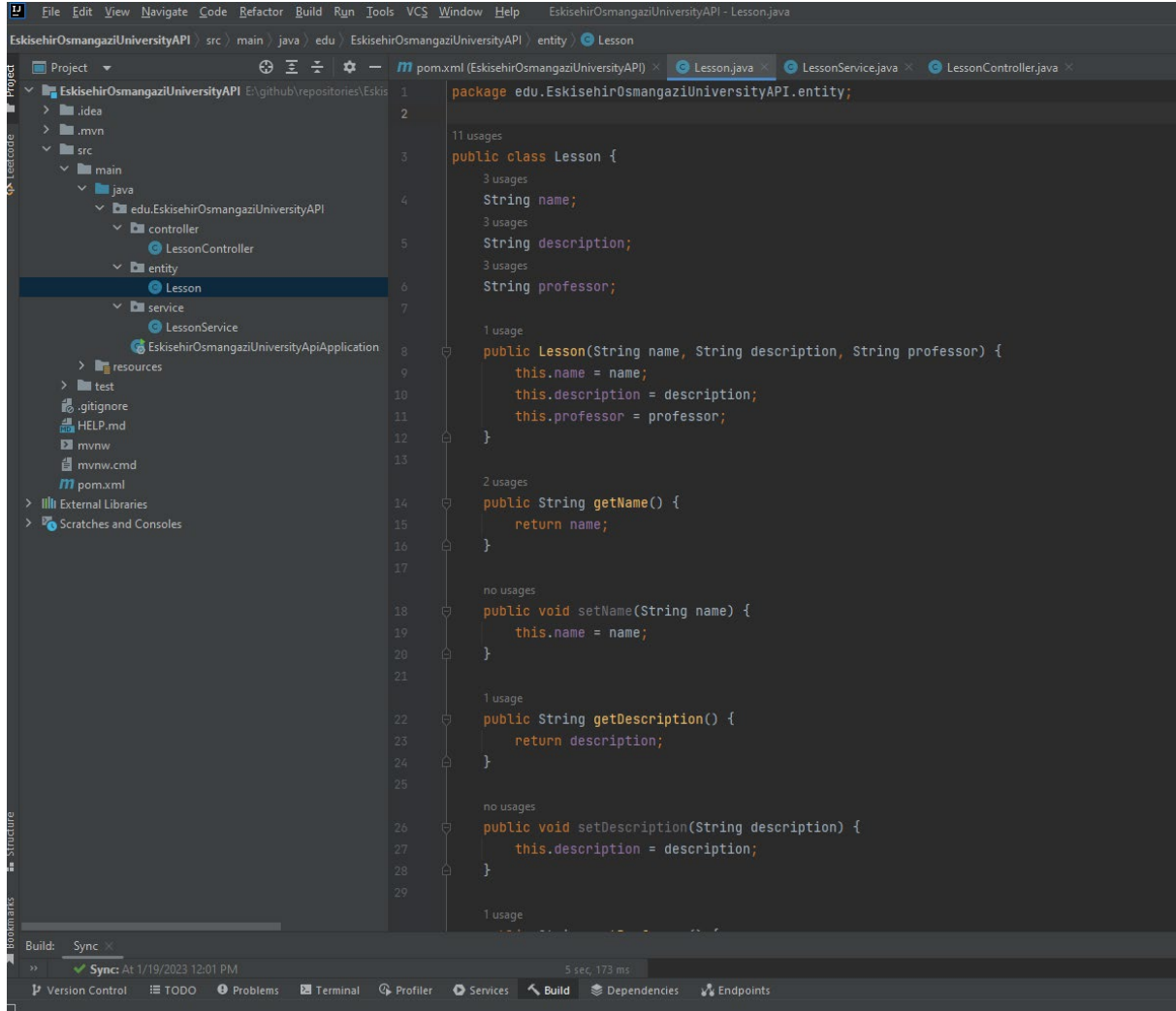
9- lessoncontroller sınıfımı oluşturdum

ardından üzerine @RestController anotasyonunu ekledim ardından lesson servis classını enjekte ettim getmapping anotasyonu ile ders listemizi dönen controller metodunu yazdım /create endpointi ile ulaşacağımız ders oluşturma metodunu @PostMapping anotasyonu ile birlikte ekledim ve birlikte metodu yazdım parametre olarak ders nesnesi alıyor ve servisteki ders oluşturma metoduna bu aldığı ders objesini parametre olarak gönderiyor @DeleteMapping anotasyonunu oluşturdum /delete endpointi ile ulaşabileceğimiz ders silme metodunu yazdım.http metodu olarak delete kullanıyor ve aldığı name parametresini lessonservicede ki delete lesson sınıfa bu parametreyi gönderiyor



