YAZILIM GELİŞTİRME ORTAM VE ARAÇLARI

Final Projesi

Git Hesabim
https://github.com/batuhank34/Final-Projesi

BATUHAN KUTLUAY
H5160025

İçindekiler

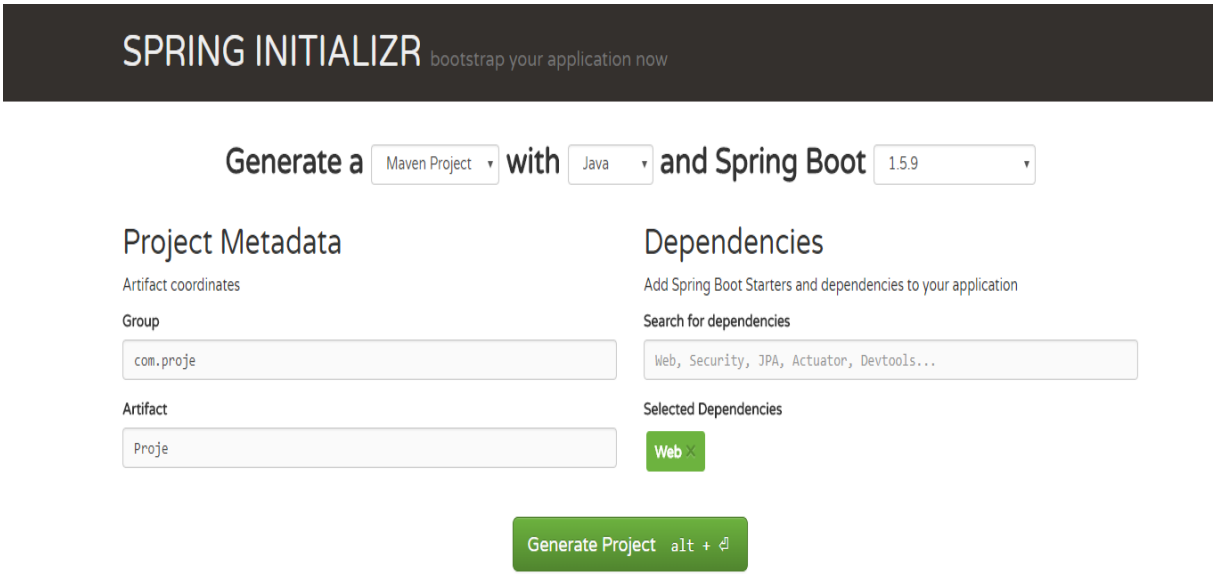
1. MAVEN PROJESİ OLUŞTURMA 3
2. PROJENİN ECLIPSE'E EKLENMESİ 4
3. GITHUB REPOSITORY OLUŞTURMA 5
4. ECLIPSE GIT BAĞLANTISI 6
5. GIT HESABIMIZA PROJENİN GÖNDERİLMESİ 11
6. JACOCO CODE COVERAGE 14
7. COBERTURA CODE COVERAGE 19
8. JAVADOC 23
9. MAVEN SITE OLUŞTURMAK 33
10. TRAVIS CI & CODECOV IO 39
    - 10.1. TRAVIS CI 40
    - 10.2. CODECOV IO 41
    - 10.3. TRAVIS VE CODECOV'UN TETİKLENMESİ 43
    - 10.4. TRAVIS VE CODECOV SONUÇLARI 44
11. POSTMAN VE JMETER 46
    - 11.1. POSTMAN 46
    - 11.2. JMETER 49
12. UML DİAGRAMI 59
13. SONARQUBE 60
14. JENKINS 61
    - 14.1. SonarQube Test Sonuçları 73

1. MAVEN PROJESİ OLUŞTURMA

https://start.spring.io/ sitesinden projemi oluşturdum.Buradan oluşturmamın sebebi hem maven hem de bir spring projesi oluşuyor.

Figur 1 İçin Adımlar

1. Yukarıda verdiğim linke girdim ve karşıma proje oluşturma ekranı geldi.
2. Buradan Artifactleri belirledim.
3. Dependecies kısmında Web'i seçtim çünkü REST servislerle çalışaçağım için Webinde dahil olması gerekiyordu.
4. "Generate Project" diyerek projemi oluşturdum.



Figur 1

2. PROJENİN ECLIPSE'E EKLENMESİ

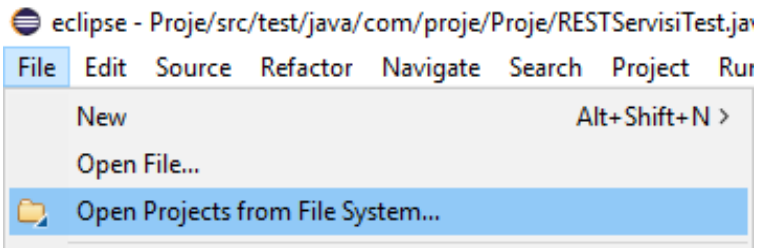
Oluşturduğum projeyi eclipse'e ekleme işlemini gerçekleştirdim.

Figur 2 İçin Adımlar

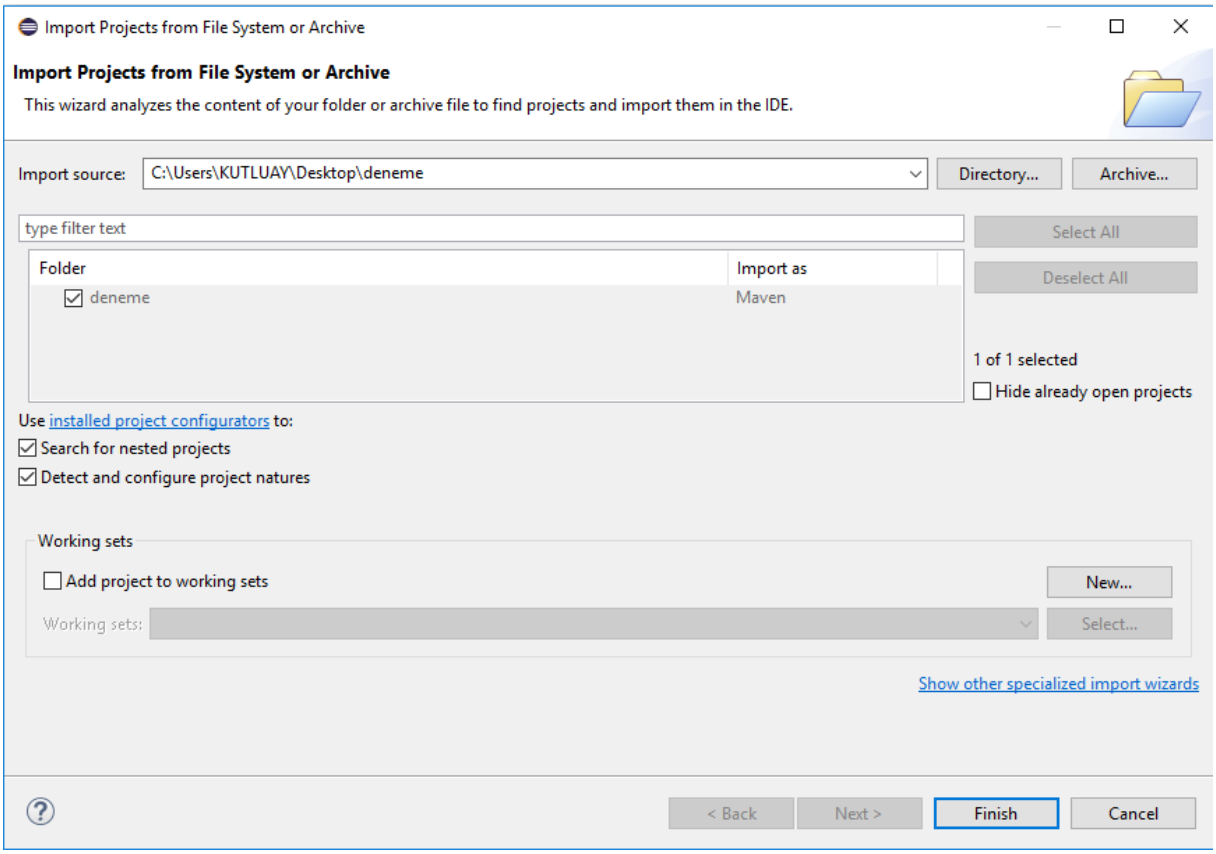
1. Toolbardan "File"ı seçiyoruz.
2. "Open Projects from File System" seçeneğini seçiyoruz.

Figur 3 İçin Adımlar

1. Gelen ekrandan "Directory" kısmından indirdiğimiz projeyi seçiyoruz.
2. "Finish" diyoruz ve projemiz Eclipse'e eklemiş oluyor.



Figur 2



Figur 3

3. GITHUB REPOSITORY OLUŞTURMA

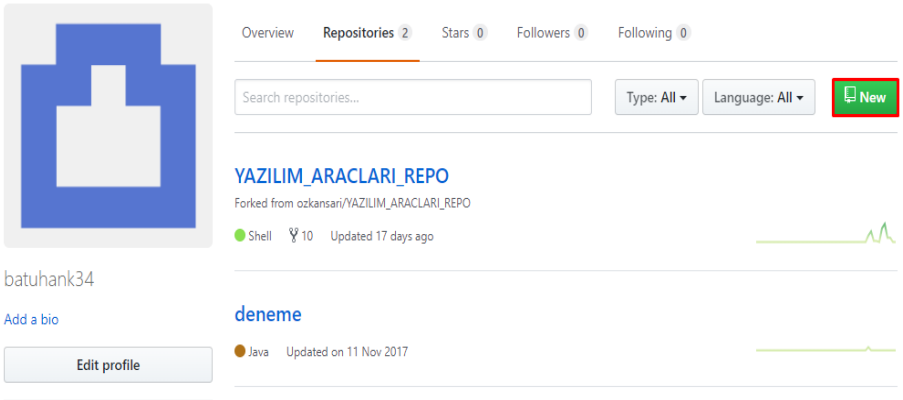
GitHub hesabımda yeni bir repository açtım.Eclipse'e eklediğim projemi GitHub'a gönderdim.

Figur 4 İçin Adımlar

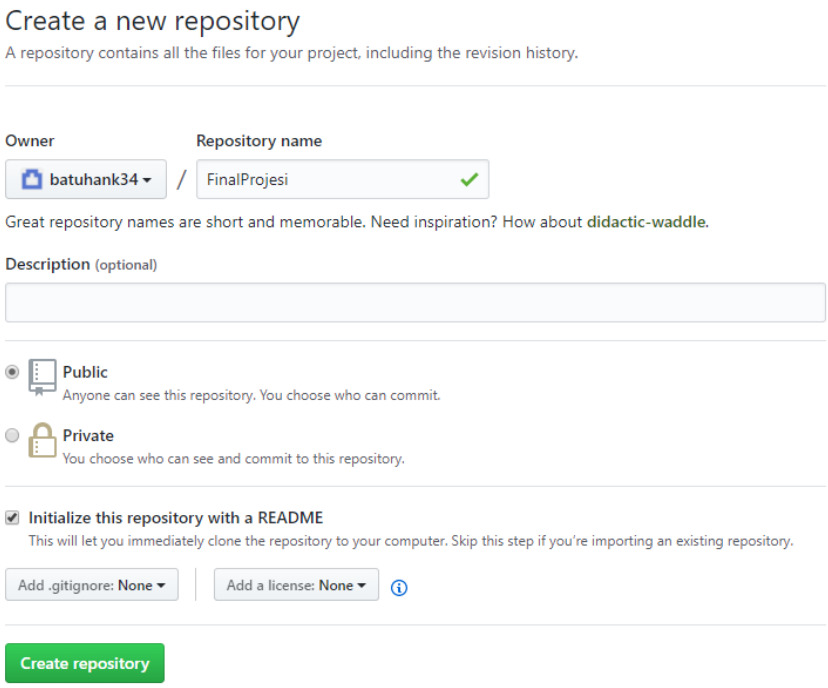
1. GitHub profilimize giriyoruz().
2. "Repositories" sekmesine tıklıyoruz.
3. "New" diyoruz.

Figur 5 İçin Adımlar

1. Açılan ekranda repositoryimize isim veriyoruz.
2. "Create Repository" diyerek repomuzu oluşturuyoruz.



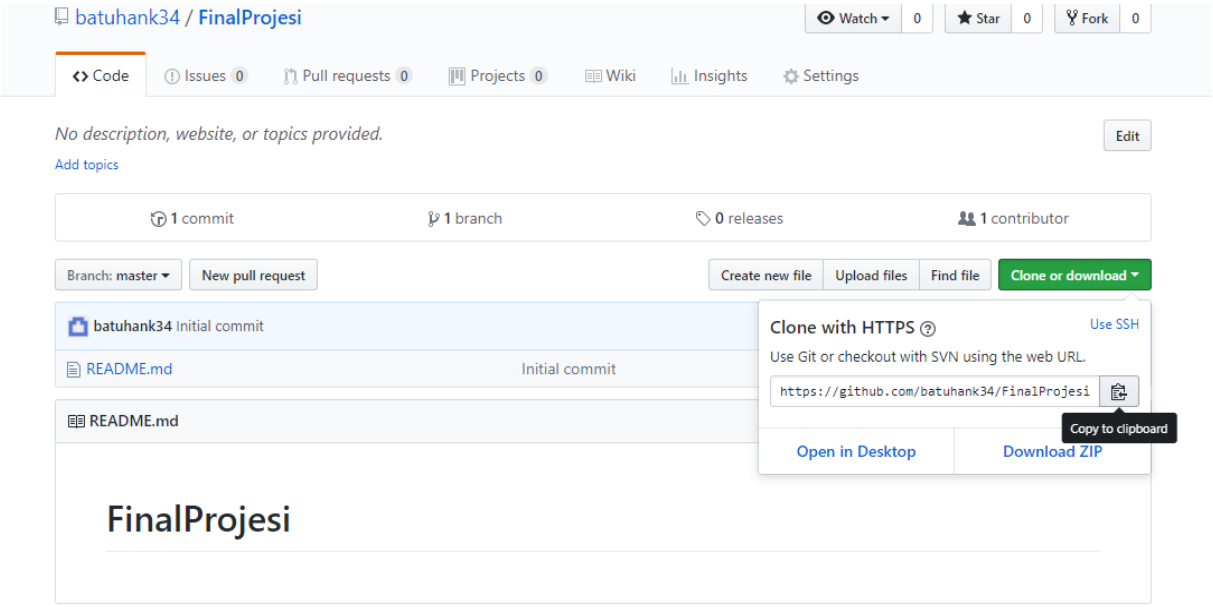
Figur 4



Figur 5

Figur 6 İçin Adımlar

1. “Clone Or Download” seçeneğini seçiyoruz.
2. Çıkan linki kopyalıyoruz.



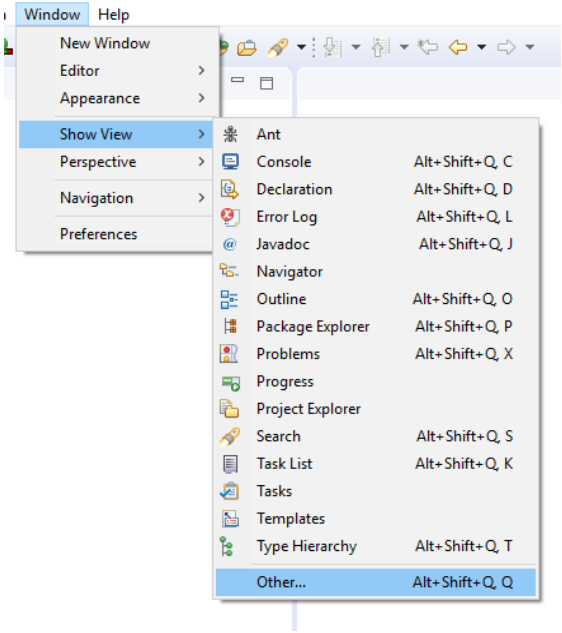
Figur 6

4. ECLIPSE GIT BAĞLANTISI

Oluşturduğum repositoryimin eclipse ile bağlantısını kurdum.

Figur 7 İçin Adımlar

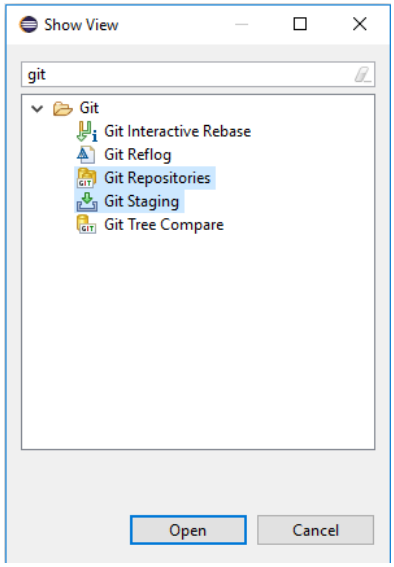
1. Toolbardan “Window” seçeneğini seçiyoruz.
2. “Show View” seçeneğini seçiyoruz.
3. “Other seçeneğini seçiyoruz”.
4. “Figur 8” karşımıza gelir.



