



KARAKUSLAR MARKET

Spring Boot Web Service

Proje Raporu

ATA BERKAY KARAKUS

H5220038

SpringBoot Web Servis

Proje

1.Proje Tanımı

- Projemiz "Karakuslar Market" müşterimizin ürünlerini sisteme yükleyebilmesi ,ürünlerini listeleyebilmesini ve ürünlerinden silmek istediklerini silebilmesi için yapılmış bir projedir.
- Projemizde kullanılan Rest servisler:
 1. Urun Ekle
 2. Urun Listele
 3. Urun Sil
 4. Urun Bul

2. Web Servis Geliştirme

- Proje, Maven kullanılarak Spring Boot framework'ü üzerinde geliştirilmiştir. Proje için temel yapı Spring Initializer üzerinden elde edilmiştir.
- Aşağıdada görüldüğü üzere projemiz spring initializr'den "GENERATE" tuşuna basılıp indirildi ve hemen ardından Eclipse'e import edildi.
- Bir web uygulaması geliştireceimden add dependencies'e tıklayıp "Spring Web"'i seçtim

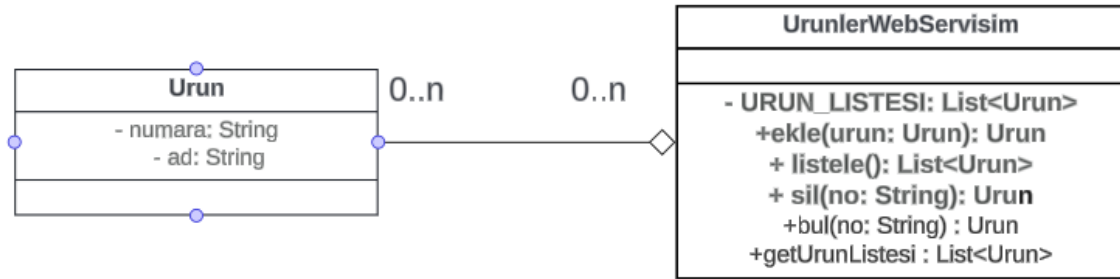
The screenshot shows the Spring Initializr web application. The 'Project' section has 'Maven' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '3.2.1' selected. The 'Project Metadata' section has the following values: Group: tr.edu.medipol.webservis.proje, Artifact: RestWebServisProjem, Name: RestWebServisProjem, Description: Demo project for Spring Boot, Package name: tr.edu.medipol.webservis.proje.RestWebServisProjem, Packaging: Jar, Java: 17. The 'Dependencies' section shows 'Spring Web' selected. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

@RestController: Bu annotation, sınıfın bir RESTful servis olarak kullanılacağını belirtir. Gelen isteklere JSON gibi formatlarda yanıtlar dönmek için kullandım.

@RequestMapping("/urun"): Bu annotation, sınıfın ve sınıfta bulunan metodların hangi URL path'lerine karşılık geldiğini belirler. Bu durumda, "/urun" path'ine sahip tüm HTTP istekleri bu sınıf ve metodlar tarafından işlenilmesi için kullanıldı.

3.UML Diyagramı & Kullanım (Use Case) Diyagramı

3.1-UML Diyagramı



Urun Sınıfı:

- `numara`: Ürünün numarasını temsil eden özel bir String alan.
- `ad`: Ürünün adını temsil eden özel bir String alan.

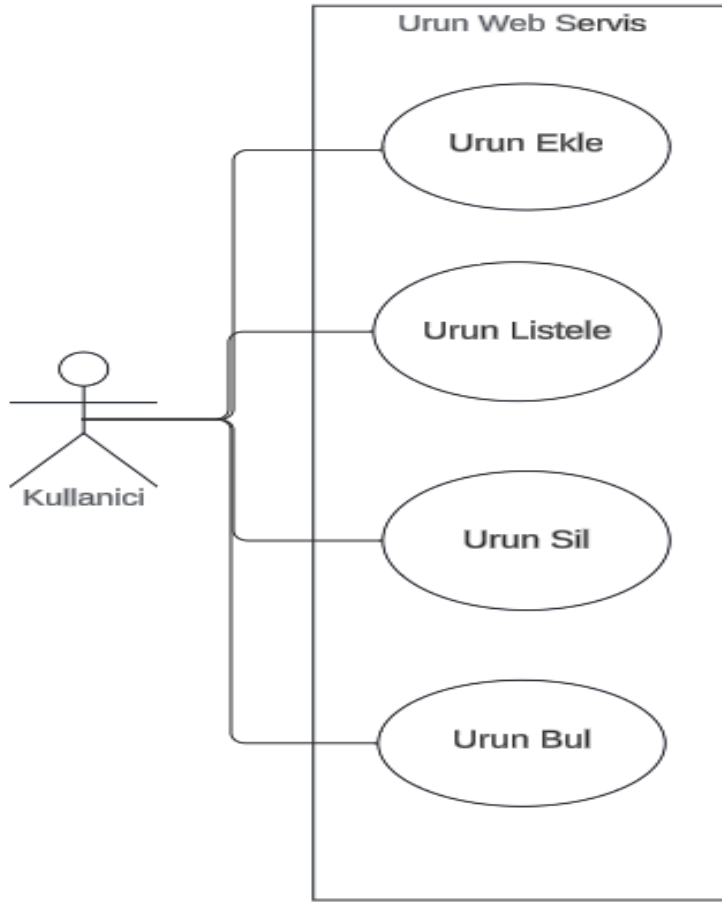
UrunlerWebServisim Sınıfı:

- `URUN_LISTESI`: Ürün nesnelerini içeren bir Liste (`List<Urun>`) özelliği.
- `ekle(urun: Urun): Urun`: Belirtilen Urun nesnesini `URUN_LISTESI`'ne ekler ve eklenen ürünü döndürür.
- `listele(): List<Urun>`: `URUN_LISTESI`'nde bulunan tüm ürünleri içeren bir Liste döndürür.
- `sil(no: String): Urun`: Belirtilen numaraya sahip ürünü `URUN_LISTESI`'nden kaldırır ve kaldırılan ürünü döndürür.
- `bul(no: String): Urun`: Belirtilen numaraya sahip ürünü `URUN_LISTESI`'nde bulur ve bulunan ürünü döndürür.
- `getUrunListesi(): List<Urun>`: `URUN_LISTESI`'ni döndürür.