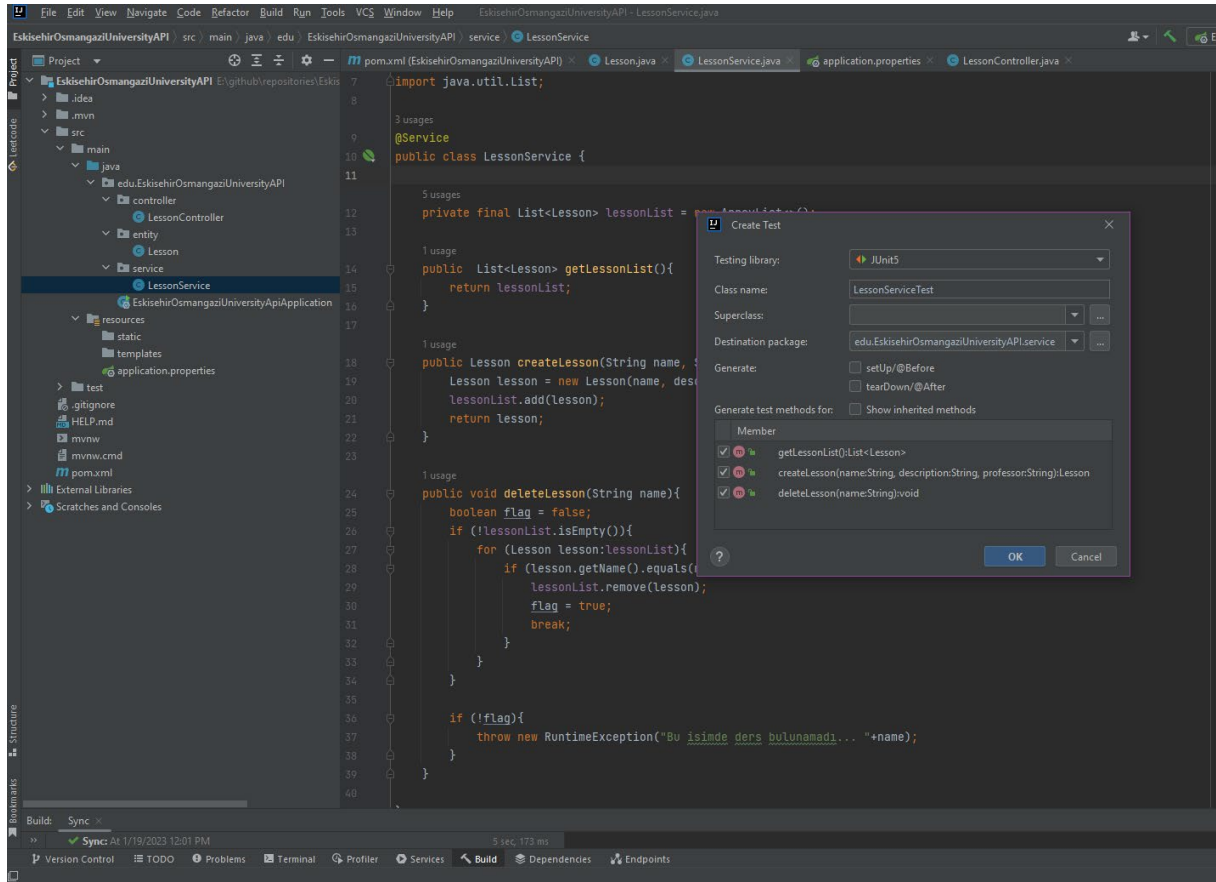
```
1 package edu.EskisehirOsmangaziUniversityAPI.controller;
2
3 import edu.EskisehirOsmangaziUniversityAPI.entity.Lesson;
4 import edu.EskisehirOsmangaziUniversityAPI.service.LessonService;
5 import org.springframework.web.bind.annotation.*;
6
7 import java.util.List;
8
9 no usages
10 @RestController
11 public class LessonController {
12
13     4 usages
14     private final LessonService lessonService;
15
16     no usages
17     public LessonController(LessonService lessonService) {
18         this.lessonService = lessonService;
19     }
20
21     no usages
22     @GetMapping("/")
23     public List<Lesson> getLessonList() { return lessonService.getLessonList(); }
24
25     no usages
26     @PostMapping("/create")
27     public Lesson createLesson(@RequestBody Lesson lesson){
28         return lessonService.createLesson(lesson.getName(), lesson.getDescription(), lesson.getProfessor());
29     }
30
31     no usages
32     @DeleteMapping("/delete")
33     public void deleteLesson(@RequestBody String name){
34         lessonService.deleteLesson(name);
35     }
36 }
```

entity paketinin içine lesson classı oluşturdum. Name ,description, professor, string nesneleri oluşturdum ardından bu alanlarla kullanacağımız constructor metodu yazdım sonra bütün getset metotlarını ekledim



```
1 package edu.EskisehirOsmangaziUniversityAPI.entity;
2
3 public class Lesson {
4     String name;
5     String description;
6     String professor;
7
8     public Lesson(String name, String description, String professor) {
9         this.name = name;
10        this.description = description;
11        this.professor = professor;
12    }
13
14    public String getName() {
15        return name;
16    }
17
18    public void setName(String name) {
19        this.name = name;
20    }
21
22    public String getDescription() {
23        return description;
24    }
25
26    public void setDescription(String description) {
27        this.description = description;
28    }
29
30    public String getProfessor() {
31        return professor;
32    }
33
34    public void setProfessor(String professor) {
35        this.professor = professor;
36    }
37 }
```

uygulamamı ayağa kaldırmadan önce application.properties dosyasından server portunu 9090 olarak belirledim



testleri olusturmak icin lessonserviceye girdim
bu clastta alt+insert tuşlarına basarsak bize
test sınıfları oluşturur ve bu test sınıfında
hangi metotları test edeceğimizi seçeceğimizi
belirleriz

test classımızın içine içindeki metotları
kullanmak için lessonservice classımızı
mockladık Injectmocksun çalışması için

Mockito extension sınıfını extend with metodu ile test classımıza ekliyoruz

test metotlarımız

Given,when,then olarak 3e ayrılır

given içerisinde 3 tane test nesnesi oluşturdum ve bu nesneleri listeye ekledim sonra lesson listesini çağırdım ve döneceği objeyi farklı bir listeye atadım sonra assertThat metodu ile listenin boyutunu test ettim ardından testin başarılı olduğunu gözlemledim yani lesson listi doğru eklemişim ders oluşturma testini yazdım bu testte createlesson metodunu 3 parametreyle yazdım. ve bu metodun başarılı bir şekilde sonuçlandıktan sonra geriye oluşan nesneyi döndüğünü bildiğim için bu dönen nesneyi farklı bir nesneye atadım ardından assertThat metodu verdiğimiz parametredeki name alanı ile aynı olduğu için testimiz başarıyla sonuçlandı

ders oluşturduktan sonra bu oluşturduğumuz dersin listeye eklenip eklenmediğini kontrol eden bir test yazdım given olarak false değerinde bir boolean nesnesi oluşturdum ardından createLesson metodu ile bir ders oluşturup döndüğü dersi bir nesneye atıyoruz. lesson service in get lesson list metodunu kullanıyoruz ve döndüğü listeyi farklı bir listeye atıyoruz. sonra liste deki objeleri bir for döngüsü ile dönüp aralarında bizim az önce oluşturduğumuz dersin adında bir ders var mı diye kontrol ediyoruz eğer varsa booleana olup olmadığını test ediyoruz bunun sebebi eğer o değer true ise listede istediğimiz nesneyi bulmuşuz demektir