Figur 7

Figur 8 İçin Adımlar

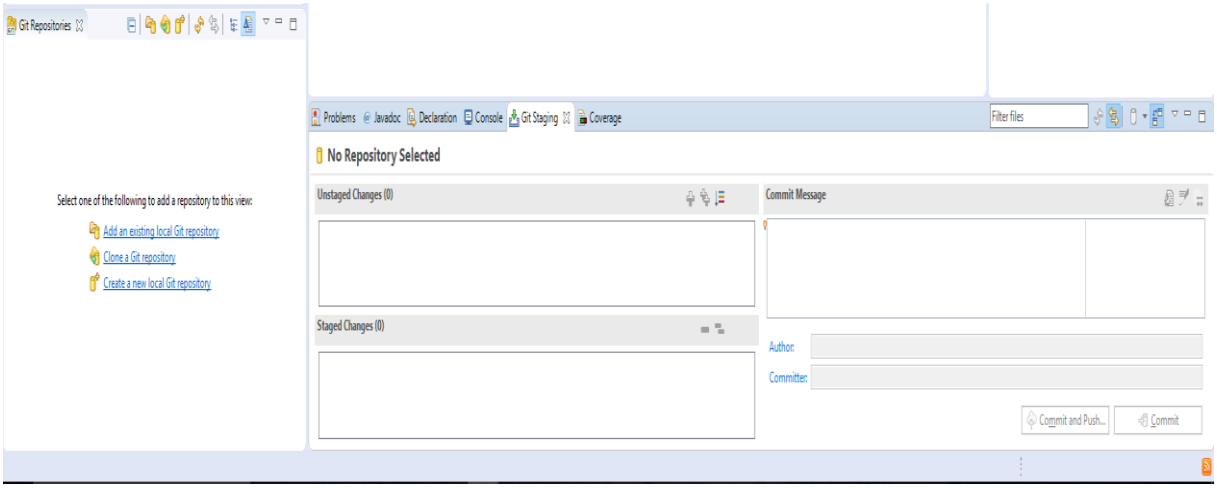
1. Arama yerine “git” yazıyoruz.
2. “Git Repositories” ve “Git Staging” seçeneklerini seçiyoruz.
3. “Open” butonuna tıklıyoruz.
4. Alt tarafta Figur 9 çıkacaktır.



Figur 8

Figur 9 İçin Adımlar

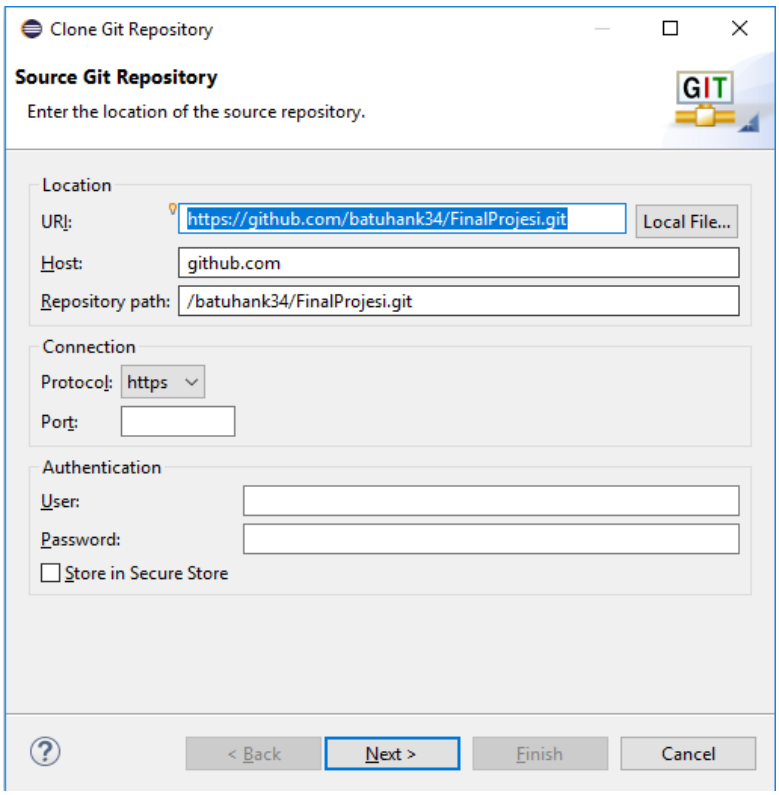
1. Figur 6 da kopyaladığımız repo linkimizi “Git Repository”e tıklayıp boş bir alanda “CTRL+V” yapıyoruz.



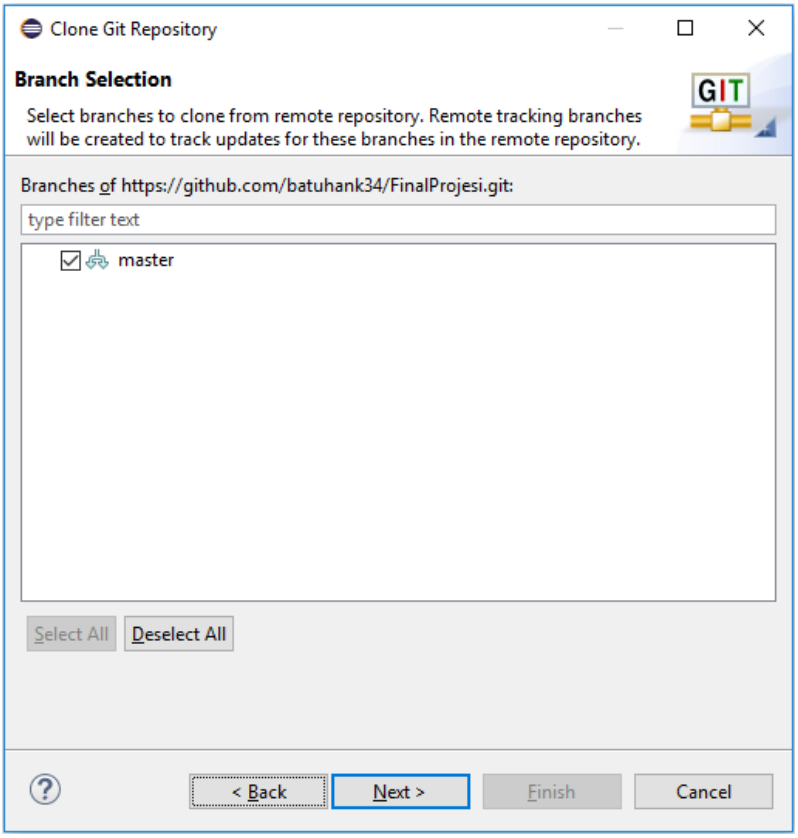
Figur 9

Figur 10 - 11 İçin Adımlar

1. “Next” diyoruz.



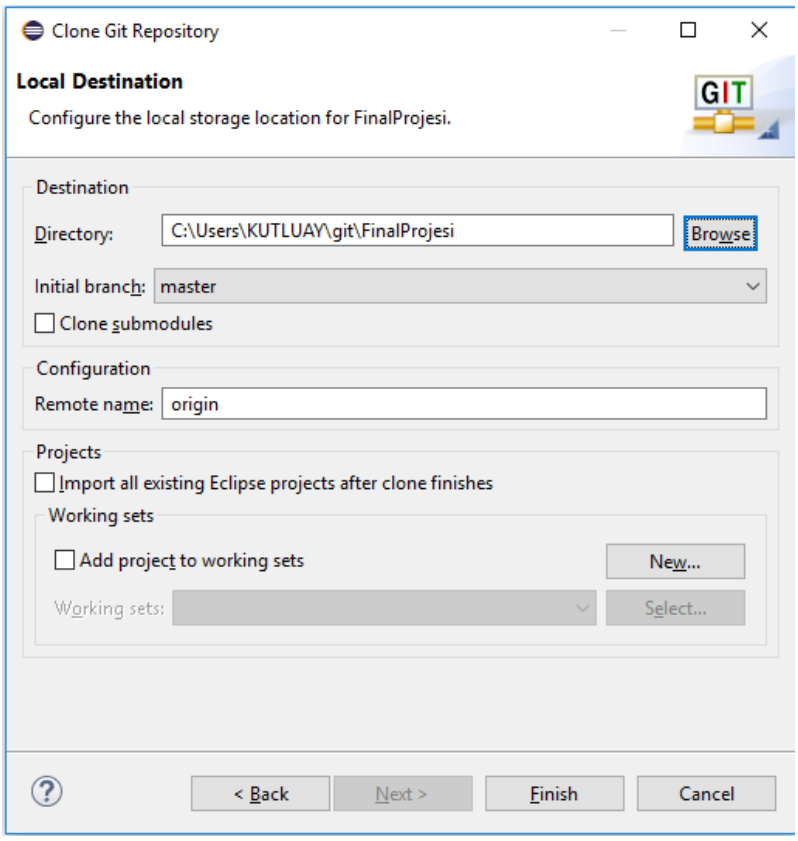
Figur 10



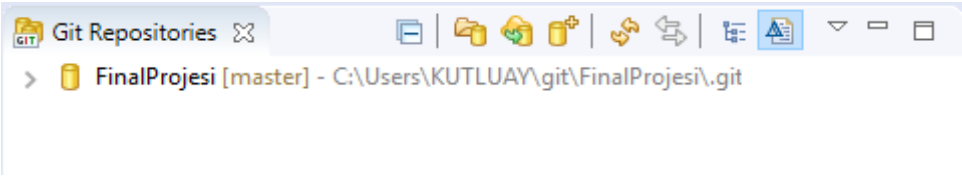
Figur 11

Figur 12 - 13 İçin Adımlar

1. “Directory” kısmında local de projeyi nerede tutabileceğimizi seçiyoruz.
2. “Finish” diyoruz.
3. “Git Repositories” kısmında gitten projeyi çektiğimizi görebiliyoruz(Figur 13).



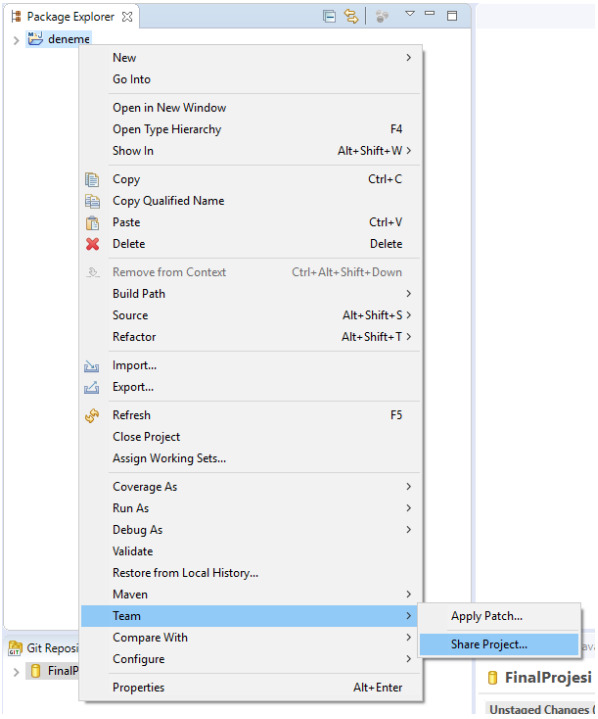
Figur 12



Figur 13

Figur 14 İçin Adımlar

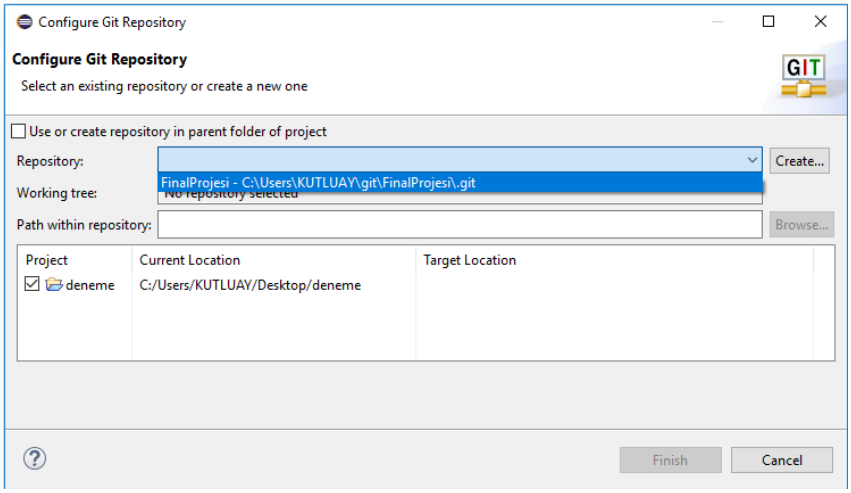
1. Projemize “Sağ Click” yapıyoruz.
2. “Team” seçeneğini seçiyoruz.
3. “Share Project” diyoruz.
4. “Figur 15” karşımıza gelir.



Figur 14

Figur 15 İçin Adımlar

1. “Repository” kısmından eklediğimiz repomumuzu seçiyoruz.
2. “Finish” diyoruz.



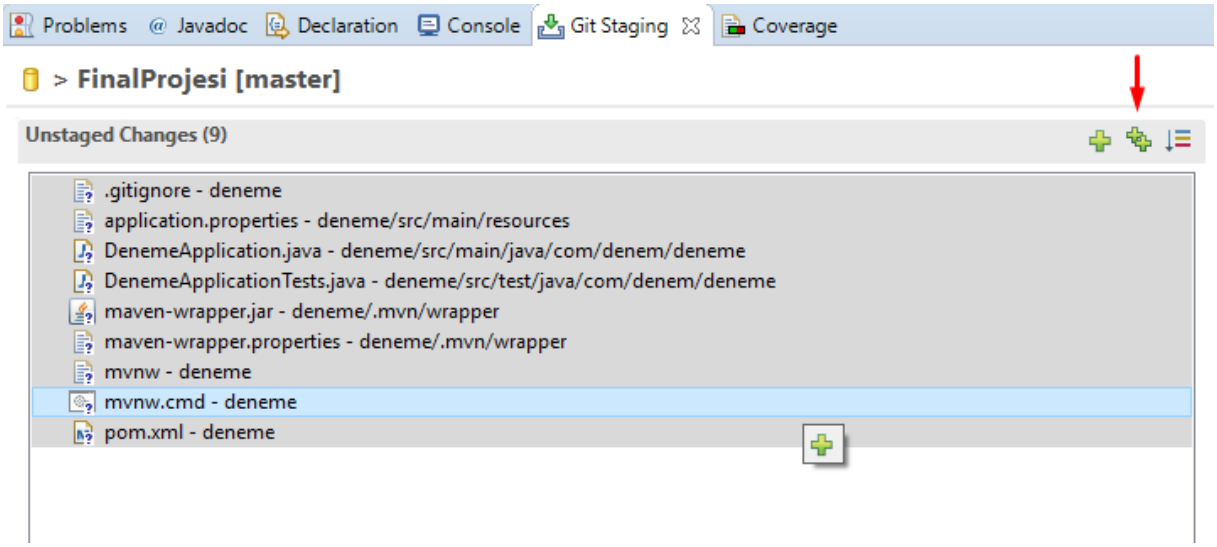
Figur 15

5. GIT HESABIMIZA PROJENİN GÖNDERİLMESİ

Projemi git hesabıma gönderme işlemlerini gerçekleştirdim.