UML açıklama :

- **Urun --- UrunlerWebServisim ilişkisi** : Urun ile UrunlerWebServisim arasında “0..n” e “0..n” lik bir ilişki vardır bunun anlamı Urun sıfır ya da sonsuz tane UrunlerWebServisim oluşturabilirken aynı şekilde hiç oluşturamayabilir. UrunlerWebServisim 0 kez Urun’e gidebilirken N’kardada gidip gelebilir.
- Tanımladığım “Aggregation” ilişkide UrunlerWebServisim yok olduğunda Urun sınıfım yok olmak zorunda değildir.

3.2-Use Case Diyagramı



- Aktör Kullanici : Kullanıcı sistemde urun ekleyebilir, urunleri listeleyebilir , urunleri silebilir veya bulmak istediği ürünleri bulabilir

4. Birim Testler

Tüm metodlar JUnit kullanılarak test edilmiştir.

JUnitin çalışması için ilk olarak Pom.xml dosyasına yanda bulunan(Şekil1) kod yüklenildi ardından junit @Test leri yazıldı ve test edildi

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <scope>test</scope>
</dependency>
```

Şekil 1 JUnit Kod

testListele() Fonksiyonu;

- Bu fonksiyon, UrunlerWebServisim sınıfının listele() metodunu test eder. Oluşturulan UrunlerWebServisim nesnesi üzerinden listele() çağrılarak ürün listesinin beklenen davranışları kontrol edildi.

testSil() Fonksiyonu;

- Bu fonksiyon, UrunlerWebServisim sınıfının sil() metodunu test eder. Önce bir test ürünü eklenir, sonra bu ürün silinir ve bu işlemlerin beklenen davranışları kontrol edildi

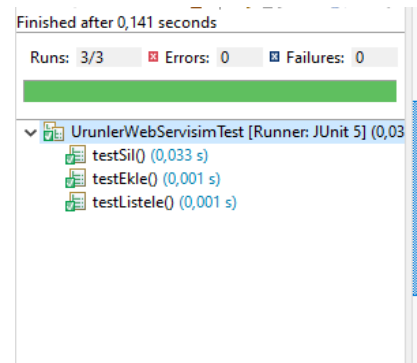
testEkle() Fonksiyonu;

- Bu fonksiyon, UrunlerWebServisim sınıfının ekle() metodunu test eder. Bir test ürünü eklenir ve eklenen ürünün beklenen davranışları kontrol edildi.

4.1 JUnit

Yazdığım test kodlarını derlemek için test sınıfıma girip programa sağ tıklayıp “JUnit Test” komutunu çalıştırdım.

Çıkan Sonucu (Şekil 3) görebilirsiniz eror ve failure almadığımı gördükten sonra programımın html raporunu görmek için “Jacoco” ekleme aşamasına geçtim.



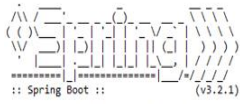
Şekil 3 Junit Test

4.2- Jacoco

Yaptığım testlerin Html Rapor'unu görebilmek için jacoco eklendi. Jacocoyu ekleyebilmek için pom.xml dosyasının içerisine yanda görülen (Şekil 4) kodu yazıp ekledim.

Jacoco uygulaması bir Maven konfigürasyonu, Jacoco (Java Code Coverage Library) eklentisini kullanarak birim test kapsama raporu oluşturmak için kullanılır.

```
<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>
```



```
Spring Boot :: (v3.2.1)