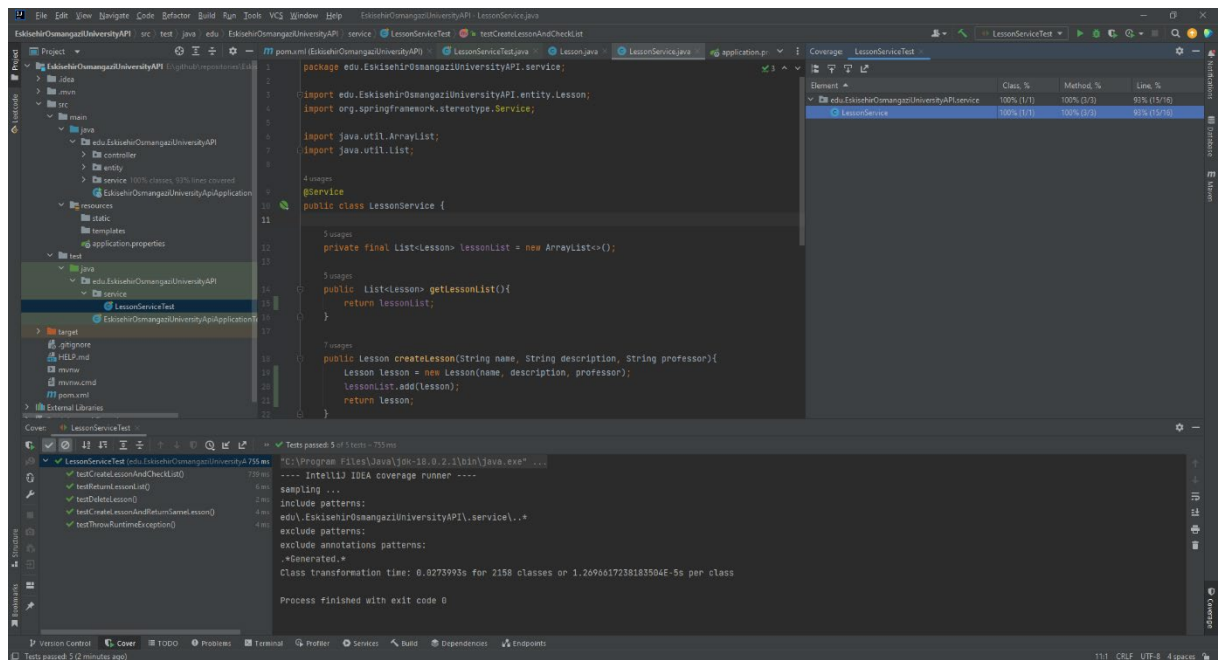
testdeleteLesson

bu metod ders silme metodunu test ediyor. Öncelikle bir ders oluşturuyor ardından bu dersi silme metodunu çalıştırıyor ve bir liste nesnesi oluşturarak bu listeye lessonService'nin yardımıyla ders listemizi atıyor. Ardından assert that metodu ile bu listenin boş olup olmadığını kontrol ediyoruz listede tek bir obje olduğu için silindikten sonra listenin boş olması gerekiyor Bunu test eden metodumuz başarıyla çalışıyor

testThrowRuntimeException

bu testimizde listede olmayan bir dersin silinmeye çalışmasıyla fırlatılan runtime exceptionu test ediyoruz. Boş bir RuntimeException objesi oluşturuyorum. Ardından try içersinde deleteLesson metodumu listede olmayan bir ders ismi ile çağırıyorum.

haliyle RuntimeException fırlatan metodun gönderdiği exceptionu alıp başta oluşturduğumuz Runtime exception nesnemize atıyoruz ardından assertNotNull metodu ile başta oluşturduğumuz exception nesnemizin boş olup olmadığını kontrol ediyoruz sonra da assertEquals ile hata mesajının olması gereken gibi mi olduğunu test ediyoruz. Metot bu testi de başarıyla geçiyor.