Figur 16 İçin Adımlar

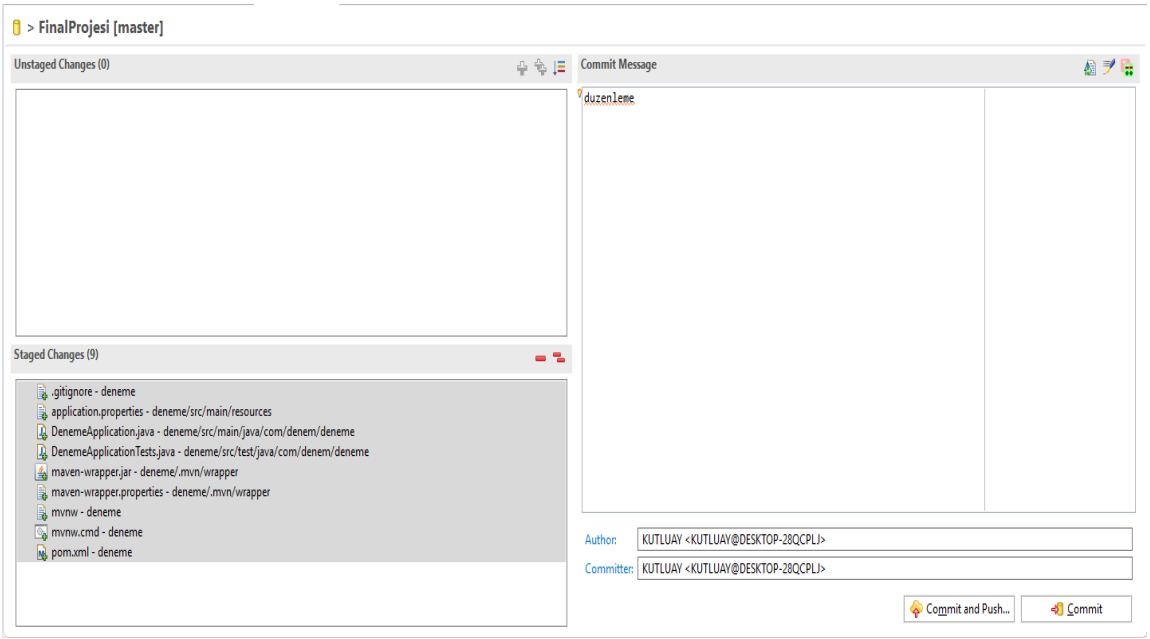
1. "Git Staging" sekmesine geçiyoruz.
2. Git' hesabımıza bütün dosyaları atmak için ok ile gösterilen butona tıklıyoruz.
3. "Unstaged Changes"ten "Stage Changes"a dosyalarımız geçiyor.(Figur 17)



Figur 16

Figur 17 İçin Adımlar

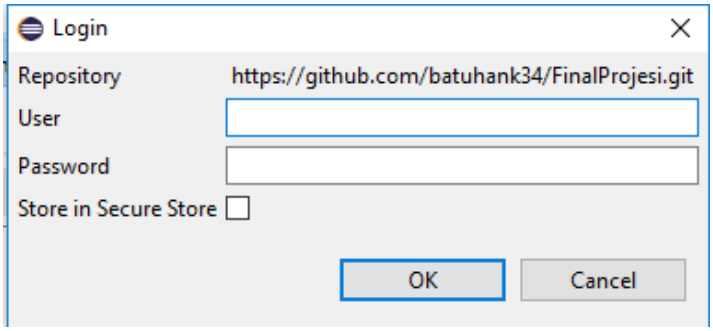
1. "Commit Message" kısmına mesaj yazmak zorundayız.Herhangi bir şey yazabiliriz ben "duzenleme" yazdım.
2. Dosyaları göndermek için "Commit And Push" butonuna tıklıyoruz.
3. Karşımıza "Figur 18" geliyor



Figur 17

Figur 18 İçin Adımlar

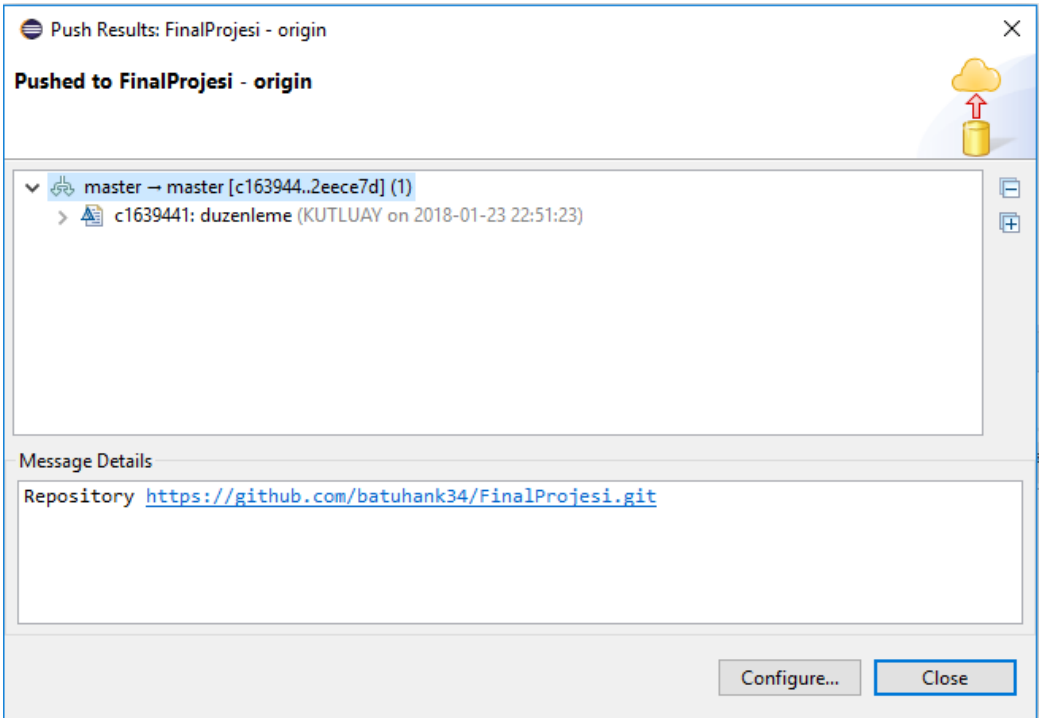
1. Gönderim işlemini gerçekleştirmek için bizden GitHub kullanıcı adı ve şifremizi istiyor onları yazıyoruz.
2. “OK” seçeneğini seçiyoruz.
3. Karşımıza “Figur 19” geliyor



Figur 18

Figur 19 İçin Adımlar

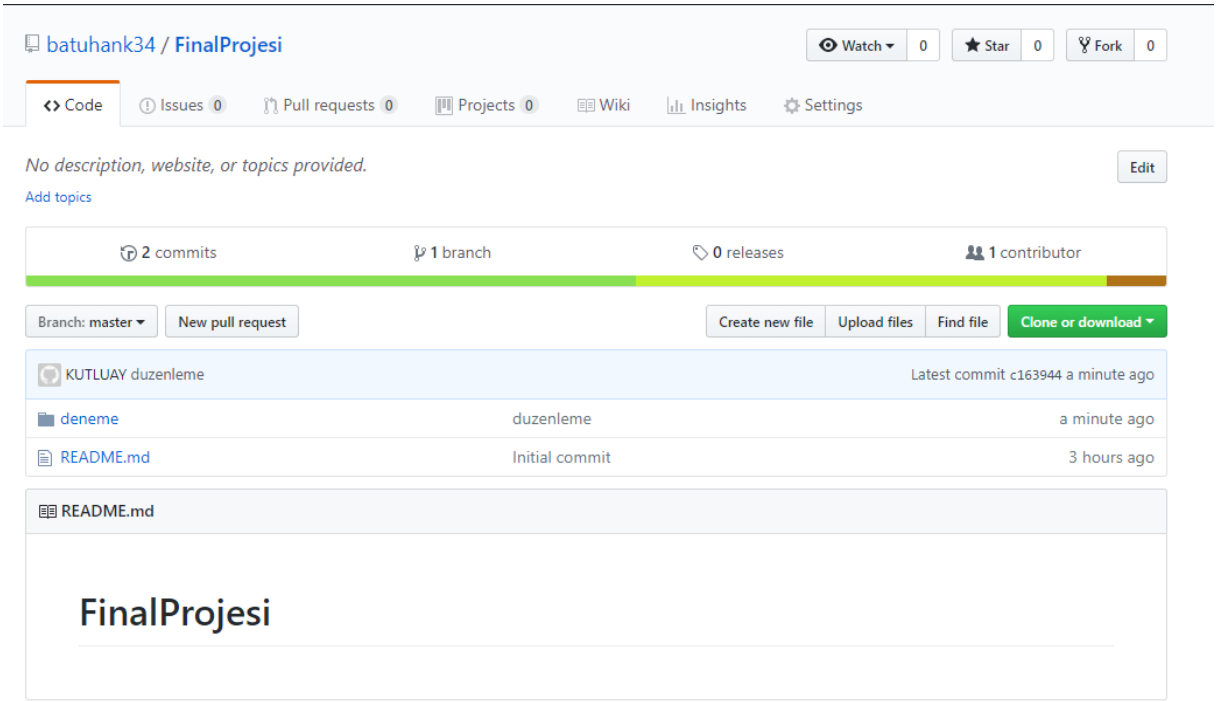
1. Gönderim işleminin bittiğini bize söylüyor.



Figur19

Figur 20 İçin Adımlar

1. GitHub'ta açtığımız reponun sayfasını yenilediğimiz zaman "Commit and Push" yaptığımız dosyaların yani eclipse'e eklediğimiz projenin repomuza gittiğini görebiliriz.



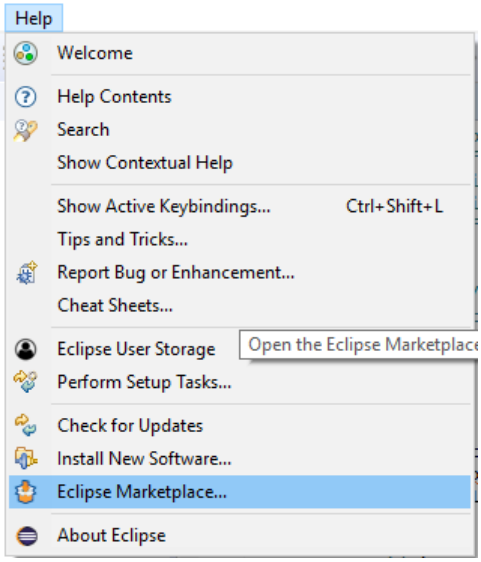
Figur 20

6. JACOCO CODE COVERAGE

Jacoco, kodumuzun ne kadarının test edildiğini bize gösterir.Jacoco'yu projemize uygulayabilmek için gerekli adımlar vardır.

Figur 21 İçin Adımlar

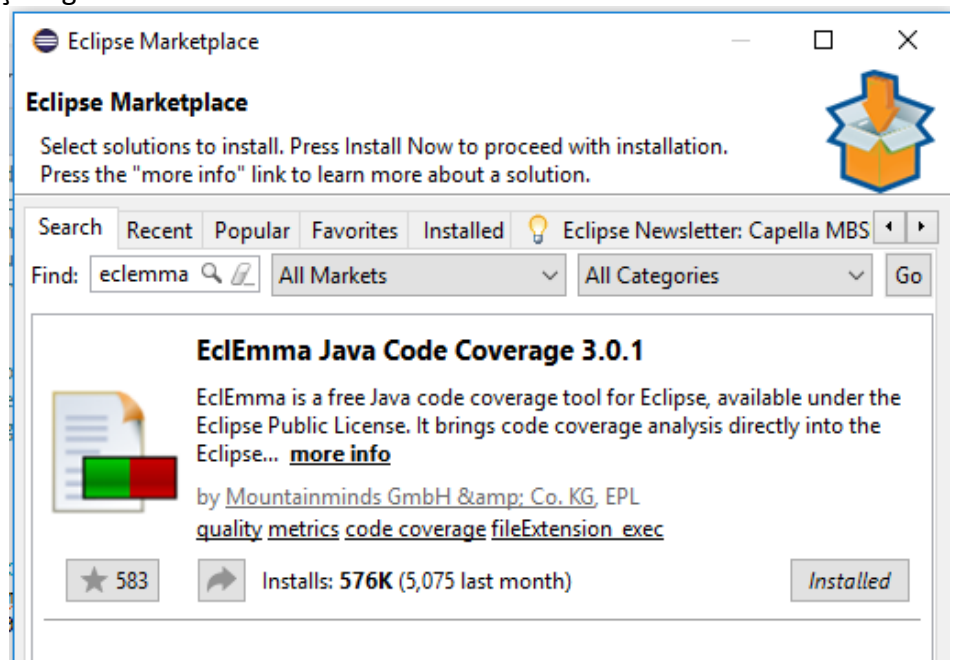
1. Toolbardan "Help" seçeneğini seçiyoruz.
2. "Eclipse Marketplace" seçeneği ile devam ediyoruz.
3. Karşımıza Figur 22 geliyor.



Figur 21

Figur 21 İçin Adımlar

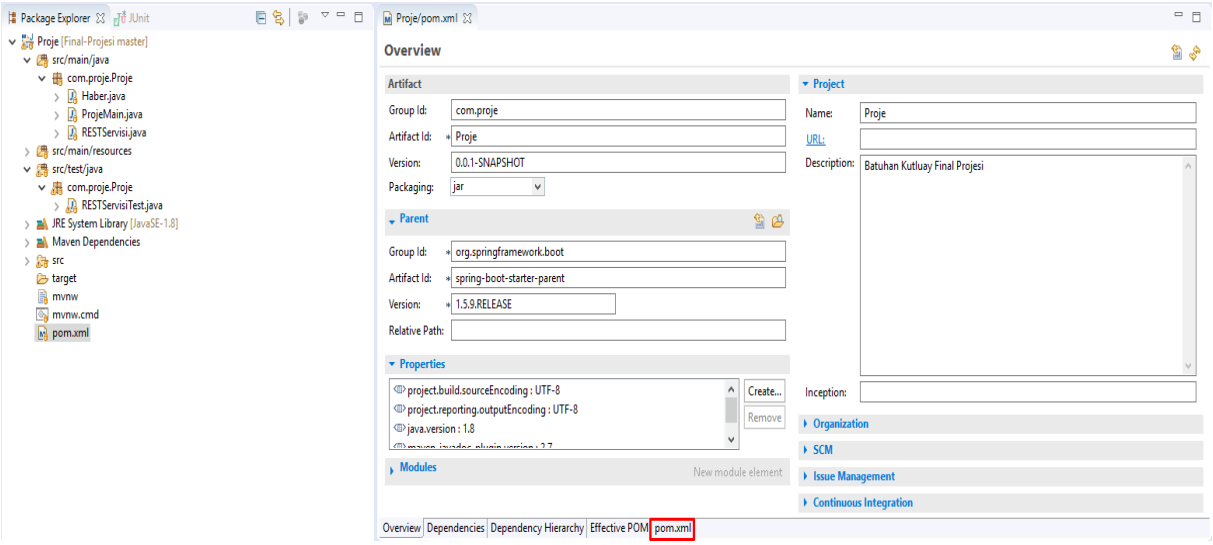
1. "Find" kısmına "eclemma" yazıyoruz.
2. "EclEmma Java Code Coverage" karşımıza gelecek.
3. Buradan yükleme işlemi yapıyoruz.Ben daha önceden yüklediğim için "Installed" seçeneği mevcut.



Figur 22

Figur 23 İçin Adımlar

1. “Pacage Explorer”dan “pom.xml” dosyasını açıyoruz.
2. Açılan sayfada aşağıdaki sekmelerden “pom.xml” seçeneğini seçiyoruz.