2024-01-13T16:55:53.955+03:00 INFO 26956 --- [main] .p.R.RestWebServisProjeApplicationTests : Starting RestWebServisProjeApplicationTests using Java 17.0.9 with PI
2024-01-13T16:55:53.957+03:00 INFO 26956 --- [main] .p.R.RestWebServisProjeApplicationTests : No active profile set, falling back to 1 default profile: "default"
2024-01-13T16:55:55.039+03:00 INFO 26956 --- [main] .p.R.RestWebServisProjeApplicationTests : Started RestWebServisProjeApplicationTests in 1.296 seconds (process
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.486 s -- in tr.edu.medipol.webservis.proje.RestWebServisProjeApplicationTests
[INFO] Running tr.edu.medipol.webservis.proje.RestWebServisProje.UrunlerWebServisimTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 s -- in tr.edu.medipol.webservis.proje.RestWebServisProje.UrunlerWebServisimTest
[INFO] Results:
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ RestWebServisProje ---
[INFO]
[INFO] --- spring-boot:3.2.1:repackage (repackage) @ RestWebServisProje ---
[INFO] Replacing main artifact C:\Users\berka\OneDrive\masaüstü\RestWebServisProje\target\RestWebServisProje-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependen
The original artifact has been renamed to C:\Users\berka\OneDrive\masaüstü\RestWebServisProje\target\RestWebServisProje-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ RestWebServisProje ---
[INFO] Installing C:\Users\berka\OneDrive\masaüstü\RestWebServisProje\pom.xml to C:\Users\berka\.m2\repository\tr\edu\medipol\webservis\proje\RestWebServisProje\0.0.1-SNAPSHOT
[INFO] Installing C:\Users\berka\OneDrive\masaüstü\RestWebServisProje\target\RestWebServisProje-0.0.1-SNAPSHOT.jar to C:\Users\berka\.m2\repository\tr\edu\medipol\webservis\pr
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 5.066 s
[INFO] Finished at: 2024-01-13T16:55:56+03:00
[INFO]
```

Yukarıda da görüldüğü üzere (Şekil 2) programımızda testleri yazdıktan sonra herhangi bir sıkıntı olup olmadığını görmek için -Maven install yapıldı ve Build Success görüldü bu programımızın düzgün çalıştığını gösteriyor