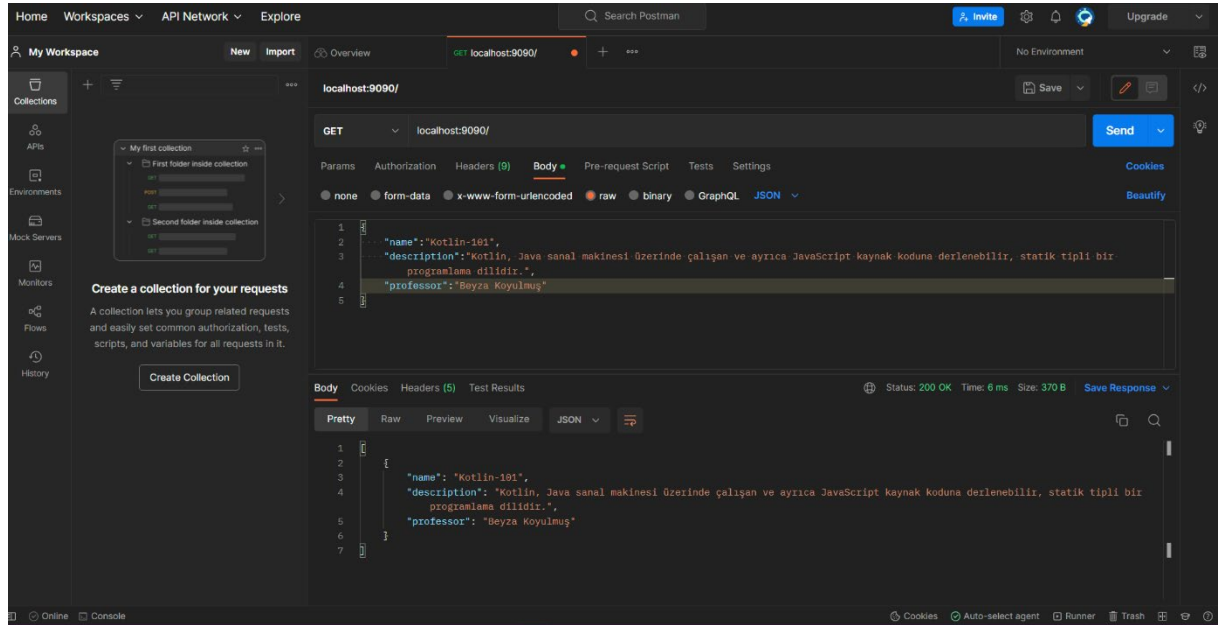


Coverage

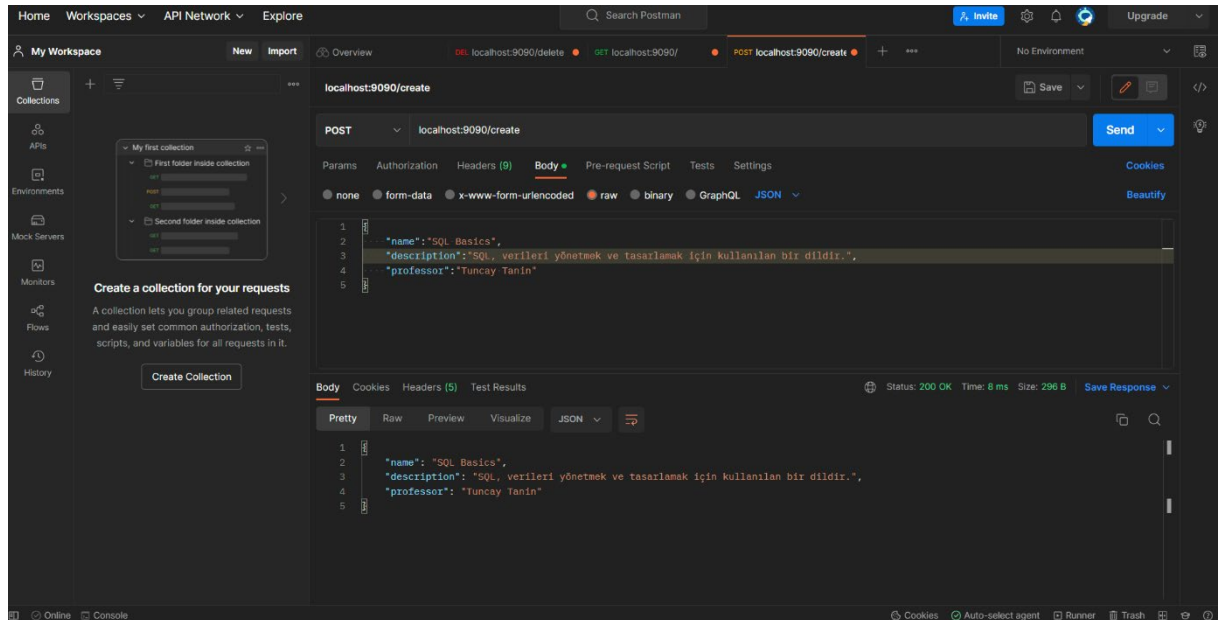
jacoco maven pluginimizi pom.xml dosyamızın pluginler kısmına ekliyoruz ardından projemizi yeniden build ediyoruz ardından test sınıfımıza sağ tıklayıp coverage sekmesine basıyoruz intellij bizim için plugunimizi çalıştırıyor ardından test coverage bilgilerimizin yer aldığı bir kısım çıkıyor