Figur 23

Figur 24 İçin Adımlar

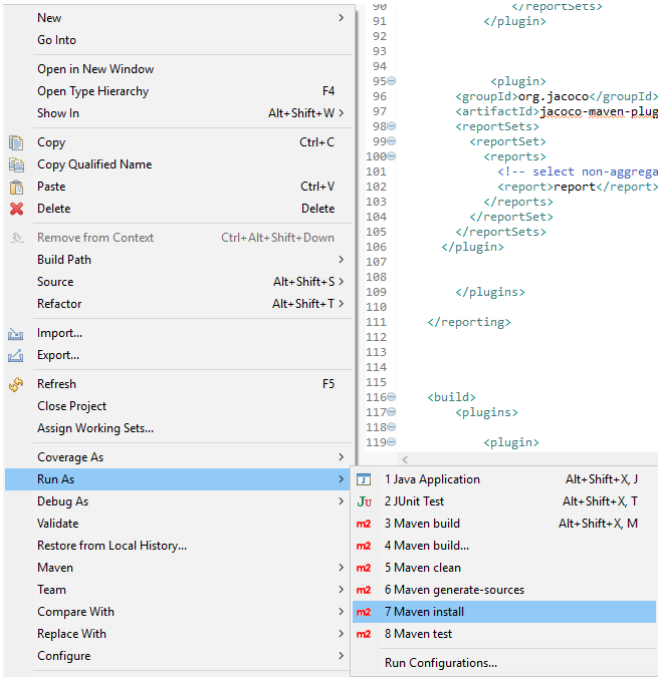
1. “<build> <plugins> …. </build> </plugins>” kısmına “Figur 22”yi yazıyoruz.

```
        <plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.6.201602180812</version>
    <executions>
        <execution>
            <id>jacoco-initialize</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>jacoco-site</id>
            <phase>package</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Figur 24

Figur 25 İçin Adımlar

1. Projemize “Sağ Click” yapıyoruz.
2. “Run As” seçeneğini seçiyoruz.
3. “Maven install” diyoruz.
4. “Figur 26” da console output gözükmekte.Projemizin 7 test çalıştırdığını ve hiç bir sıkıntı olmadığı için “BUILD SUCCES” yazısını görebiliyoruz.



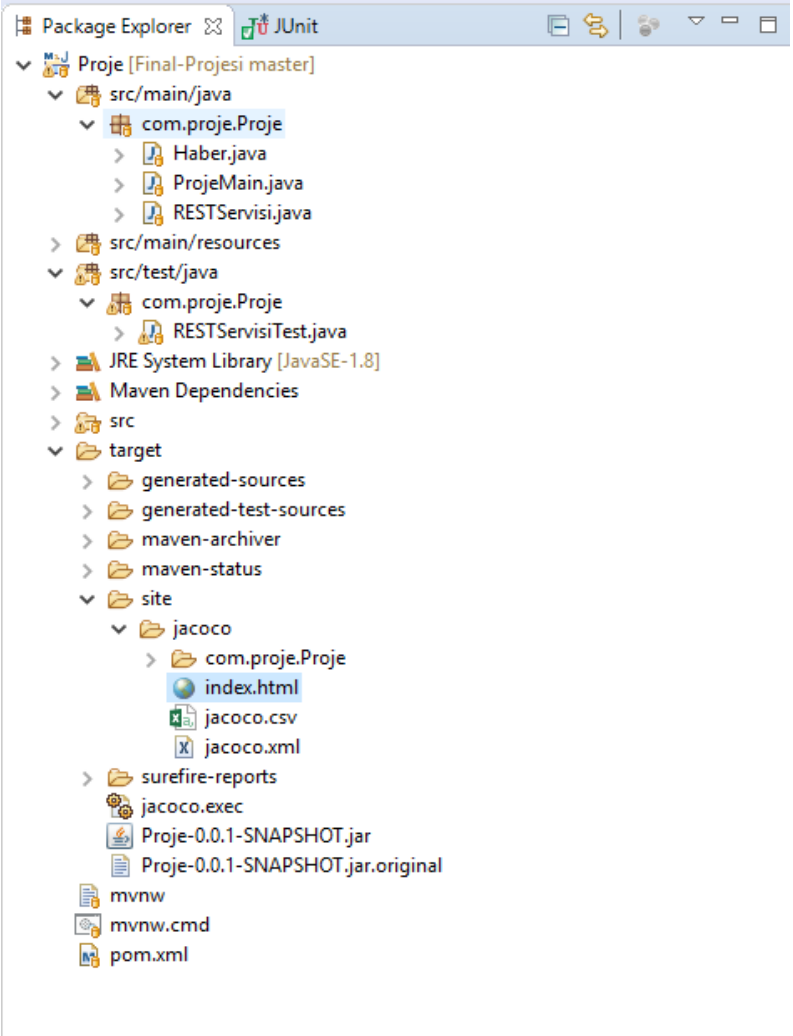
Figur 25

```
Results :

Tests run: 7, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.6:jar (default-jar) @ Proje ---
[INFO] Building jar: C:\Users\KUTLUAY\git\Final-Projesi\Proje\target\Proje-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:1.5.9.RELEASE:repackage (default) @ Proje ---
[INFO]
[INFO] --- jacoco-maven-plugin:0.7.6.201602180812:report (jacoco-site) @ Proje ---
[INFO] Analyzed bundle 'Proje' with 3 classes
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ Proje ---
[INFO] Installing C:\Users\KUTLUAY\git\Final-Projesi\Proje\target\Proje-0.0.1-SNAPSHOT.jar to C:\Users\KUTLUAY\.m2\repository\com\proje\Proje\0.0.1-SNAPSHOT\Proje-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\KUTLUAY\git\Final-Projesi\Proje\pom.xml to C:\Users\KUTLUAY\.m2\repository\com\proje\Proje\0.0.1-SNAPSHOT\Proje-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.109 s
[INFO] Finished at: 2018-01-24T17:02:44+02:00
[INFO] Final Memory: 36M/270M
[INFO] ------------------------------------------------------------------------
```

Figur 26

Figur 27 İçin Adımlar

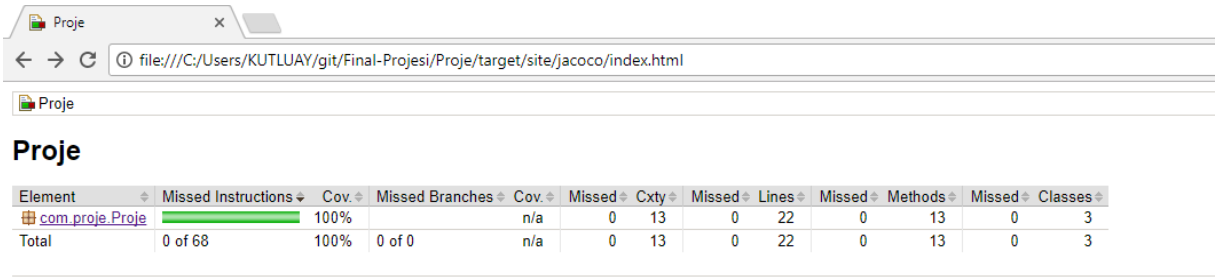
1. “Package Explorer” kısmında projemizin “target” dosyasını açıyoruz.
2. “site” adında bir klasör oluşacak.Onuda açıyoruz
3. “jacoco” dosyası gözükecek.Bu dosyayı da açtığımız zaman “index.html”e ulaşıyoruz.
4. “index.html”: jacoco test ettiği testleri bir site haline getiriyor.Bu olayın gerçekleşmesini “pom.xml”e yazdığımız jacoco plugini gerçekleştirdi.
5. “index.html”i açarsak karşımıza “Figur 28” gelir.



Figur 27

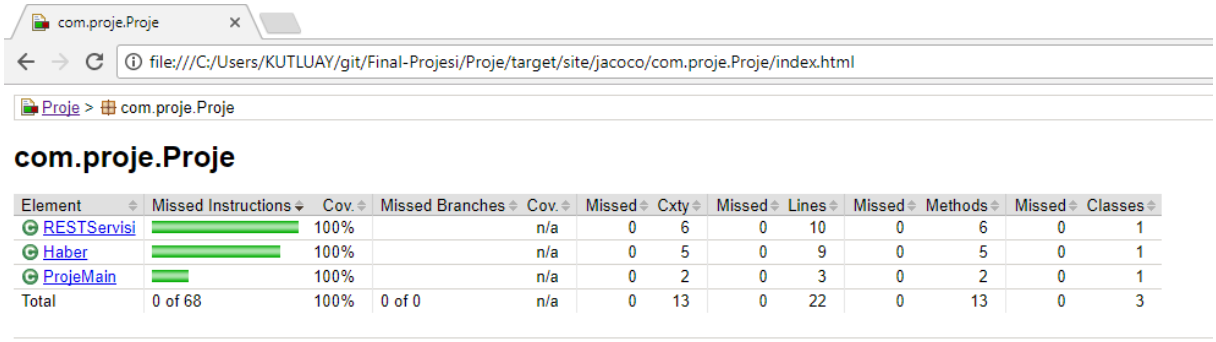
Figur 28 İçin Adımlar

1. “com.proje.Proje” yani gözüken package ismine tıklarsak karşımıza “Figur 29” gelir.



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.proje.Proje		100%		n/a	0	13	0	22	0	13	0	3
Total	0 of 68	100%	0 of 0	n/a	0	13	0	22	0	13	0	3

Figur 28



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
RESTServisi		100%		n/a	0	6	0	10	0	6	0	1
Haber		100%		n/a	0	5	0	9	0	5	0	1
ProjeMain		100%		n/a	0	2	0	3	0	2	0	1
Total	0 of 68	100%	0 of 0	n/a	0	13	0	22	0	13	0	3

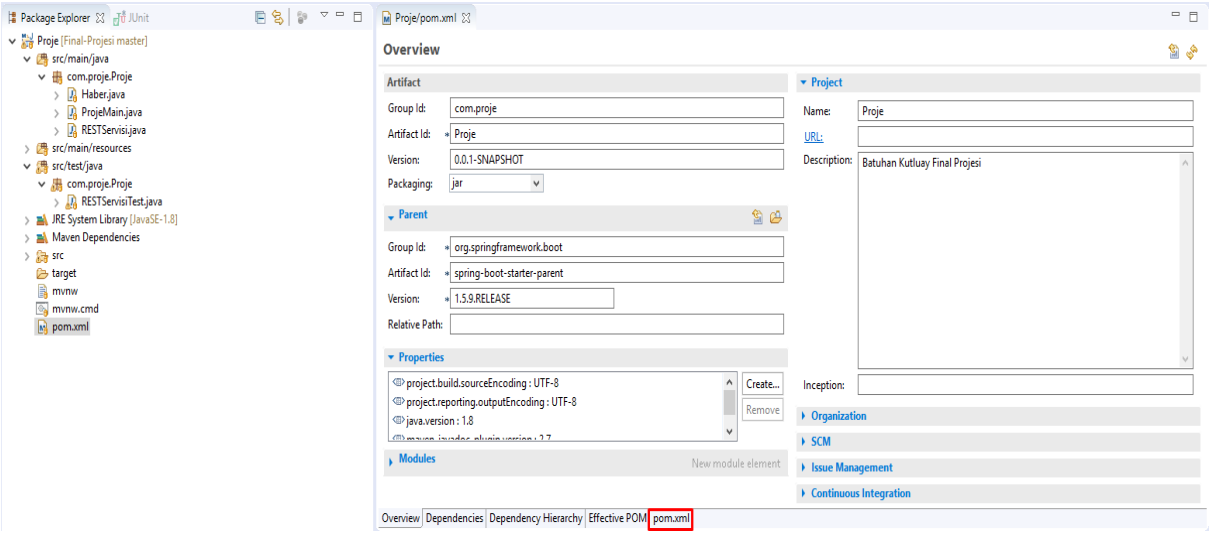
Figur 29

7. COBERTURA CODE COVERAGE

Cobertura, kodumuzun ne kadarının test edildiğini bize gösterir.Coberturayı projemize uygulayabilmek için gerekli adımlar vardır.

Figur 30 İçin Adımlar

1. “Pacage Explorer”dan “pom.xml” dosyasını açıyoruz.
2. Açılan sayfada aşağıdaki sekmelerden “pom.xml” seçeneğini seçiyoruz.



Figur 30

Figur 31 İçin Adımlar

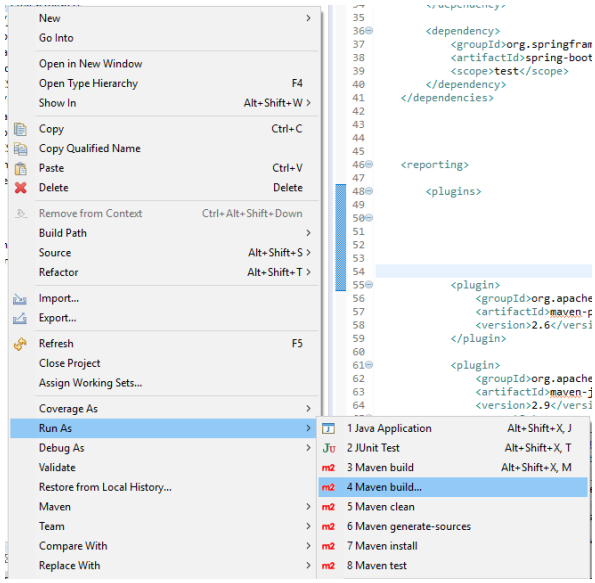
1. “<build> <plugins> …. </build> </plugins>” kısmına “Figur 31”i yazıyoruz.

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>cobertura-maven-plugin</artifactId>
  <version>2.7</version>
  <configuration>
      <formats>
          <format>html</format>
          <format>xml</format>
      </formats>
      <check />
  </configuration>
</plugin>
```

Figur 31

Figur 32 İçin Adımlar

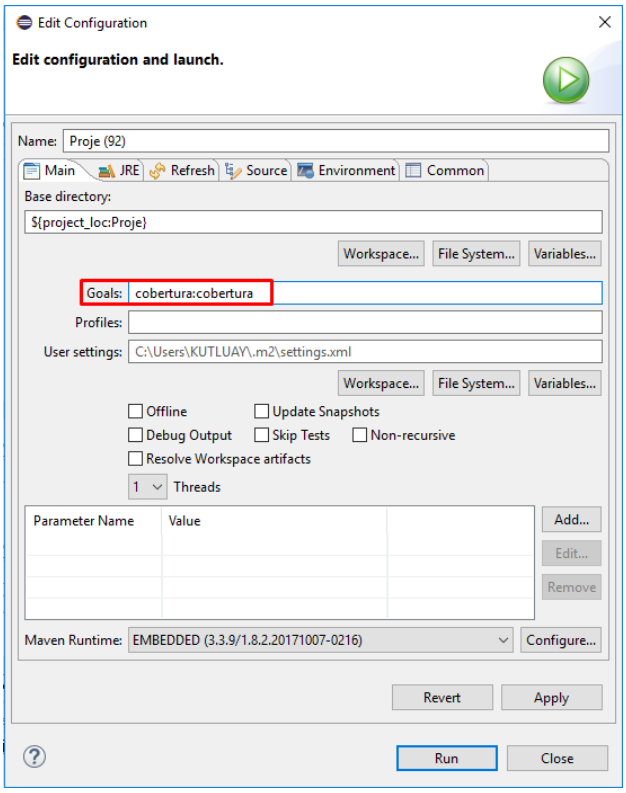
1. Projemize Sağ Click yapıyoruz.
2. “Run As” seçeneğini seçiyoruz.
3. “Maven buid...” seçeneğini seçtiğimiz zaman karşımıza “Figur 33” gelecektir.



Figur 32

Figur 33 İçin Adımlar

1. Maven Goal olarak “cobertura:cobertura” yazıyoruz.
2. “Run As” seçeneğini seçiyoruz.
3. “Figur 34” de console output gözükmekte.Projemizin 7 test çalıştırdığını ve hiç bir sıkıntı olmadığı için “BUILD SUCCES” yazısını görebiliyoruz.



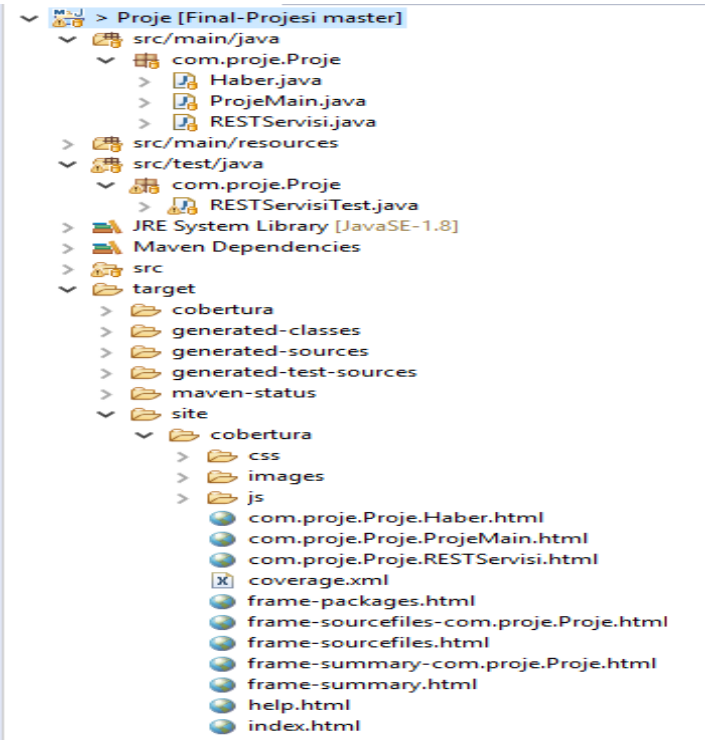
Figur 33

```
Problems  @ Javadoc  Declaration  Console  Git Staging  Coverage
<terminated> Proje (20) [Maven Build] C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (Jan 24, 2018, 6:41:11 PM)

Results :

Tests run: 7, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] <<< cobertura-maven-plugin:2.7:cobertura (default-cli) < [cobertura]test @ Proje <<<
[INFO]
[INFO] --- cobertura-maven-plugin:2.7:cobertura (default-cli) @ Proje ---
[INFO] Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
[INFO] Cobertura: Loaded information on 3 classes.
Report time: 104ms

[INFO] Cobertura Report generation was successful.
[INFO] Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
[INFO] Cobertura: Loaded information on 3 classes.
Report time: 97ms

[INFO] Cobertura Report generation was successful.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.360 s
[INFO] Finished at: 2018-01-24T18:41:20+02:00
[INFO] Final Memory: 33M/275M
[INFO] ------------------------------------------------------------------------
```