Birim Test Kapsama raporunu görmek için dosyalarımın target>site>jacoco ardından index.html dosyasına sağ tıklayıp open with > web editör seçeneklerini seçtim ve web editörde yazdığım testlerin raporunu aşağıda paylaştım (Şekil 5)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
tr.edu.medipol.webservis.proje.RestWebServisProje	10 of 97	%89	2 of 6	%66	4 15	3 22	2 12	0 3
Total	10 of 97	%89	2 of 6	%66	4 15	3 22	2 12	0 3

Şekil 5 HTML Rapor

5. Sürekli Entegrasyon (Github Actions)

Projemizi githuba yükledikten sonra githubta kod kapsama oranını görebilmek için aşağıdaki adımlar takip edildi

Actionsa girildi > Java With Maven açıldı > Java with maven içerisine kodlar yazıldı > Commit changes yapıldı > Create Readme.md açıldı > Yeni bir branch ve pull request yapmasını istedim > Propose changes'e basıldı ve ekrana gelen rapor aşağıda paylaşıldı. > merge pull request ile pr birleştirildi.

GitHub Actions Konfigürasyonu (maven.yml):

- maven.yml dosyası, GitHub Actions tarafından kullanılan bir konfigürasyon dosyasıdır.
- Bu dosya, her bir pull request'in ana dal (main) üzerine geldiğinde çalışacak iki farklı iş (job) içerir: build ve coverage.
- build işi, projeyi derlemek ve test etmek için kullanılır.
- coverage işi, Jacoco aracılığıyla kapsama raporu oluşturmak ve bu raporu pull request içinde bir yorum olarak paylaşmak için kullanılır.
- Her iki iş de Ubuntu işletim sistemini kullanır ve Java 17 sürümünü kurar.

README.md Dosyası:

- Bu dosyaya projenin adını, sürekli entegrasyon durumu için bir bağlantıyı ve kapsama raporunu nasıl bulabileceklerini gösteren bilgiler eklendi.

6. Postman & JMeter Testleri

6.1 Postman

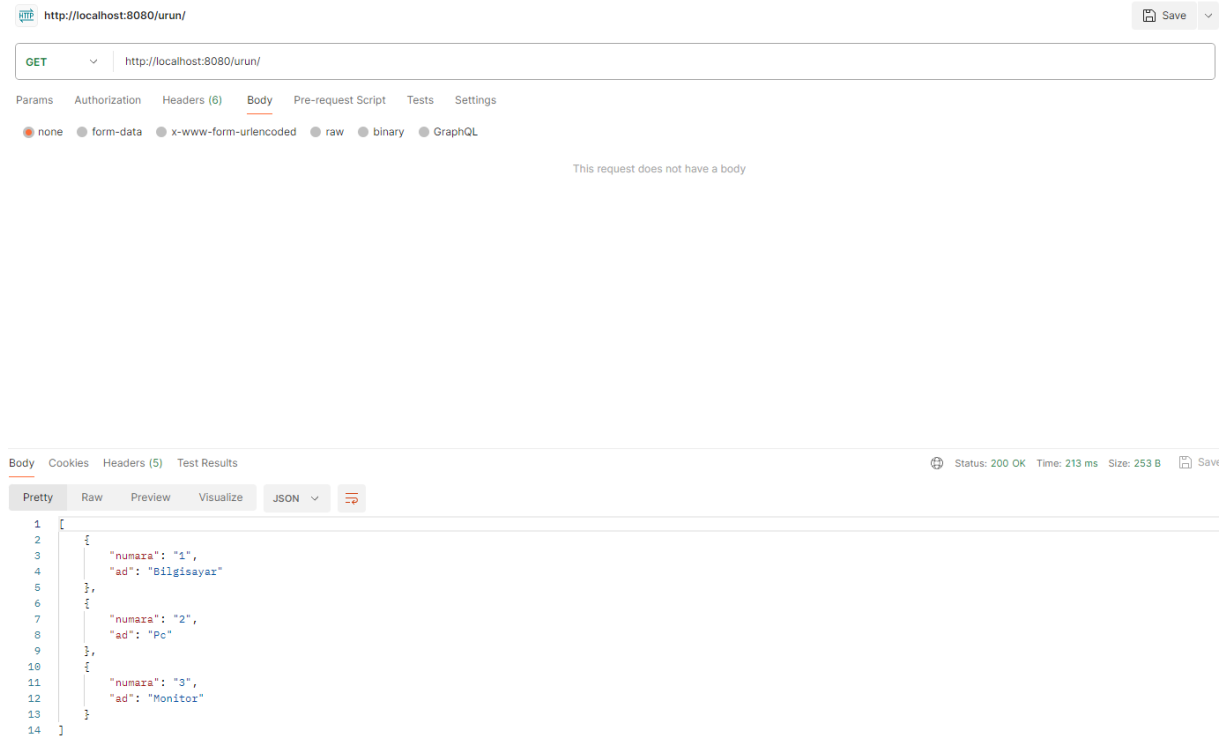
Postman, API'lerin geliştirilmesi, test edilmesi ve belgelenmesi için kullanılan bir platformdur.

RestWebServisimin çalışması ve içerisindeki verileri görebilmek ve içerisine veri girebilmek için öncelikle Eclipse'de bulunan projemi açıp main classımı çalıştırdım ve portumun açılmasını sağladım