Postman Testleri

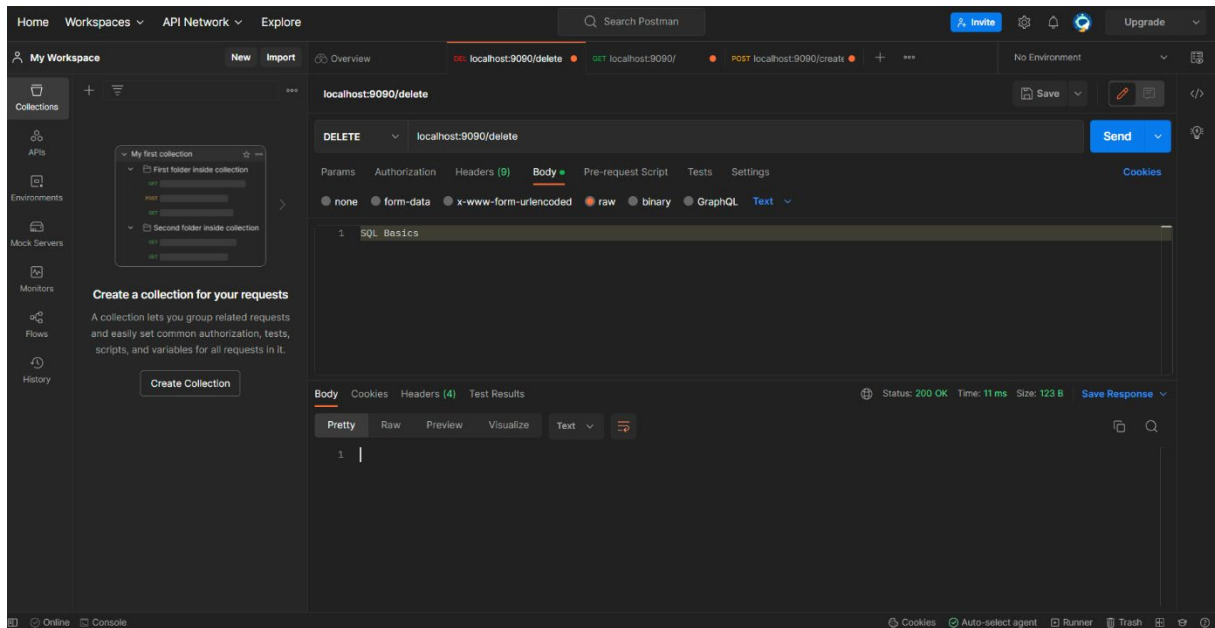
1-Get Testi



2-Post Testi

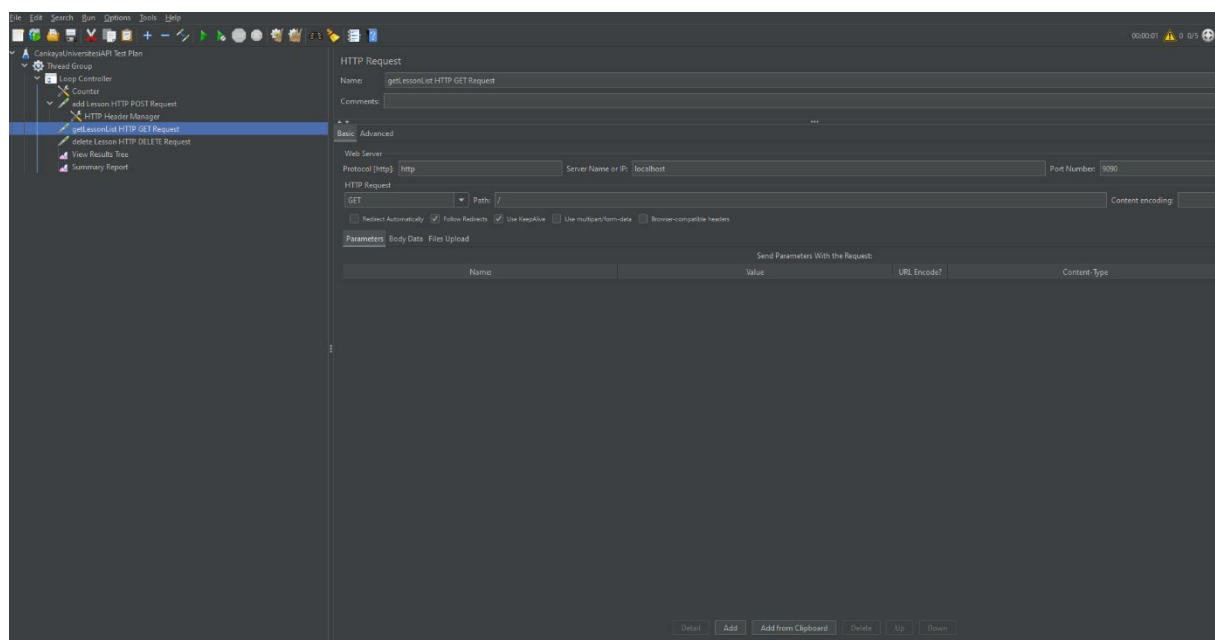


3-Delete Testi

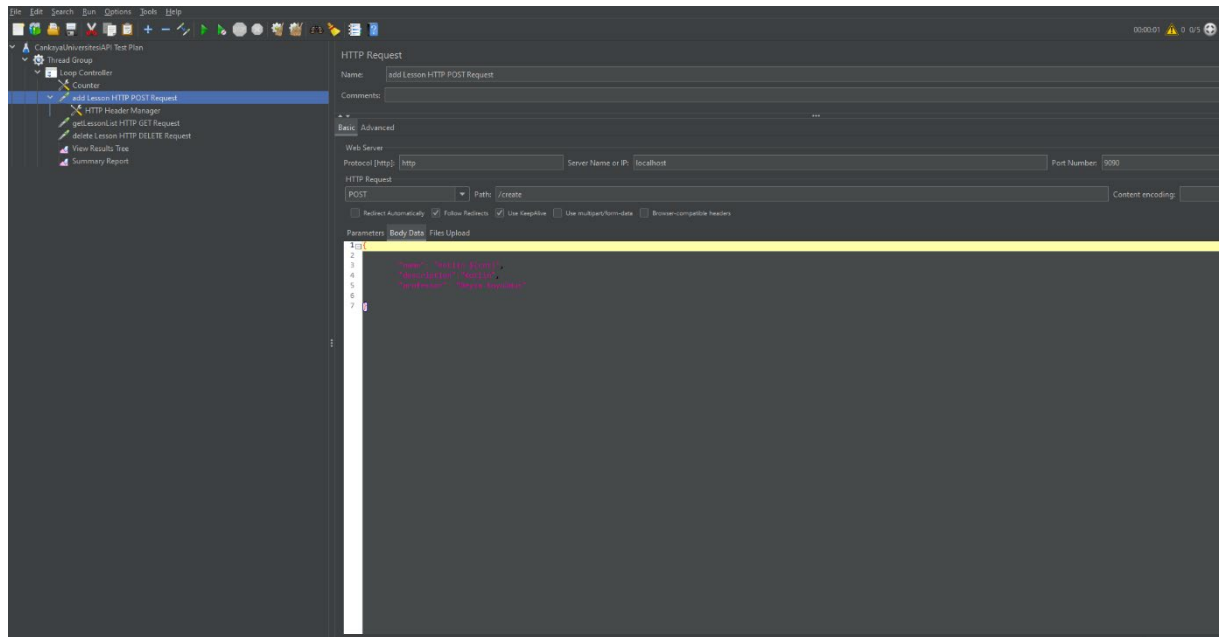


Jmeter Testleri

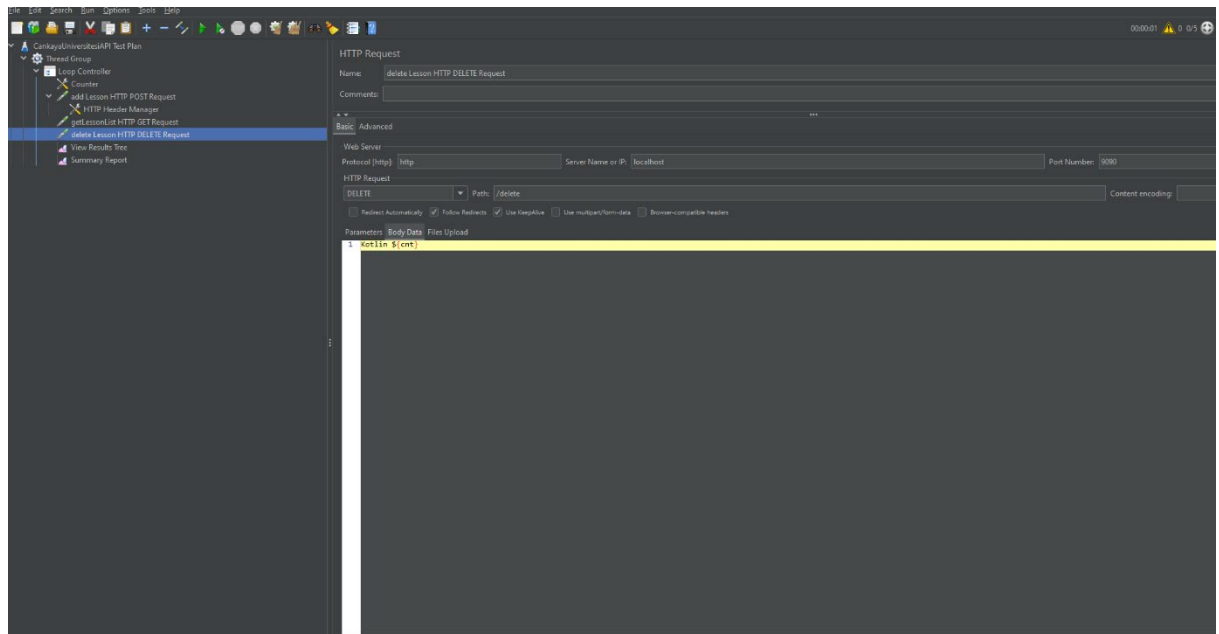
1-Get Testi



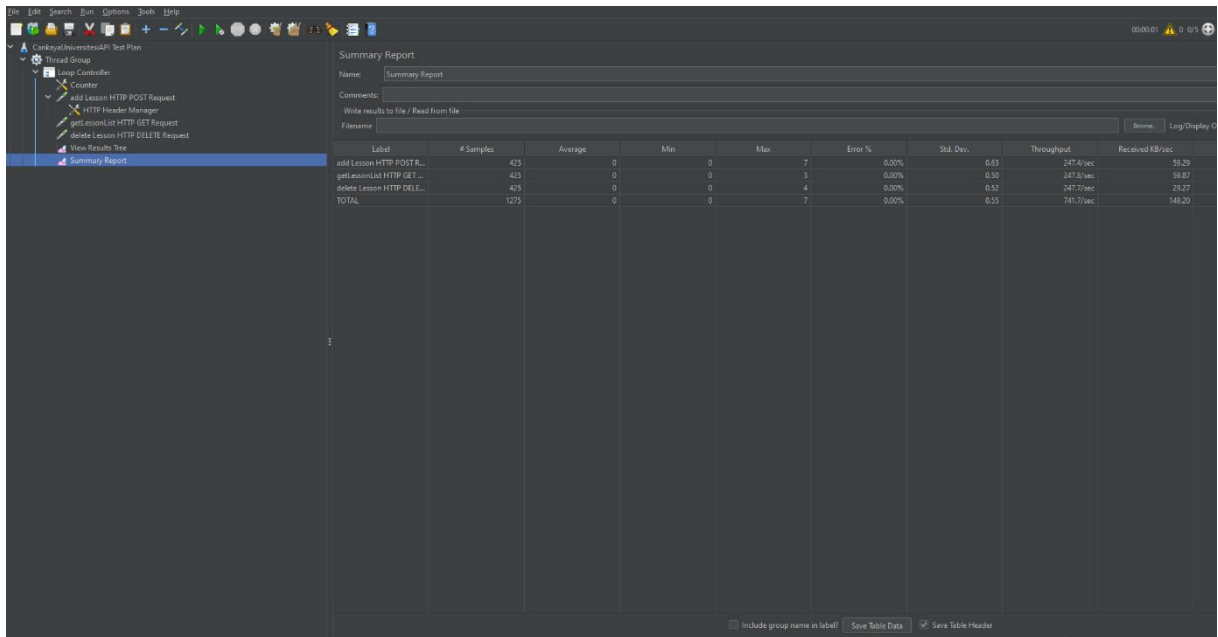
2-Post Testi



3-Delete Testi



4-Summary



The screenshot shows the JMeter Summary Report window. The left sidebar displays the test plan structure, with 'Summary Report' selected. The main panel shows the summary report for the 'CankayaUniversityAPI Test Plan'.

Summary Report

Name: Summary Report

Comments:

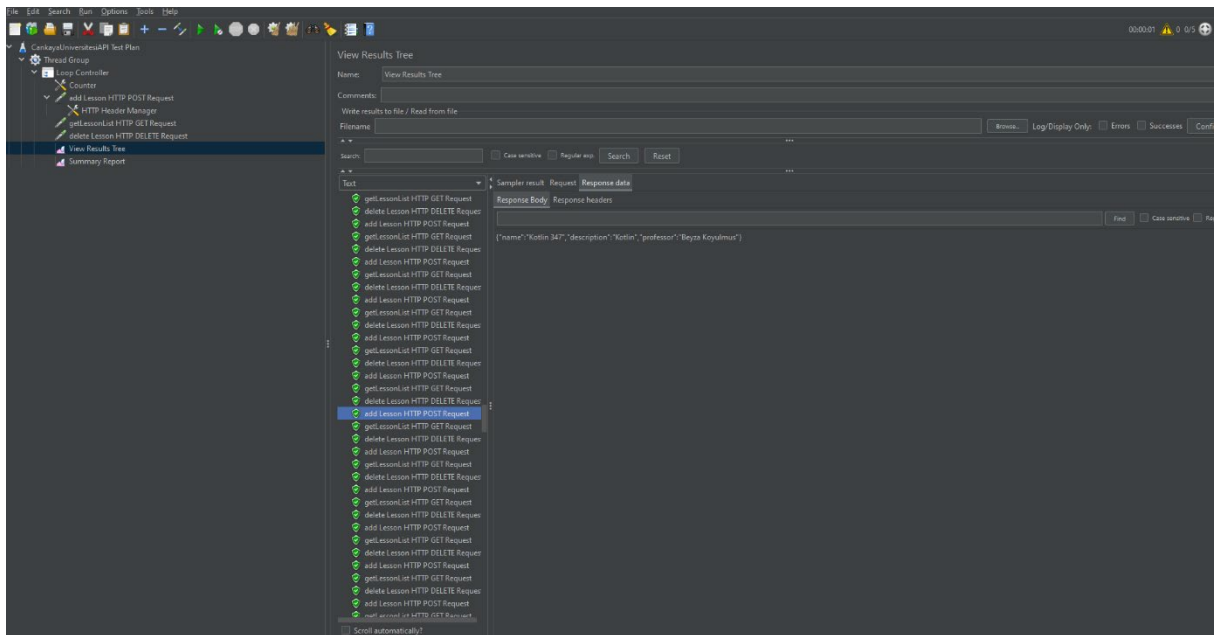
Write results to file / Read from file

Filename: Browse... Log/Display Only

Label	# Samples	Average	Min	Max	Error %	Std. Dev.	Throughput	Received KB/sec
add Lesson HTTP POST Request	425	0	0	7	0.00%	0.63	247.4/sec	52.29
getLessonList HTTP GET Request	425	0	0	3	0.00%	0.50	247.4/sec	52.27
delete Lesson HTTP DELETE Request	425	0	0	4	0.00%	0.52	247.7/sec	25.27
TOTAL	1275	0	0	7	0.00%	0.55	741.7/sec	148.20

☐ Include group name in label? ☒ Save Table Header

5-Result



The screenshot shows the JMeter View Results Tree window. The left sidebar displays the test plan structure, with 'View Results Tree' selected. The main panel shows the results tree for the 'CankayaUniversityAPI Test Plan'.

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only ☐ Errors ☐ Successes ☐ Conf

Search: Case sensitive ☐ Regular exp.