Figur 34

Figur 35 İçin Adımlar

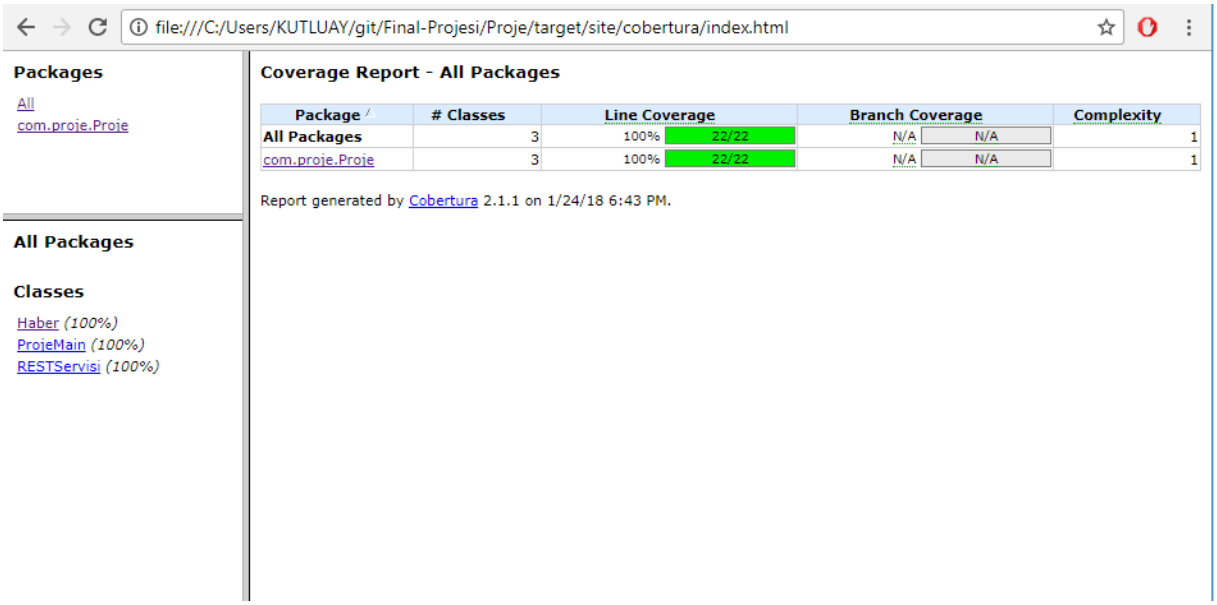
1. “Package Explorer” kısmında projemizin “target” dosyasını açıyoruz.
2. “site” adında bir klasör oluşacak.Onuda açıyoruz
3. “cobertura” dosyası gözükecek.Bu dosyayı da açtığımız zaman “index.html”e ulaşıyoruz.
4. “index.html”: cobertura test ettiği testleri bir site haline getiriyor.Bu olayın gerçekleşmesini “pom.xml”e yazdığımız cobertura plugini gerçekleştirdi.
5. “index.html”i açarsak karşımıza “Figur 36” gelir.



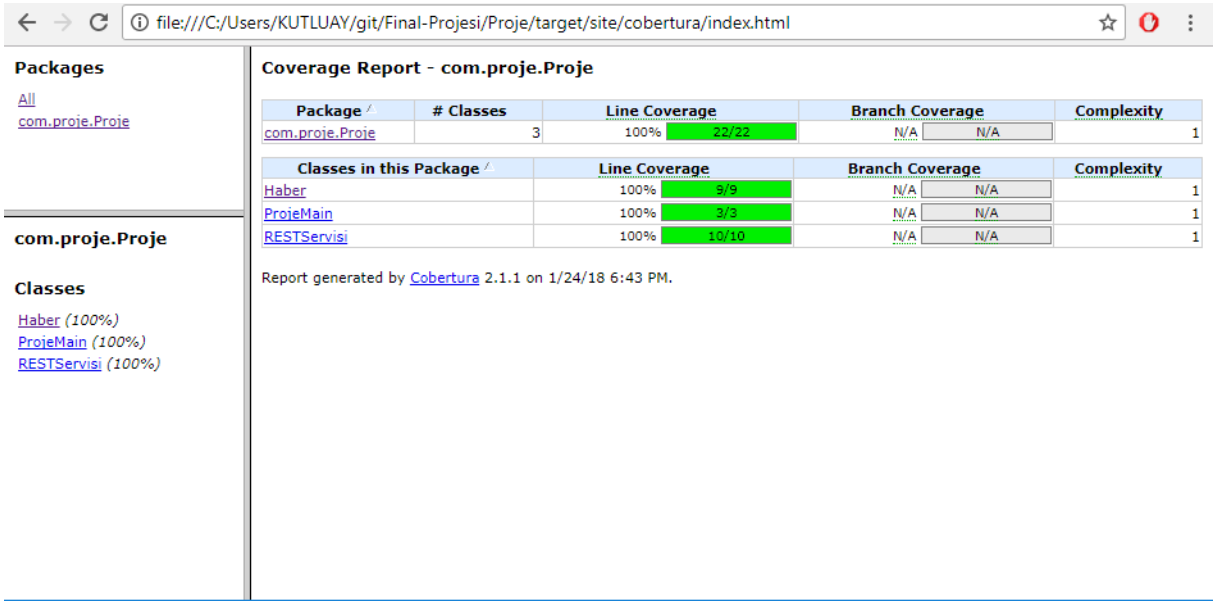
Figur 35

Figur 36 İçin Adımlar

1. “com.proje.Proje” yani gözüken package ismine tıklarsak karşımıza “Figur 37” gelir.



Figur 36



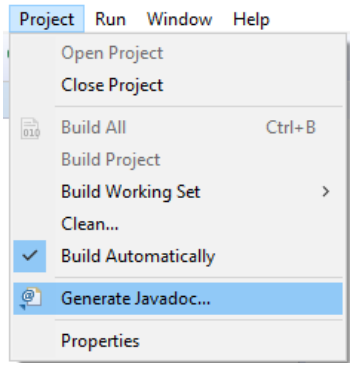
Figur 37

8. JAVADOC

Javadoc dökümantasyon işlemleri için geçerlidir.Javadoc oluşturmak için gerekli adımlar vardır.

Figur 38 İçin Adımlar

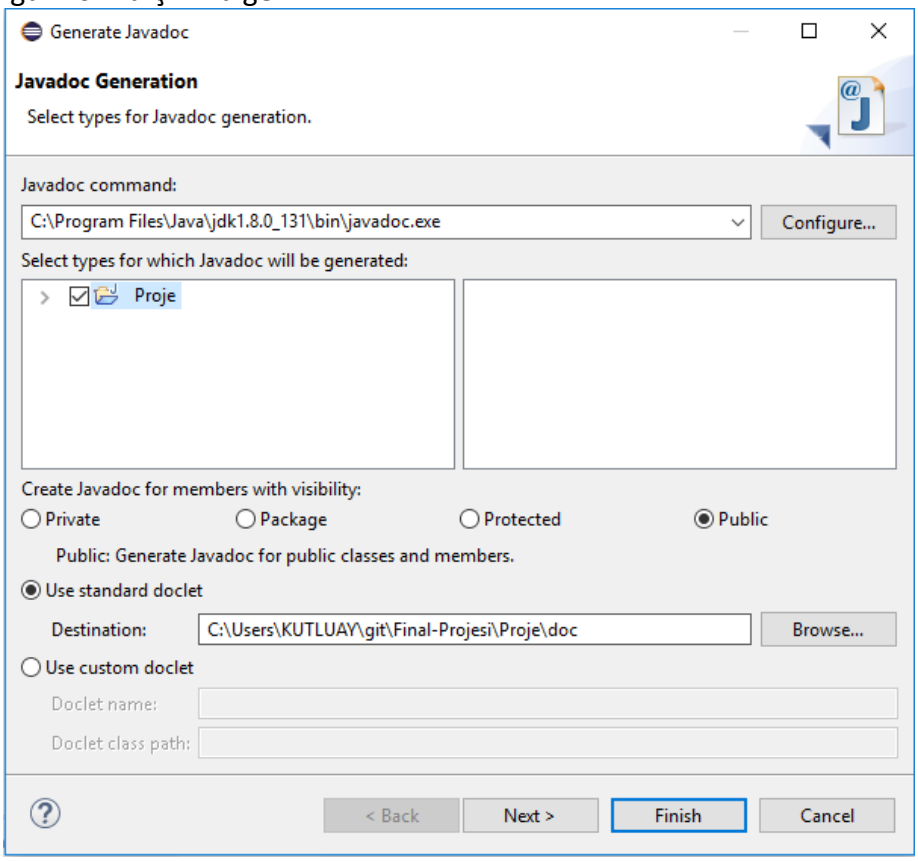
1. Toolbardan “Project” seçeneği seçilir.
2. “Generate Javadoc..” seçilir.
3. “Figur 39” karşımıza gelir.



Figur 38

Figur 39 İçin Adımlar

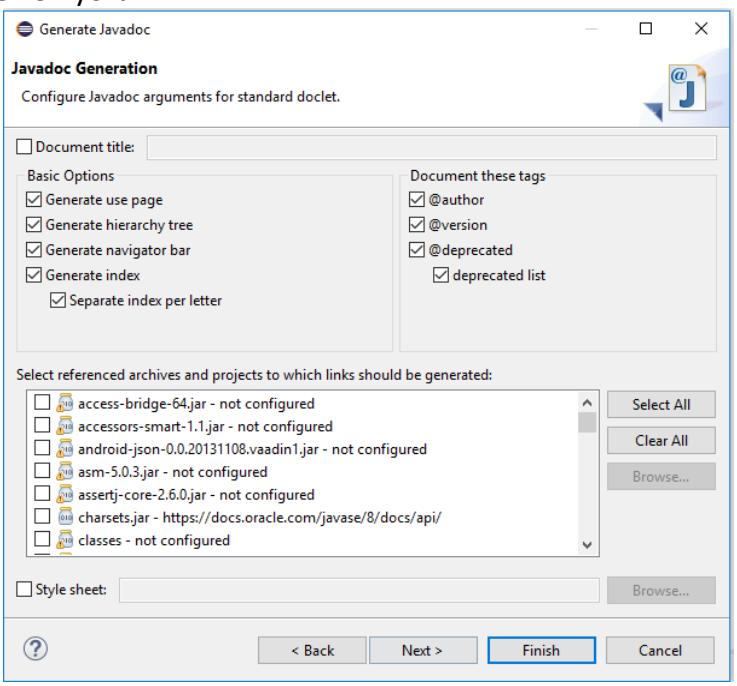
1. “Javadoc Command” olarak sistemimizde kurulu javanın “jdk\bin\javadoc.exe” yolu seçilir.
2. Projemizi seçip “Next” diyoruz.
3. “Figur 40” karşımıza gelir.



Figur 39

Figur 40 İçin Adımlar

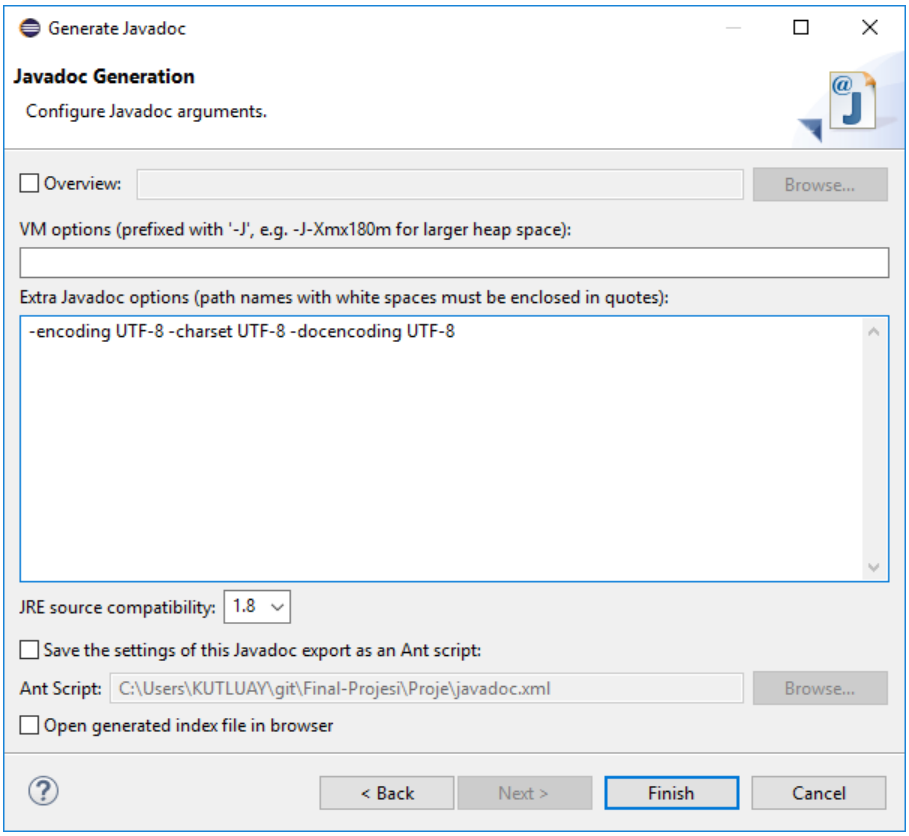
1. “Next” ile ilerliyoruz.



Figur 40

Figur 40 İçin Adımlar

1. Dökümantasyonumuzun “UTF-8” şeklinde oluşturulmasını istediğimiz için “Extra Javadoc options” kısmına aşağıda encodingi yazıyoruz.



Figur 41

Figur 42 İçin Adımlar

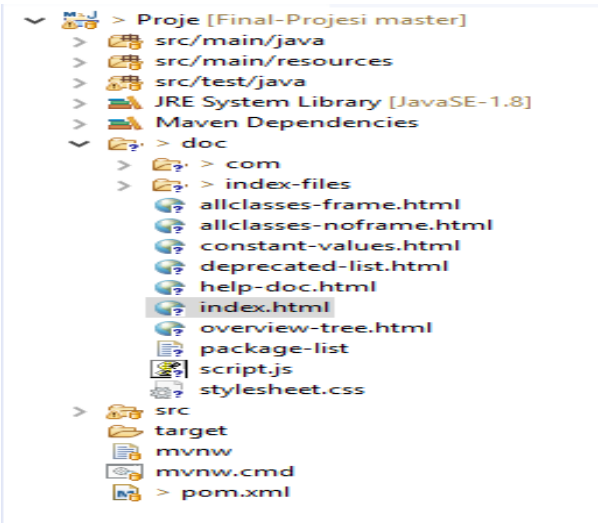
1. Konsolumuzda generate işleminin bitmesini bekliyoruz.

```
<terminated> Javadoc Generation
Loading source files for package com.proje.Proje...
Constructing Javadoc information...
Standard Doclet version 1.8.0_131
Building tree for all the packages and classes...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\Haber.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\ProjeMain.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\RESTServisi.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\RESTServisiTest.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\package-frame.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\package-summary.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\package-tree.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\constant-values.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\class-use\RESTServisiTest.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\class-use\RESTServisi.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\class-use\ProjeMain.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\class-use\Haber.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\com\proje\Proje\package-use.html...
Building index for all the packages and classes...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\overview-tree.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-1.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-2.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-3.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-4.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-5.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-6.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index-files\index-7.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\deprecated-list.html...
Building index for all classes...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\allclasses-frame.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\allclasses-noframe.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\index.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\doc\help-doc.html...
```

Figur 42

Figur 43 İçin Adımlar

1. Generate işlemi bittikten sonra otomatik olarak “Figur 43”te görünen “doc” dosyası oluşacaktır.
2. “doc” dosyasına giriyoruz.
3. “index.html” dosyasını açarak oluşan dökümantasyona bakabiliriz.Html dosyasini açarsak karşımıza “Figur 44” gelir.



Figur 43

file:///C:/Users/KUTLUAY/git/Final-Projesi/Proje/doc/index.html?com/proje/Proje/package-summary.html

All Classes

Haber
ProjeMain
RESTServisi
RESTServisiTest

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Package com.proje.Proje

Class Summary

Class	Description
Haber	Haber işlemlerinden sorumlu sınıf.
ProjeMain	Main
RESTServisi	Rest Servisi işlemlerinden sorumlu sınıf
RESTServisiTest	Test işlemlerinden sorumlu sınıf.

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Figur 44

Haber Sınıfı

com.proje.Proje

Class Haber

java.lang.Object
com.proje.Proje.Haber

```
public class Haber
extends java.lang.Object
```

Haber işlemlerinden sorumlu sınıf.

Author:
Batuhan Kutluay

Constructor Summary

Constructors

Constructor and Description
Haber(java.lang.String haberBaslik, java.lang.String haberIcerik)

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
java.lang.String	getHaberBaslik() Mevcut Haber Başlığını return eder.
java.lang.String	getHaberIcerik() Mevcut Haber İçeriğini return eder.
int	getId() Mevcut id'yi return eder.