GET:

--GET metodu, bir kaynağı almak veya sorgulamak için kullanılan bir metoddur

Postman'i açtım new e basıp http'yi seçtim ve ardından GET'i seçtim ve GET'in yanında bulunan "URL" girme yerine gelerek <http://localhost:8080/urun/> Url'mi yapıştırdım ardından en sağda bulunan "SEND" tuşuna bastım



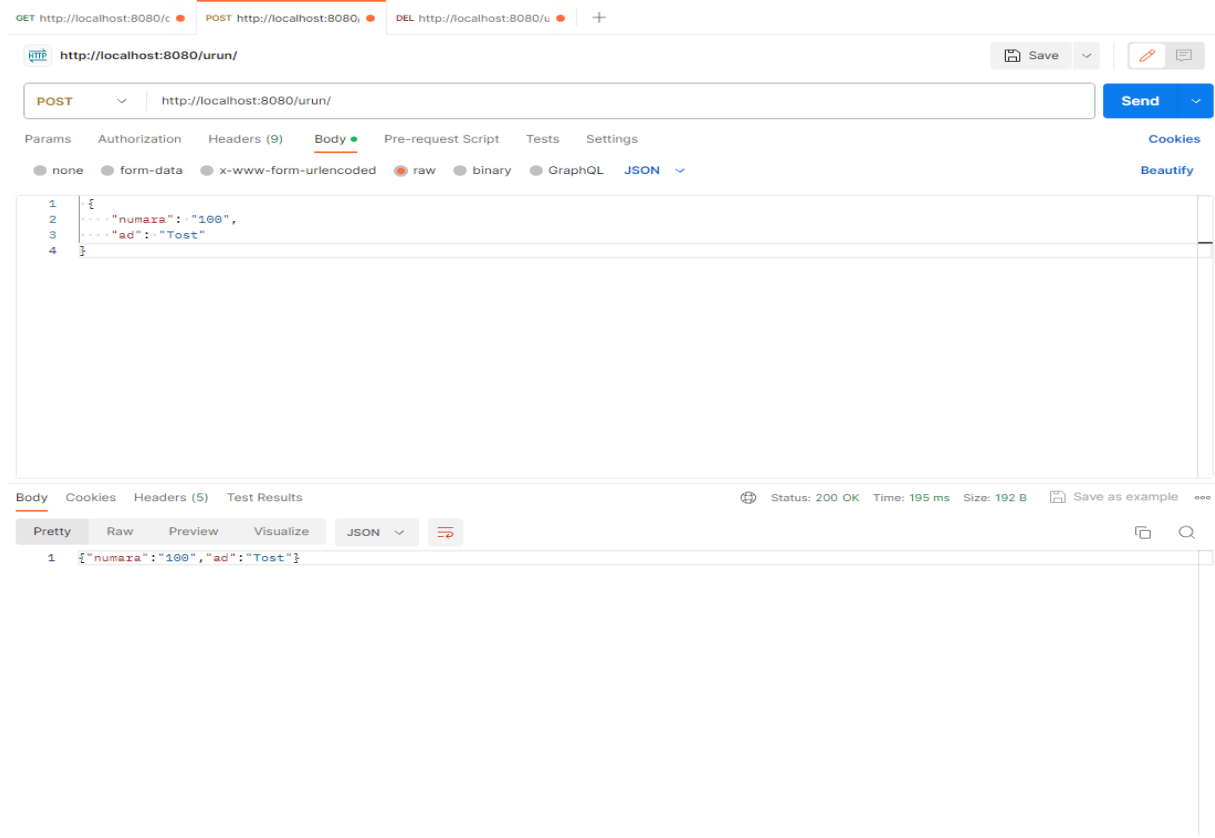
Yukarıdaki fotoğrafta Postman'e eklediğim "GET" ile birlikte eğer web servisimin içerisinde halihazırda veriler varsa onları direk ekrana yazdırmasını sağladım.

POST:

-- POST metodu, bir kaynağı oluşturmak, güncellemek veya bir işlemi gerçekleştirmek için kullanılan bir metoddur .

GET'in yanında bulunan aşağı doğru bulunan ok işaretine basıp POST seçiyoruz ve URL Kısımına basıyoruz ardından Url'mizi giriyoruz <http://localhost:8080/urun/> girilen url'nin ardından ufak bir ayar yapmamız gerekiyor.

Url kısmının altında bulunan sekmelerden "raw" yazan yeri seçip sağda bulunan dropdown'dan JSON yazanı seçiyorum bunu yapmamızın sebebi web servisimiz Json formatında çalışır.

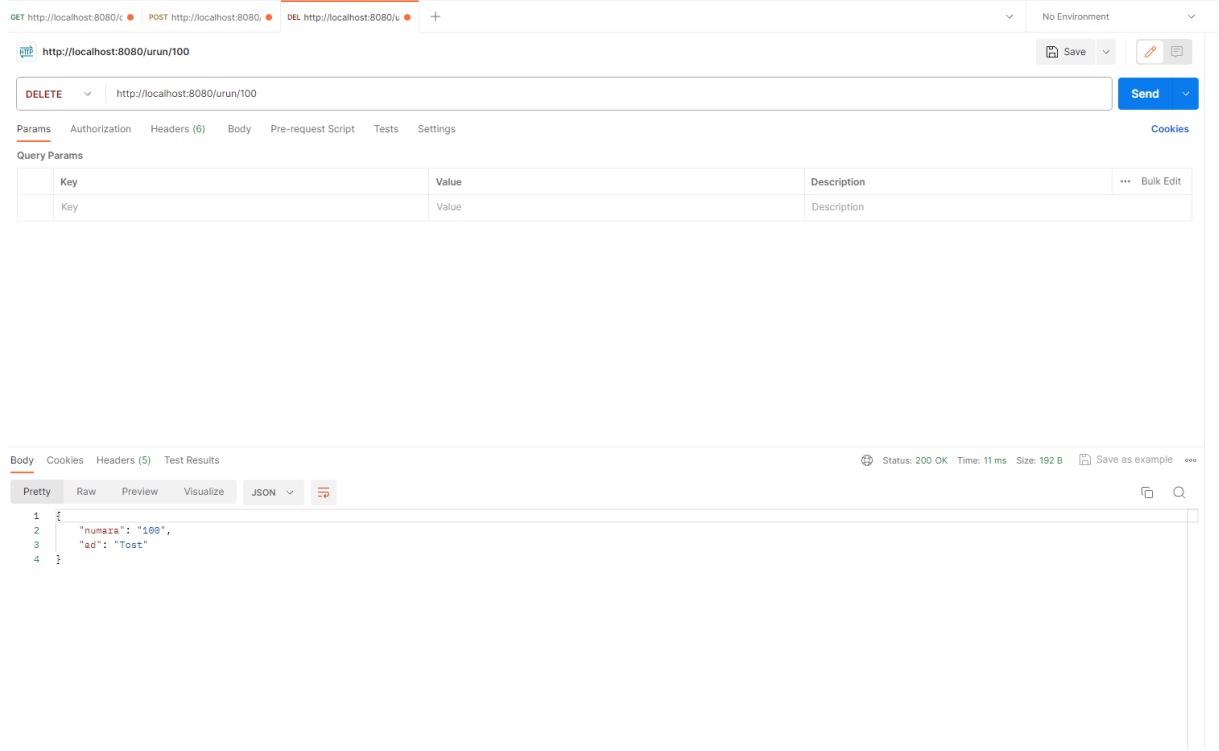


Yukarıdaki fotoğrafta görüldüğü gibi "body" kısmına web servisine göndereceğim verileri girip Send'e basıyorum ve verilerim artık servisimizde görünür olacaktır.

DELETE:

--DELETE metodu, belirtilen kaynağı silmek veya kaldırmak için kullanılır.

Yukarıda yapmış olduğum gibi aşağı doğru ok olan imlece basıp DELETE' i seçiyorum ve ardından url girilen yere <http://localhost:8080/urun/silmekistediğimizid> silmek istediğimiz id yazan yere web servisin içerisinde bulunan bir numara yazmalısınız eğer içerisinde bulunan bir numara yoksa eklemelisiniz. Benim web servisimin içerisinde yukarıda eklediğim 100 id'sini sileceğim <http://localhost:8080/urun/100>



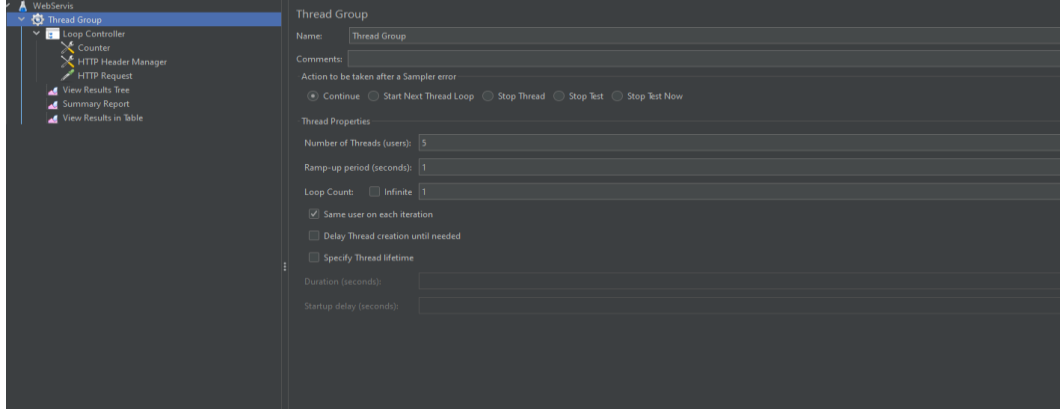
Yukarıdaki fotoğraf ile birlikte urun servisinin içerisinde bulunan 100 id'li tablo silinmiş olacaktır eğer istiyorsanız yeniden GET metodunu seçip çalıştırıp görebilirsiniz.

6.2 JMeter