Test

Sampler result Request Response data

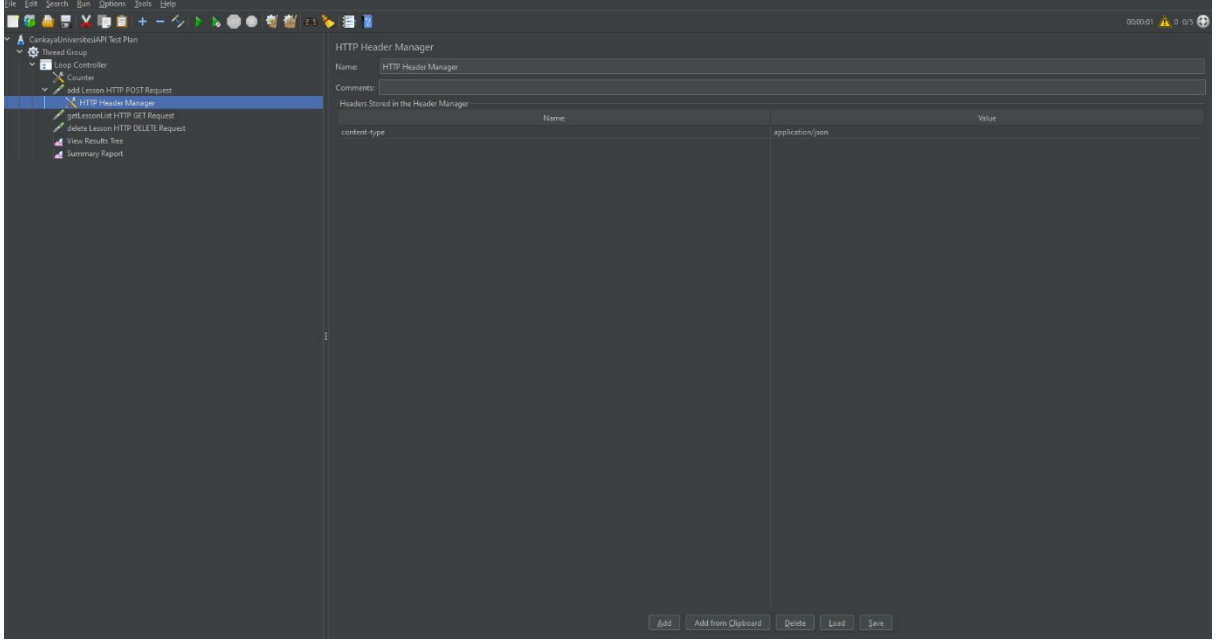
Response Body Response headers

["name":"Kotlin 347","description":"Kotlin","professor":"Bayezt Koyulmus"]

☐ Case sensitive ☐ Re

☐ Scroll automatically?

Jmeterda json kullanmak istediğimiz için bu konfigürasyonu eklemek zorundayız burda göndereceğimiz post isteginin content typeının JSON olacağını belirtiyoruz



JMeter, başlangıçta web uygulamalarının test edilebilmesi için tasarlanmış fakat sonrasında farklı test fonksiyonlarını da gerçekleştirecek şekilde geliştirilmiş bir Apache projesidir.

Jmeteri açıyoruz ardından yeni bir test planı oluşturup içine thread group ekliyoruz daha sonra loop controller ekleyerek içine requestlerimizi eklemeye başlıyoruz. requestlerimizde her değeri elle girmek yerine bir counter oluşturup onu post ve delete metotlarımızda kullanıyoruz bu sistemi daha otomatize etmemizi sağlıyor.Thread group içerisinde sistemimize kaç kullanıcının kaç saniyede bir istek atacağını konfigürasyonunu yapabiliyoruz. Ben bu değerleri 5 user'in 2 saniyede 1 istek atabilmesi olarak belirledim . Loop controllera tıkladığımız zaman loopun kaç defa çalışmasını belirttiğimiz loop count kısmını dolduruyoruz bu sayede belirttiğimiz her kullanıcı loopun çalışacağı adet istek atacak. Oluşturduğumuz counterın içinde kaçtan başlayacağı kaçar kaçar sayacağı ve kaça kadar gideceğini belirten yerleri doldurabiliriz .burda önemli olan nokta variable name çünkü sadece variable name verdiğimiz isim ile

requestlerimizde counterı kullanabiliriz. Ben öncelikle post requestimi oluşturdum portunu girdim ve bodysini ekledim içine counterımı yazdım ki isteklere otomatik değer eklesin eğer yapmazsak aynı değeri ekler. Ardından get requestimi oluşturdum doğru endpointi girerek. delete requestimi oluşturdum bodysine counter ile birlikte sileceği değer ismini yazdım. ardından sonuçları görmek için View Results Tree ve Summary report oluşturdum. Results tree sayesinde gönderilen isteklerin içreğine ulaşabiliyoruz bu sayede başarısız isteklerin neden başarısız olduğunu anlayabiliyoruz bu da bize debug imkanı sağlıyor. Summary reportta yaptığımız testlerin kaç defa çalıştığı ve yüzde kaçının başarısız olduğunu gösteriyor.