Figur 45

Constructor Detail

Haber

```
public Haber(java.lang.String haberBaslik,
             java.lang.String haberIcerik)
```

Parameters:

`haberBaslik - RESTServisi sınıfından oluşturulan haber başlığı burada alınır.`

`haberIcerik - RESTServisi sınıfından oluşturulan haber içeriği burada alınır.`

Method Detail

getHaberBaslik

```
public java.lang.String getHaberBaslik()
```

Mevcut Haber Başlığını return eder.

Returns:

`Haberin Başlığı`

getHaberIcerik

```
public java.lang.String getHaberIcerik()
```

Mevcut Haber İçeriğini return eder.

Returns:

`Haberin İçeriği`

getId

```
public int getId()
```

Mevcut id'yi return eder.

Returns:

`Haberin id'si`

Figur 46

RestServisi Sınıfı

com.proje.Proje

Class RESTServisi

java.lang.Object
com.proje.Proje.RESTServisi

```
@RestController
public class RESTServisi
extends java.lang.Object
```

Rest Servisi işlemlerinden sorumlu sınıf

Author:
Batuhan Kutluay

Constructor Summary

Constructors

Constructor and Description
`RESTServisi()`

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`static java.util.List<Haber>`	`haberListele()` Oluşan haberleri listelemek için kullanılır.
`static java.lang.String`	`haberSil(int index)` Verilen indexteki haberi silmek için kullanılır.
`static java.lang.String`	`haberTemizle()` Oluşturulan bütün haberleri temizlemek için kullanılır.
`static Haber`	`haberYAZ(java.lang.String haberbasligi, java.lang.String habericerigi)` Haber oluşturmak için kullanılır.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Figur 47

Constructor Detail

RESTServisi

```
public RESTServisi()
```

Method Detail

haberYAZ

```
@RequestMapping(value="/haber/olustur")
public static Haber haberYAZ(java.lang.String haberbasligi,
                                                                    java.lang.String habericerigi)
```

Haber oluşturmak için kullanılır.

Parameters:

haberbasligi - Kullanıcıdan gelen haber başlığı.

habericerigi - Kullanıcıdan gelen haber içeriği.

Returns:

Oluşan Haber.

haberListele

```
@RequestMapping(value="/haber/listele")
public static java.util.List<Haber> haberListele()
```

Oluşan haberleri listelemek için kullanılır.

Returns:

Oluşan bütün haberler.

haberTemizle

```
@RequestMapping(value="/haber/temizle")
public static java.lang.String haberTemizle()
```

Oluşturulan bütün haberleri temizlemek için kullanılır.

Returns:

Haberlerin temizlendiğine dair kullanıcıya mesaj

haberSil

```
@RequestMapping(value="/haber/sil")
public static java.lang.String haberSil(int index)
```

Verilen indexteki haberi silmek için kullanılır.

Parameters:

index - Kullacının verdiği index değeri

Returns:

Verilen indexteki haberin silidindiğie dair kullanıcıya mesaj.

Figur 48

RESTServisiTest Sınıfı

com.proje.Proje

Class RESTServisiTest

java.lang.Object
com.proje.Proje.RESTServisiTest

```
@SpringBootTest
 @JsonTest
public class RESTServisiTest
extends java.lang.Object
```

Test işlemlerinden sorumlu sınıf.

Author:

Batuhan Kutluay

Constructor Summary

Constructors

Constructor and Description
`RESTServisiTest()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	`testClass()` RESTServisi sınıfını test eder.
void	`testHaber()` Haber sınıfını test etmek için yazılmış bir testtir.
void	`testHaberListele()` RESTServisinde bulunan haberListele metotunu test eder.
void	`testHaberSil()` RESTServisinde bulunan haberSil metotunu test eder.
void	`testHaberTemizle()` RESTServisinde bulunan haberTemizle metotunu test eder.
void	`testHaberYaz()` RESTServisinde bulunan haberYAZ metotunu test eder.
void	`testMain()` Maini test eder

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Figur 49

Constructor Detail

RESTServisiTest

```
public RESTServisiTest()
```

Method Detail

testHaber

```
public void testHaber()
                throws java.lang.Exception
```

Haber sınıfını test etmek için yazılmış bir testtir.

Throws:

`java.lang.Exception` - Hata Fırlatır.

testHaberYaz

```
public void testHaberYaz()
                  throws java.lang.Exception
```

RESTServisinde bulunan haberYAZ metotunu test eder.

Throws:

`java.lang.Exception` - Hata fırlatır.

testHaberListele

```
public void testHaberListele()
                      throws java.lang.Exception
```

RESTServisinde bulunan haberListele metotunu test eder.

Throws:

`java.lang.Exception` - Hata fırlatır.

Figur 50

testHaberTemizle

```
public void testHaberTemizle()
                        throws java.lang.Exception
```

RESTServisinde bulunan haberTemizle metotunu test eder.

Throws:

`java.lang.Exception - Hata fırlatır.`

testHaberSil

```
public void testHaberSil()
                    throws java.lang.Exception
```

RESTServisinde bulunan haberSil metotunu test eder.

Throws:

`java.lang.Exception - Hata fırlatır.`

testClass

```
public void testClass()
                  throws java.lang.Exception
```

RESTServisi sınıfını test eder.

Throws:

`java.lang.Exception - Hata Fırlatır.`

testMain

```
public void testMain()
```

Maini test eder

Figur 51

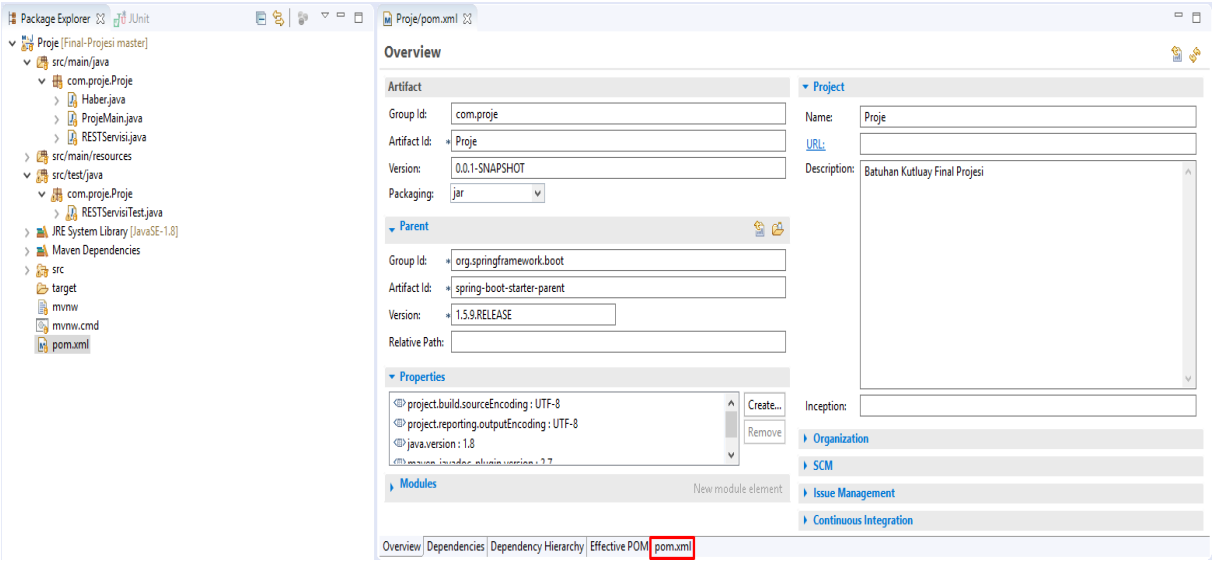
9. MAVEN SITE OLUŞTURMAK

Maven Site plugini projemizin sitesini oluşturur.Ayrıca oluşturulan sitede “pom.xml”e tanımladığımız proje raporlarını içinde barındırır.Bunları gerçekleştirmek için belirli adımlar vardır.

Maven Site’da JaCoCo Raporu İçin Gereken Adımlar

Figur 52 İçin Adımlar

1. “Pacage Explorer”dan “pom.xml” dosyasını açıyoruz.
2. Açılan sayfada aşağıdaki sekmelerden “pom.xml” seçeneğini seçiyoruz.



Figur 52

Figur 53 İçin Adımlar

1. “<reporting> <plugins> …. </plugins> </reporting>” kısmını oluştuyoruz ve oraya “Figur 53”ü yazıyoruz.

```
<plugin>
   <groupId>org.jacoco</groupId>
   <artifactId>jacoco-maven-plugin</artifactId>
   <reportSets>
     <reportSet>
       <reports>
         <!-- select non-aggregate reports -->
         <report>report</report>
       </reports>
     </reportSet>
   </reportSets>
</plugin>
```

Figur 53

Maven Site'da Cobertura Raporu İçin Gereken Adımlar
Figur 54 İçin Adımlar

1. Oluşturduğumuz "<reporting> <plugins> …. </plugins> </reporting>" kısmına "Figur 54"ü yazıyoruz.

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>cobertura-maven-plugin</artifactId>
    <version>2.7</version>
    <configuration>
        <formats>
            <format>html</format>
            <format>xml</format>
        </formats>
        <check />
    </configuration>
</plugin>
```

Figur 54

Maven Site'da Javadoc Raporu İçin Gereken Adımlar
Figur 55 İçin Adımlar

1. Oluşturduğumuz "<reporting> <plugins> …. </plugins> </reporting>" kısmına "Figur 55"ü yazıyoruz.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>2.9</version>
    <reportSets>
        <reportSet><!-- by default, id = "default" -->
            <reports><!-- select non-aggregate reports -->
                <report>javadoc</report>
                <report>test-javadoc</report>
            </reports>
        </reportSet>
        <reportSet><!-- aggregate reportSet, to define in poms having modules -->
            <id>aggregate</id>
            <inherited>false</inherited><!-- don't run aggregate in child modules -->
            <reports>
                <report>aggregate</report>
            </reports>
        </reportSet>
    </reportSets>
</plugin>
```

Figur 55

Maven Site'da Projemiz Hakkında Rapor Oluşması İçin Gereken Adımlar
Figur 56 İçin Adımlar

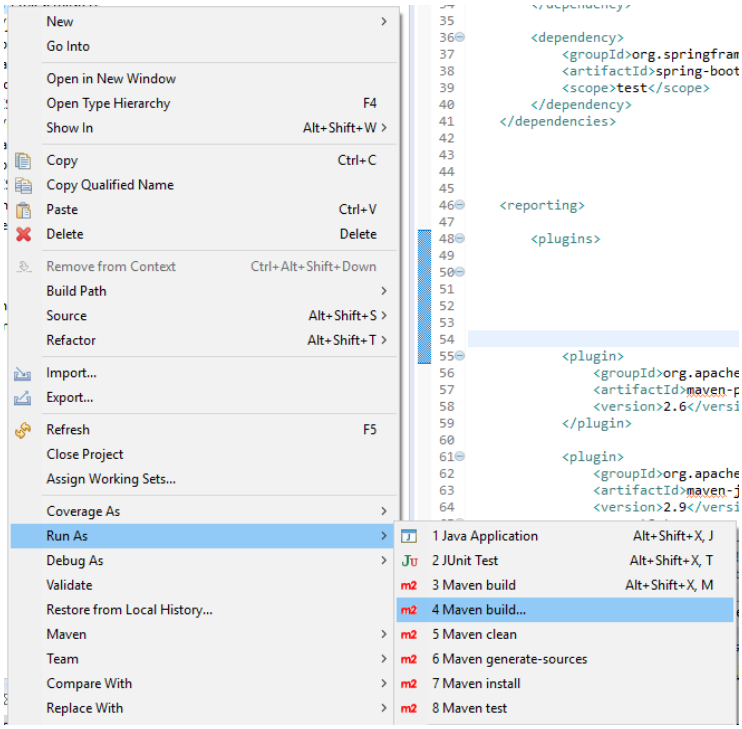
1. Oluşturduğumuz "<reporting> <plugins> …. </plugins> </reporting>" kısmına "Figur 56"yı yazıyoruz.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-project-info-reports-plugin</artifactId>
    <version>2.6</version>
</plugin>
```

Figur 56

Maven Site'da Oluşturmak İçin Gereken Adımlar
Figur 57 İçin Adımlar

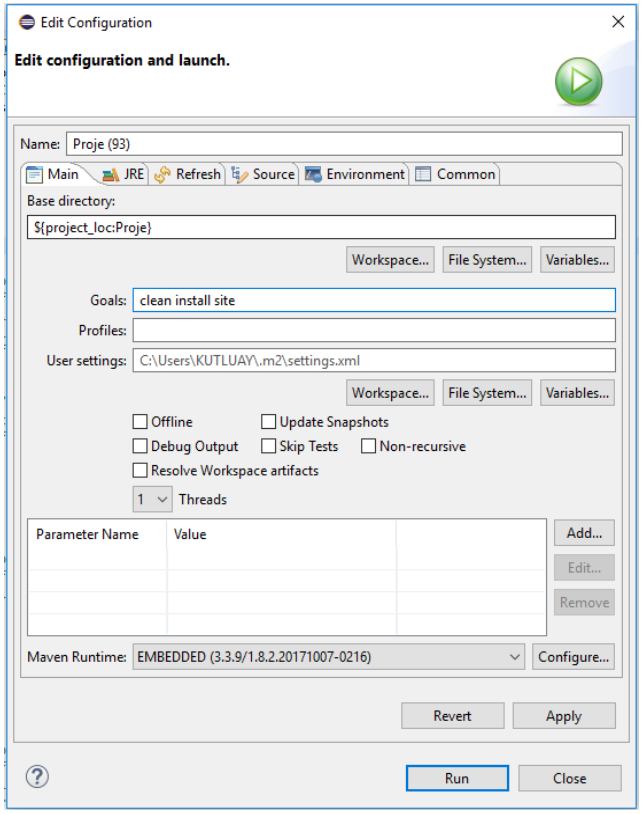
1. Projemize Sağ Click yapıyoruz.
2. "Run As" seçeneğini seçiyoruz.
3. "Maven buid..." seçeneğini seçtiğimiz zaman karşımıza "Figur 58" gelecektir.



Figur 57

Figur 58-59 İçin Adımlar

1. Maven Goal olarak “clean install site” yapıyoruz.
2. “Run” seçeneğini seçiyoruz.
3. Konsolda siteyi oluşturma işlemleri başlıyacaktır.
4. İşlemler bittikten sonra “Figur 59”daki gibi “BUILD SUCCESS” yazısı gelecektir.



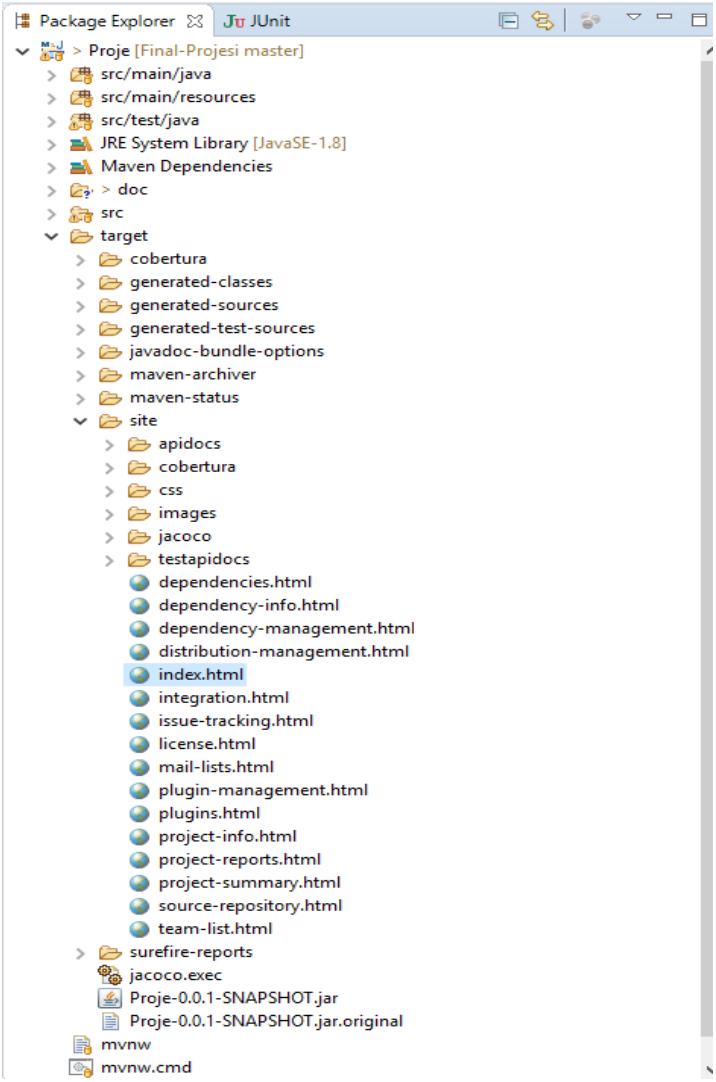
Figur 58