JMeter, web uygulamaları üzerindeki performansın, yük dayanıklılığının ve genel sistem performansının test edilmesine olanak sağladığı için bu projede JMeter kullandık.

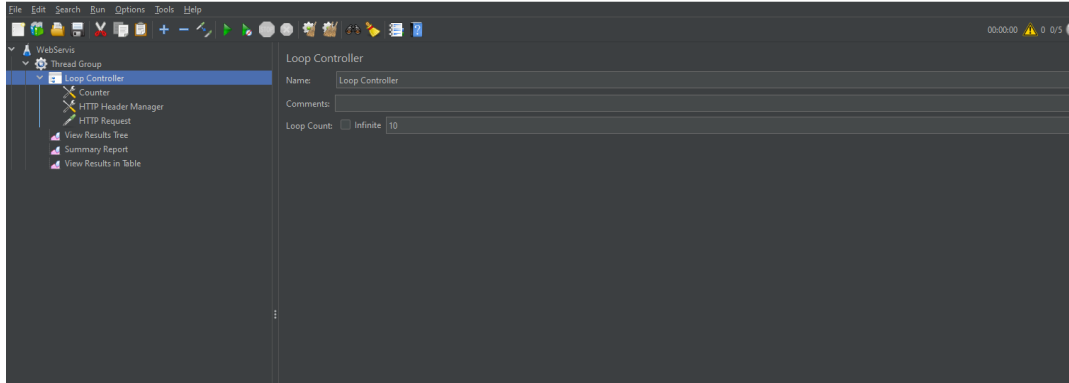
1-Thread Group:

- Number of Thread: Threadimizin içerisinde kaç kullanıcı olacağını belirledik. 5 kullanıcı olacağını söyledim
- Ramp-up period : Threadin kaç saniye içerisinde başlayacağını belirledik. 1 saniyede başlayacağını söyledim
- Loop Count : Her kullanıcı için kaç kere tekrarlanacağını söylemeye yarar. Burada 1 olarak girdim fakat burada değil asıl loop controllerın içerisinde kaç kere tekrarlanacağını söyledim.



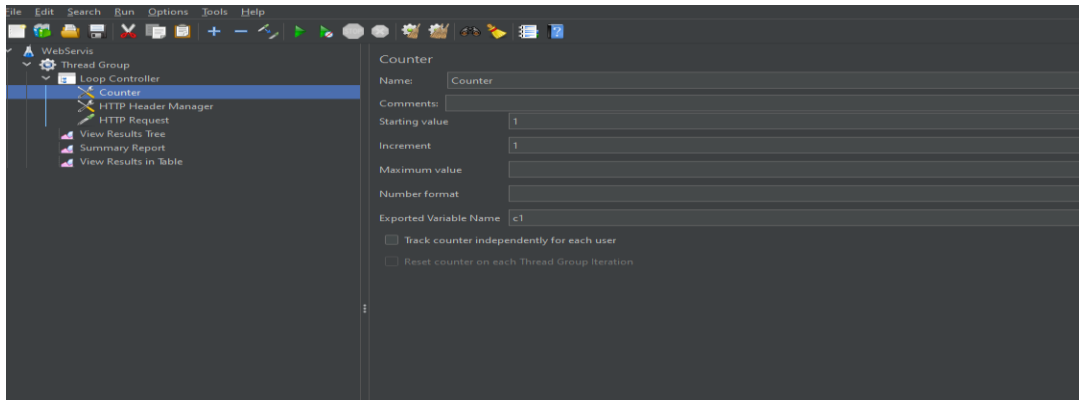
2-Loop Controller:

- Loop Count : Her 1 kullanıcı için 10 kere tekrarlanacağını yani 5 kullanıcı için 50 kere tekrarlanacağını girdim.



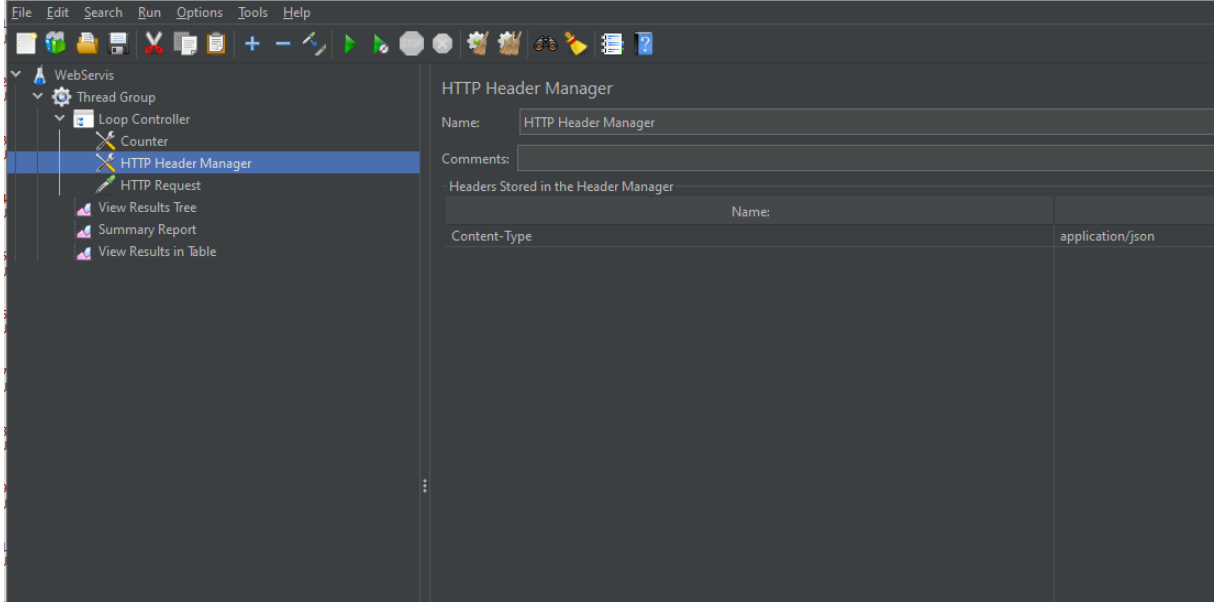
3-Counter

- Starting Value : başlangıç değerini 1 olarak yaptım
- Increment : sayıcı her çalıştırıldığında değer 1 artırılabacak
- Exported Variable Name : değer her artırıldığında 1 1 artacak değeri belirledim



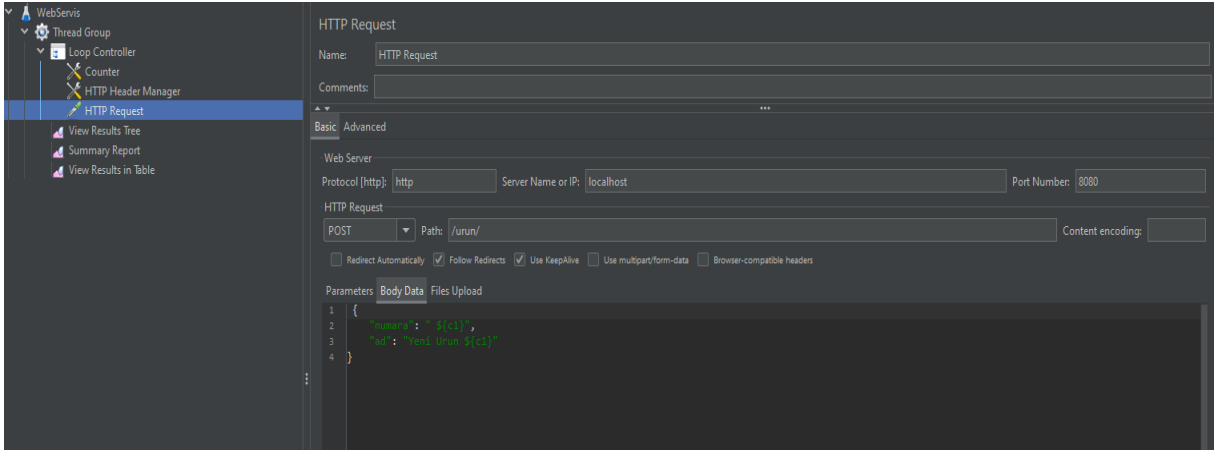
4-HTTP Header Manager

- Girilen verilerin json formatında olacağı için ayar yapmamız gerekiyor.
- Name yerine Content-Type Value yerine ise application/json yazdım ve verilerin json formatında gidip test edilmesini sağladım.



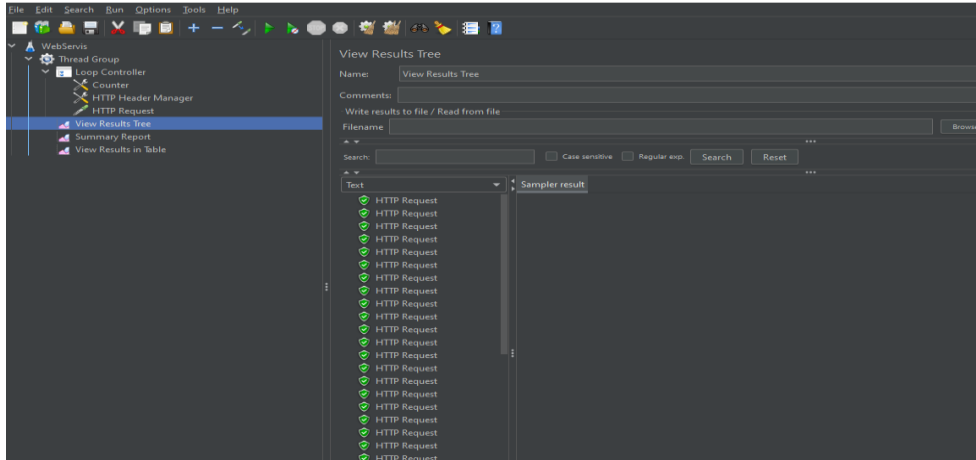
5- HTTP Request

- Protocolunu http belirttim server name ını localhost ve portumda 8080'de ayarlandığı için 8080 yazdım.
- Verileri web servisine göndereceğim için http request yazan yeri post yaptım
- Path kısmına "/urun/" yazdım kaynağa erişebilmek için
- Web servisimde bulunan şekilde verilerin girilmesi için body data yerine web servisimde bulunan şekilde giriş yaptım



6- View Result Tree

- Gönderdiğimiz http isteklerini detaylı bir şekilde görmek için açtım.



7-Summary Report

- Testin genel başarı durumu ve performansın özetini görmek için kullandım.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Arg. Bytes