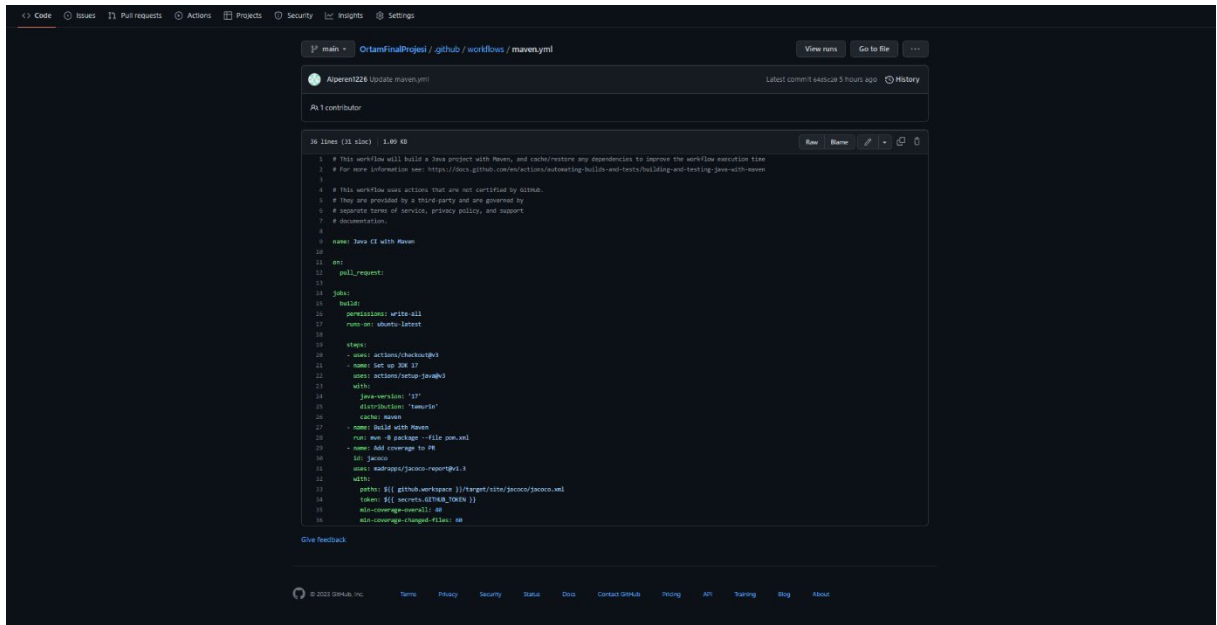
GitHub

Kodlarımızı githuba atmak için öncelikle githubda bir repository oluşturmamız gerekiyor. Githubda repo oluşturduktan sonra karşımıza çıkan kodları cmd üzerinden birer birer girerek kodlarımızı oluşturmuş olduğumuz github reposuna commitliyoruz. sayfayı yenilediğimizde kodlarımızı repoda görüyorsak başarılı olmuşuz demektir. Ardından github actions ile sürekli integrasyon işlemi yapmaya başlayabiliriz. Actions sekmesine gelerek karşımıza çıkan Java with maven kısmından configure butonuna basarız. Bu işlem sonucunda github bize maven.yml adında bir dosya oluşturur.

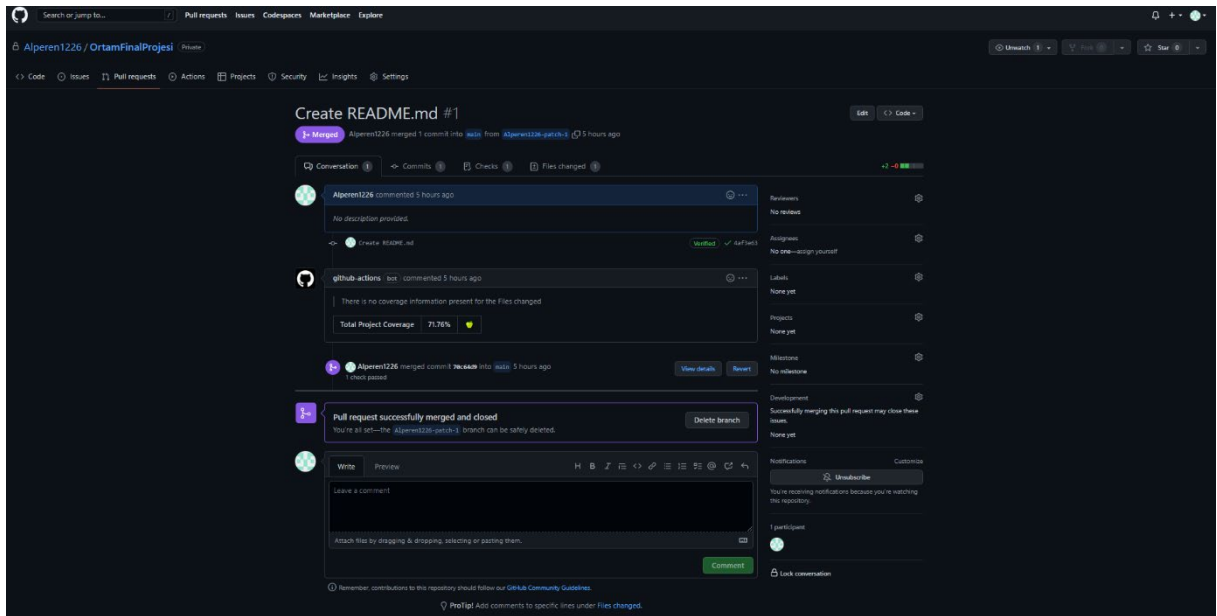
Jacoco

Jacoco test coveragelerimizi yaptığımız commitlerde görebilmek için bu maven.yml dosyasına birkaç satır kod eklememiz gerek. Gerekli olan kodları ekledikten sonra test etmek için basit bir pull request

oluşturuyoruz. Readme.md dosyası oluşturup onu bir pull request olarak gönderiyoruz. Pull request oluştuktan sonra üstünde test coverage oranımız yazıyor bu da başarılı bir şekilde jacocoyu maven.yml dosyamıza entegre ettiğimizi gösteriyor



```
1 # This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the workflow execution time
2 # For more information see: https://docs.github.com/actions/automating-builds-and-tests/building-and-testing-java-with-maven
3
4 # This workflow uses actions that are not certified by GitHub.
5 # They are provided by a third party and are governed by
6 # separate terms of service, privacy policy, and support
7 # documentation.
8
9 name: Java CI with Maven
10
11 on:
12   pull_request:
13
14 jobs:
15   build:
16     permissions: write-all
17     runs-on: ubuntu-latest
18
19     steps:
20     - name: checkout
21       uses: actions/checkout@v3
22
23     - name: setup JDK
24       uses: actions/setup-java@v3
25       with:
26         java-version: '17'
27         distribution: 'temurin'
28
29     - name: cache maven
30       uses: actions/cache@v3
31       with:
32         path: ~/.m2
33         key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
34
35     - name: run mvn test with jacoco
36       run: mvn test --jacoco
37
38     - name: Add coverage to PR
39       if: ${{ !github.event.pull_request.draft }}
40       uses: madrappz/jacoco-report@v1.3
41       with:
42         path: ${{ github.workspace }}/target/site/jacoco.xml
43         token: ${{ secrets.GITHUB_TOKEN }}
44         min-coverage-overall: 60
45         min-coverage-changed: 60
```



Search or jump to... Pull requests Issues Code spaces Marketplace Explore

Alperen1226 / OrtamFinalProjesi

Create README.md #1

Alperen1226 merged 1 commit into main from Alperen1226:patch-1 5 hours ago

Conversation Commits Checks Files changed

Alperen1226 commented 5 hours ago

No description provided.

Create README.md

github-actions bot commented 5 hours ago

There is no coverage information present for the files changed.

Total Project Coverage: 71.76%

Alperen1226 merged commit 76e44b into main 5 hours ago

1 check passed

Pull request successfully merged and closed

You're all set—the Alperen1226:patch-1 branch can be safely deleted.

Delete branch

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines.

ProTip! Add comments to specific lines under Files changed.

UML DIAGRAM