```
Building index for all classes...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\target\site\testapidocs\allclasses-frame.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\target\site\testapidocs\allclasses-noframe.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\target\site\testapidocs\index.html...
Generating C:\Users\KUTLUAY\git\Final-Projesi\Proje\target\site\testapidocs\help-doc.html...
[INFO] Generating "JaCoCo Test" report          --- jacoco-maven-plugin:0.7.6.201602180812:report
[INFO] Analyzed bundle 'Proje' with 3 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 32.114 s
[INFO] Finished at: 2018-01-24T23:53:04+02:00
[INFO] Final Memory: 58M/792M
[INFO] ------------------------------------------------------------------------
```

Figur 59

Figur 60 İçin Adımlar

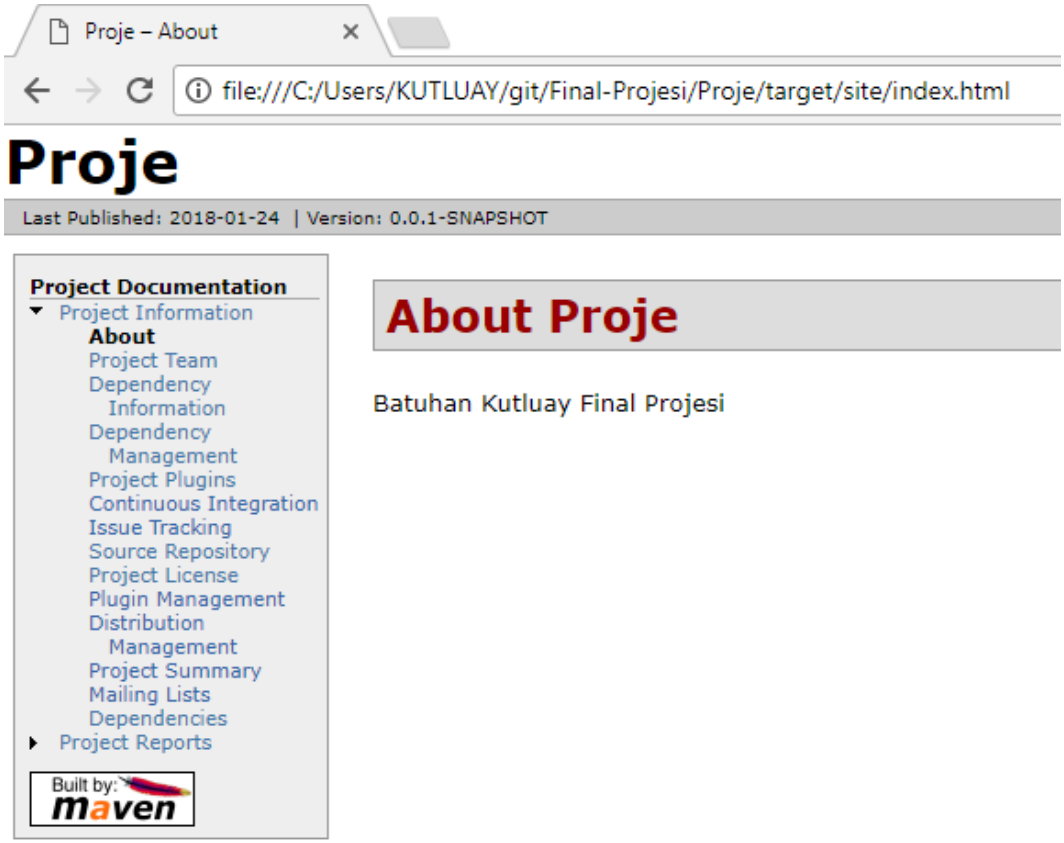
1. Oluşturduğumuz siteyi görebilmek için projeyi yenilemek yeterli olacaktır.
2. Daha sonra “target” dosyasını açıyoruz.
3. “site” dosyasını da açıyoruz
4. Oradaki “index.html” bizim oluşturduğumuz Saven Site’dır.
5. “index.html”i açtığımız zaman karşımıza “Figur 61” gelir.



Figur 60

Figur 61 İçin Adımlar

1. Soldaki menüden “Project Reports” seçeneği seçilirse oluşan raporlar gözükecektir.
2. Seçildiği taktirde “Figur 62” ekrana gelir.



Figur 61

Figur 62

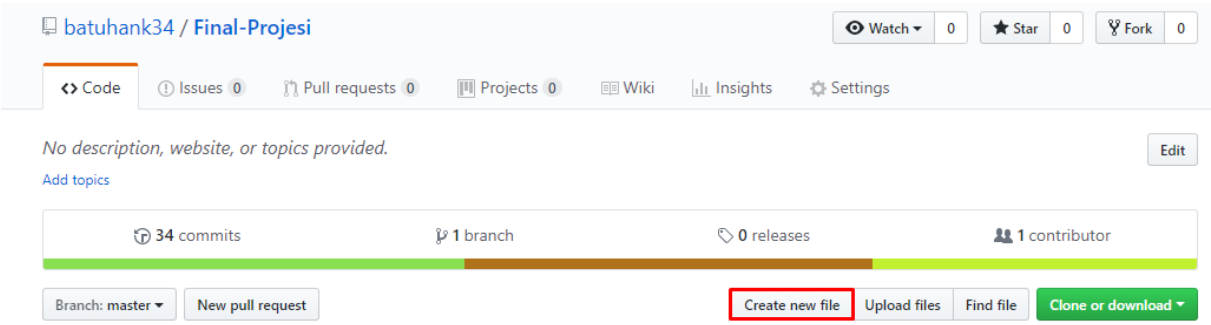
10. TRAVIS CI & CODECOV IO

Travis, projemizde yapılan her commitin ardından tüm sistemin çalışır durumda olduğunu otomatik olarak kontrol eder.Codecov ise kodumuzun kalitesini bize söyler.Yazdığımız testlere bakar.

Travis ve Codecov'da ki adımlar bittikten sonra git sonucunu koyacağım.Şuan da hiç bir işlem olmadığı için gözükmüyor.

Repositorydemizde Travis ve CodeCov Sonuçlarını Göstermek İçin Adımlar

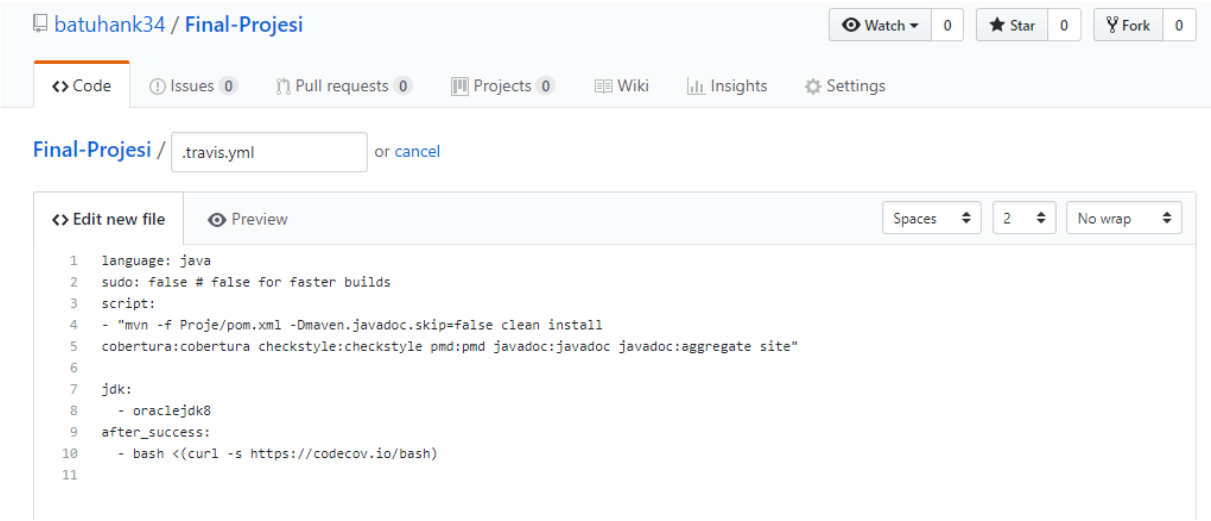
1. Oluşturduğumuz repomuza gidiyoruz.
2. "Create new file" seçeneğini seçiyoruz.
3. "Figur 64" karşımıza gelecektir.



Figur 63

Figur 64 İçin Adımlar

1. Dosya ismine ".travis.yml" adını veriyoruz.
2. Dosyanın içine ise aşağıdaki gibi kod satırlarımızı yazıyoruz.



Figur 64

10.1. TRAVIS CI

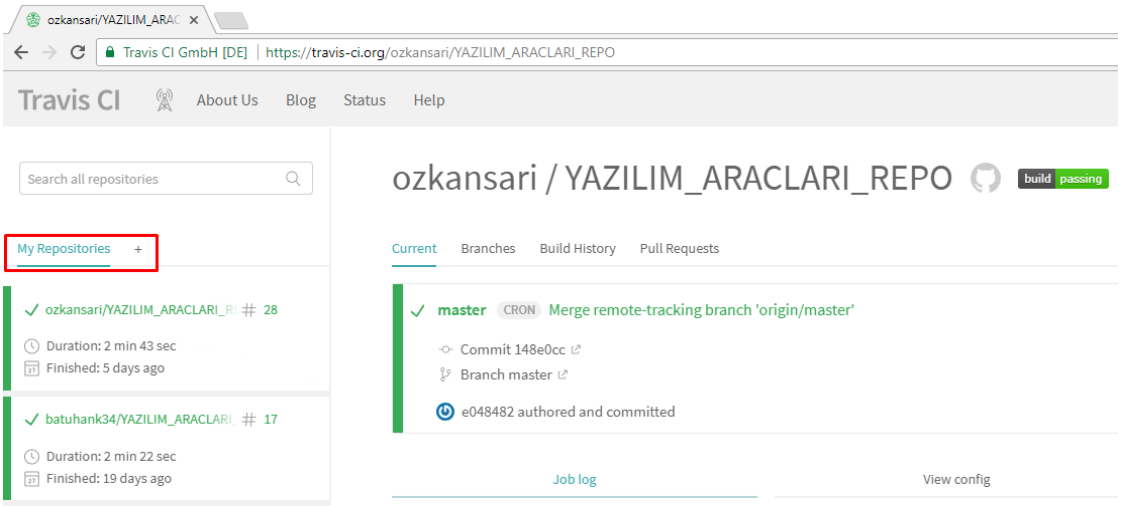
Figur 65 İçin Adımlar

1. https://travis-ci.org/ sitesine giriyoruz.
2. “Sign in with GitHub” butonuna tıklıyoruz.