HTTP Request	50	2	0	44	5.91	0.00%	60.8/sec	12.39	13.46	208.6
TOTAL	50	2	0	44	5.91	0.00%	60.8/sec	12.39	13.46	208.6

8-View Result in Table

- Test sırasında aldığım yanıtları görmek için kullandım

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time
1	18:16:00.840	Thread Group 1-1	HTTP Request	44	✓	207	225	43	
2	18:16:00.884	Thread Group 1-1	HTTP Request	3	✓	207	225	2	
3	18:16:00.887	Thread Group 1-1	HTTP Request	3	✓	207	225	2	
4	18:16:00.890	Thread Group 1-1	HTTP Request	2	✓	207	225	2	
5	18:16:00.892	Thread Group 1-1	HTTP Request	2	✓	207	225	2	
6	18:16:00.894	Thread Group 1-1	HTTP Request	2	✓	207	225	2	
7	18:16:00.896	Thread Group 1-1	HTTP Request	3	✓	207	225	3	
8	18:16:00.899	Thread Group 1-1	HTTP Request	3	✓	207	225	3	
9	18:16:00.902	Thread Group 1-1	HTTP Request	2	✓	207	225	2	
10	18:16:00.904	Thread Group 1-1	HTTP Request	2	✓	209	227	2	
11	18:16:01.051	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
12	18:16:01.054	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
13	18:16:01.057	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
14	18:16:01.059	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
15	18:16:01.061	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
16	18:16:01.063	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
17	18:16:01.065	Thread Group 1-2	HTTP Request	2	✓	209	227	2	
18	18:16:01.068	Thread Group 1-2	HTTP Request	1	✓	209	227	1	
19	18:16:01.069	Thread Group 1-2	HTTP Request	2	✓	209	227	1	
20	18:16:01.071	Thread Group 1-2	HTTP Request	1	✓	209	227	1	
21	18:16:01.253	Thread Group 1-3	HTTP Request	3	✓	209	227	3	
22	18:16:01.256	Thread Group 1-3	HTTP Request	3	✓	209	227	3	
23	18:16:01.259	Thread Group 1-3	HTTP Request	2	✓	209	227	2	
24	18:16:01.261	Thread Group 1-3	HTTP Request	2	✓	209	227	2	
25	18:16:01.263	Thread Group 1-3	HTTP Request	2	✓	209	227	2	
26	18:16:01.265	Thread Group 1-3	HTTP Request	2	✓	209	227	2	

9-JMeterda test için oluşturduğumuz verileri görmek için tarayıcıya paylaştığımız verileri görmek için tarayıcımıza localhost:8080/urun/ yazıp verileri gördük

```
1  [
2      {
3          "numara": " 1",
4          "ad": "Yeni Urun 1"
5      },
6      {
7          "numara": " 2",
8          "ad": "Yeni Urun 2"
9      },
10     {
11         "numara": " 3",
12         "ad": "Yeni Urun 3"
13     },
14     {
15         "numara": " 4",
16         "ad": "Yeni Urun 4"
17     },
18     {
19         "numara": " 5",
20         "ad": "Yeni Urun 5"
21     },
22     {
23         "numara": " 6",
24         "ad": "Yeni Urun 6"
25     },
26     {
27         "numara": " 7",
28         "ad": "Yeni Urun 7"
29     },
30     {
31         "numara": " 8",
32         "ad": "Yeni Urun 8"
33     },
34     {
35         "numara": " 9",
36         "ad": "Yeni Urun 9"
37     },
38     {
39         "numara": " 10",
40         "ad": "Yeni Urun 10"
41     },
42     {
43         "numara": " 11",
44         "ad": "Yeni Urun 11"
45     },
46     {
47         "numara": " 12",
48         "ad": "Yeni Urun 12"
49     },
50     {
51         "numara": " 13",
52         "ad": "Yeni Urun 13"
53     },
54     {
55         "numara": " 14",
56         "ad": "Yeni Urun 14"
57     },
58     {
59         "numara": " 15",
60         "ad": "Yeni Urun 15"
61     },
62     {
63         "numara": " 16",
64         "ad": "Yeni Urun 16"
65     },
66     {
67         "numara": " 17",
68         "ad": "Yeni Urun 17"
69     },
70 ]
```


7. Git

Github Repository Adres : [wayolapre/SpringBootWebServis_Proje \(github.com\)](https://github.com/wayolapre/SpringBootWebServis_Proje)


Github'a kodlarımı eklemek için Eclipse'i kullandım

1- Githubta oluşturduğum repositoryye ait linki kopyaladım.

2- Eclipse'te Git repositories yerine gelip CTRL+V yapıp yapıştırdım ve Next > Next >Finish yapıp Github Repo mu ekledim

 Clone Git Repository

Source Git Repository
Enter the location of the source repository.



Location

URI:

Host:

Repository path:

Connection

Protocol:


Port:

Authentication

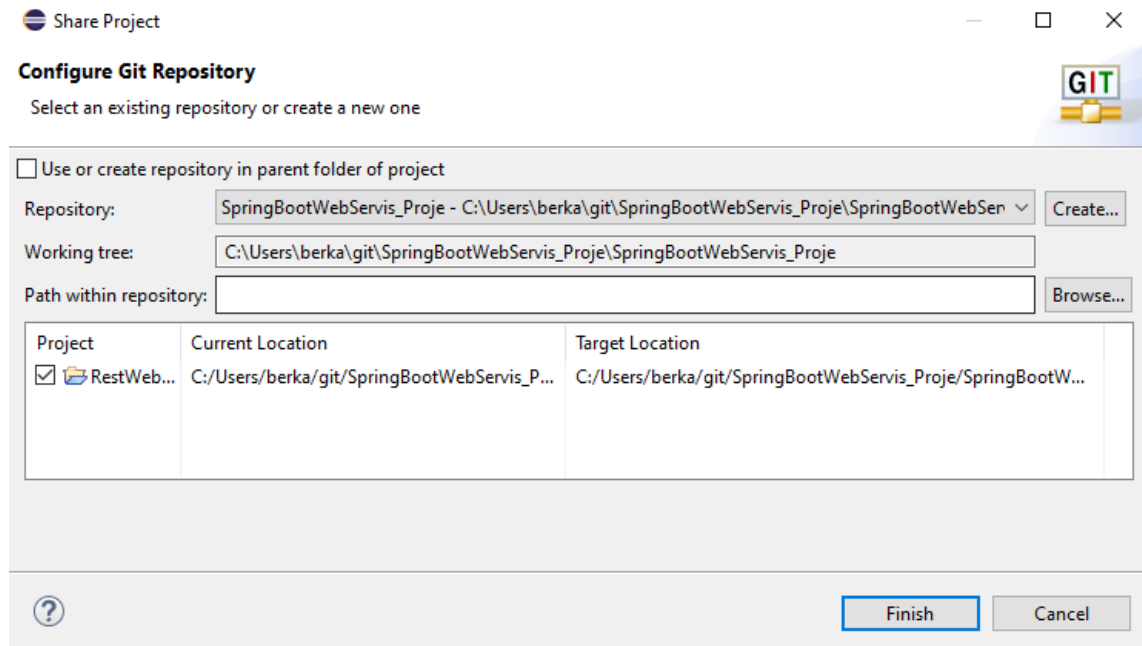
User:

Password:

☒ Store in Secure Store



3-RestWebProjeme gelip sağ tıkladım ardından Team > Share Project yapıp yukarıda oluşturduğum github dosyamı projeme aşağıdaki fotoğrafta bulunan şekilde bağladım



4-Ardından projemde bulunan dosyaları githuba yüklemek için git staging'e gelip "unstaged changes" de bulunan dosyaları seçip "Staged Changes" kısmına aldım ve "Commit Message" ekleyip commit and push yapıp projemi githuba yükledim.