3. Karşımıza “Figur 66” gelecektir.

Figur 65

Figur 66 İçin Adımlar

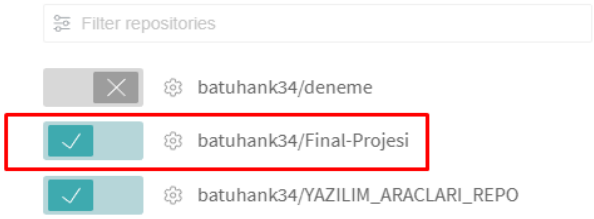
1. Sol tafata bulunan “+”ya tıklıyoruz
2. Karşımıza “Figur 77” gelecektir.



Figur 66

Figur 67 İçin Adımlar

1. Mevcut repositorylerimizin gözüktüğü bir ekran gelecektir.
2. Proje için açtığımız repomuzu yani “Final-Projesi” reposunu aktif hala getiriyorum.

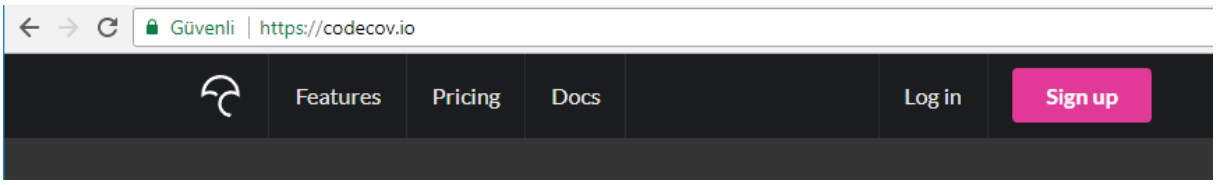


Figur 67

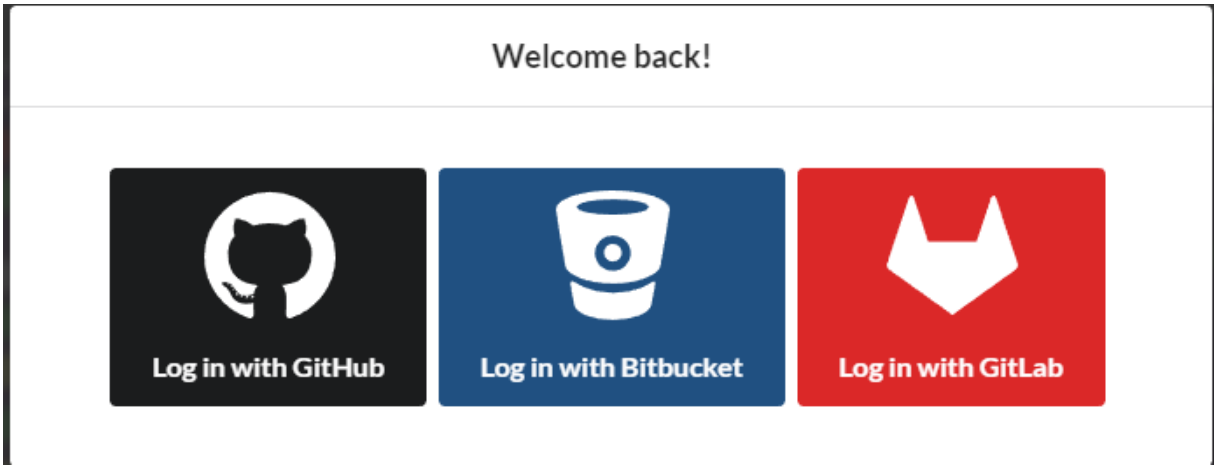
10.2. CODECOV IO

Figur 68 - 69 İçin Adımlar

1. https://codecov.io sitesine giriyoruz.
2. “Log in” seçeneğini seçiyoruz
3. “Log in with “GitHub” seçeneğini seçiyoruz
4. Karşımıza “Figur 70” gelecektir.



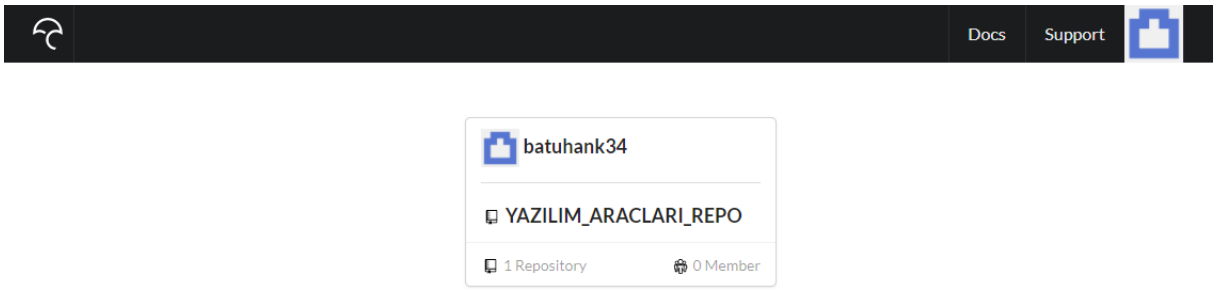
Figur 68



Figur 69

Figur 70 İçin Adımlar

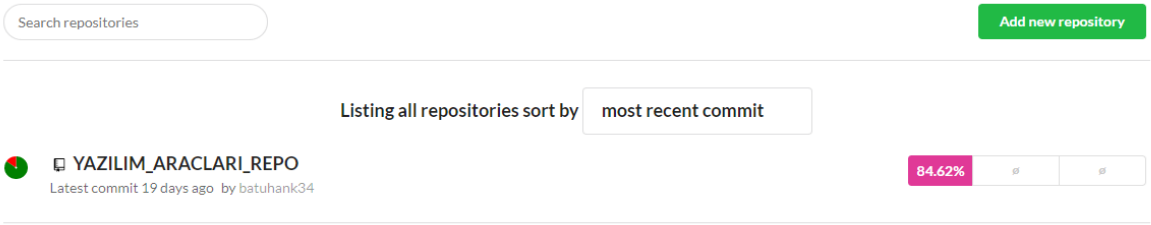
1. GitHub profil ismimize tıklıyoruz.



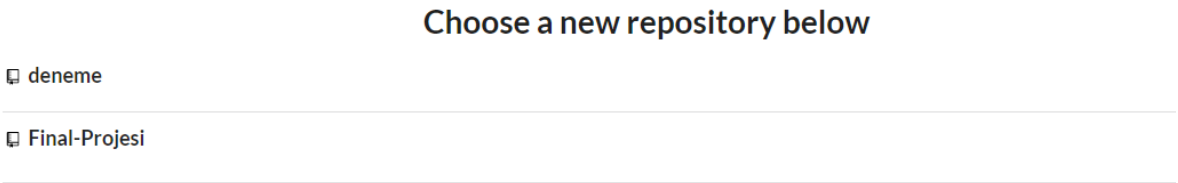
Figur 70

Figur 71-72-73 İçin Adımlar

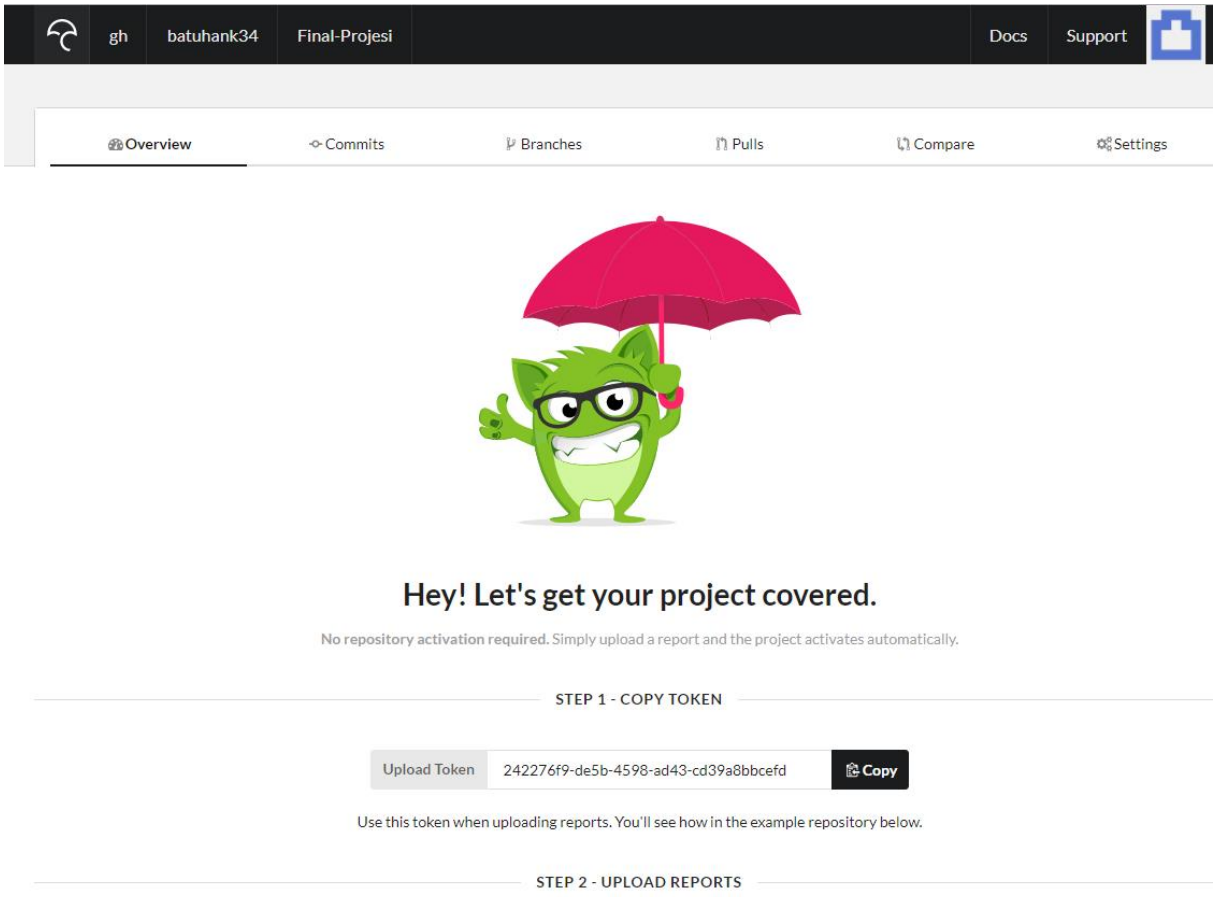
1. Çıkan sayfanın en altında “Add new repository” seçeneğini seçiyoruz.
2. Oluşturduğumuz repomuzu seçiyoruz “Final-Projesi”.
3. “Figur 73” ekrana gelecektir.



Figur 71



Figur 72



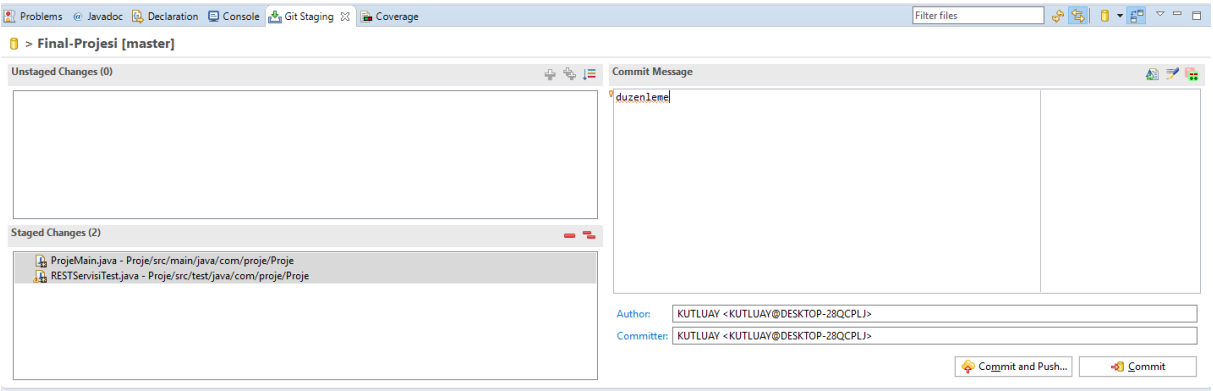
Figur 73

10.3. TRAVIS VE CODECOV'UN TETİKLENMESİ

Kodlarımızı Commit ederek bu işlemi gerçekleştiriyoruz.

Figur 74 İçin Adımlar

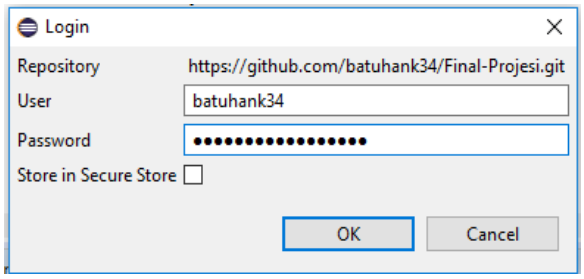
1. Değişiklik yaptığımız kodlarımızı "Git Staging" altında görebiliyoruz.
2. Bu iki kodu "Commit And Push" yapıyoruz.
3. Karşımıza "Figur 75" gelecektir.



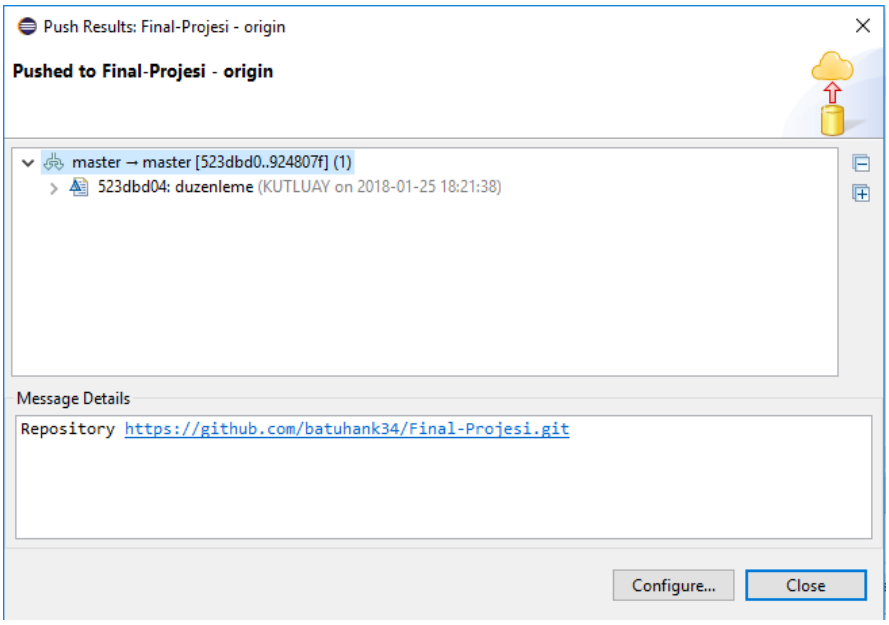
Figur 74

Figur 75 İçin Adımlar

1. GitHub Kullanıcı Adı ve Şifremizi giriyoruz.
2. Karşımıza "Figur 76" gelecektir.Gönderilme işleminin başarılı olduğunu gösterir.



Figur 75

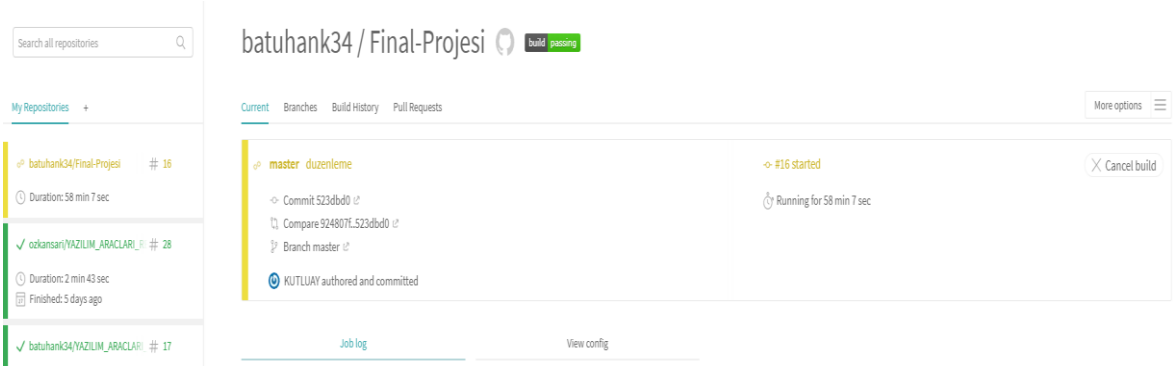


Figur 76

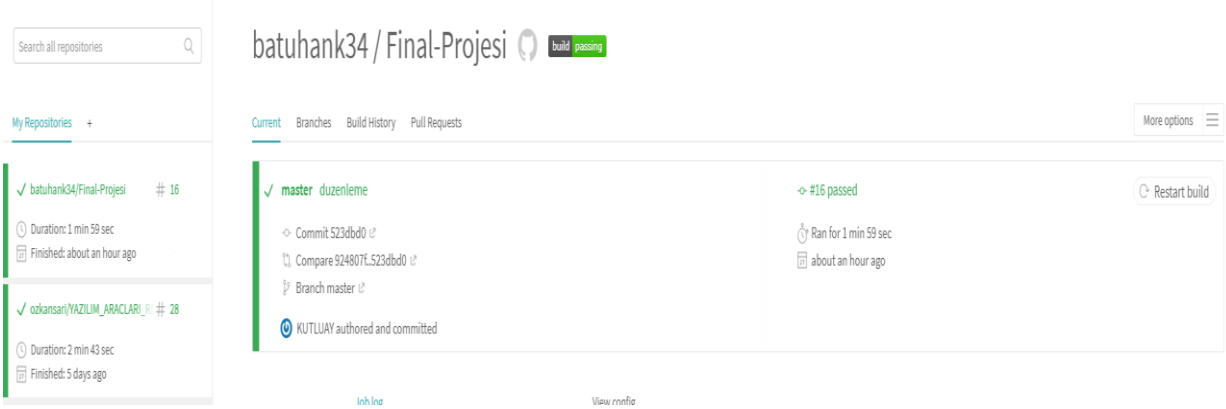
10.4. TRAVIS VE CODECOV SONUÇLARI

Figur 77 - 78 - 79 İçin Adımlar

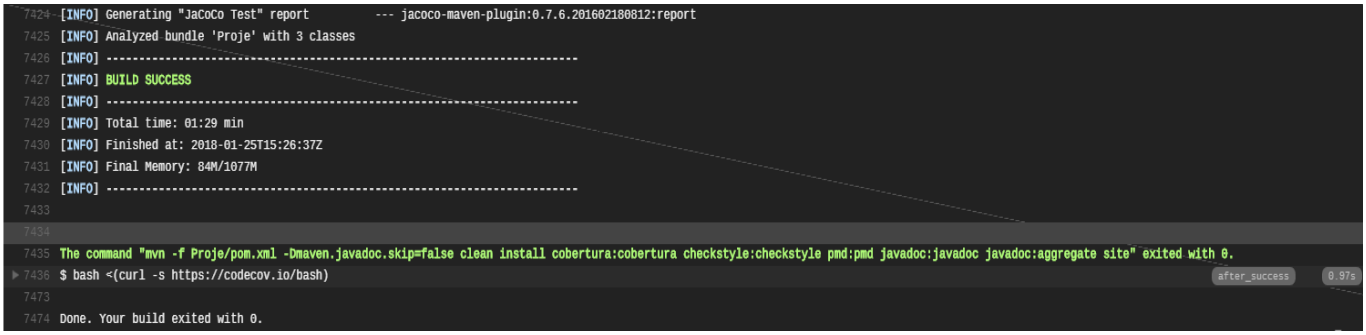
1. Commit işlemi bittikten sonra Travis otomatik olarak projemizi derlemeye başladı.
2. Build işlemi bittikten sonra “Figur 78” ekrana gelecektir.Başarılı bir şekilde build olduğunu görebiliyoruz.
3. Repositoryimizin “Job Log”una bakarsak “BUILD SUCCES” yazısını görebiliriz (Figur 79).



Figur 77



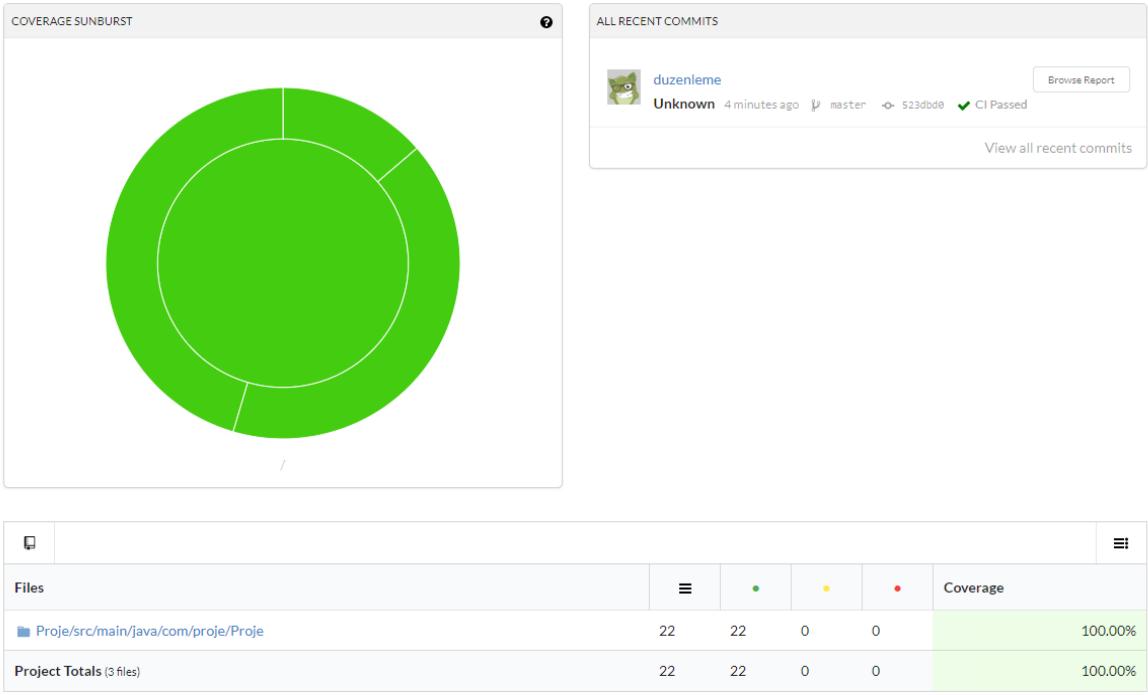
Figur 78



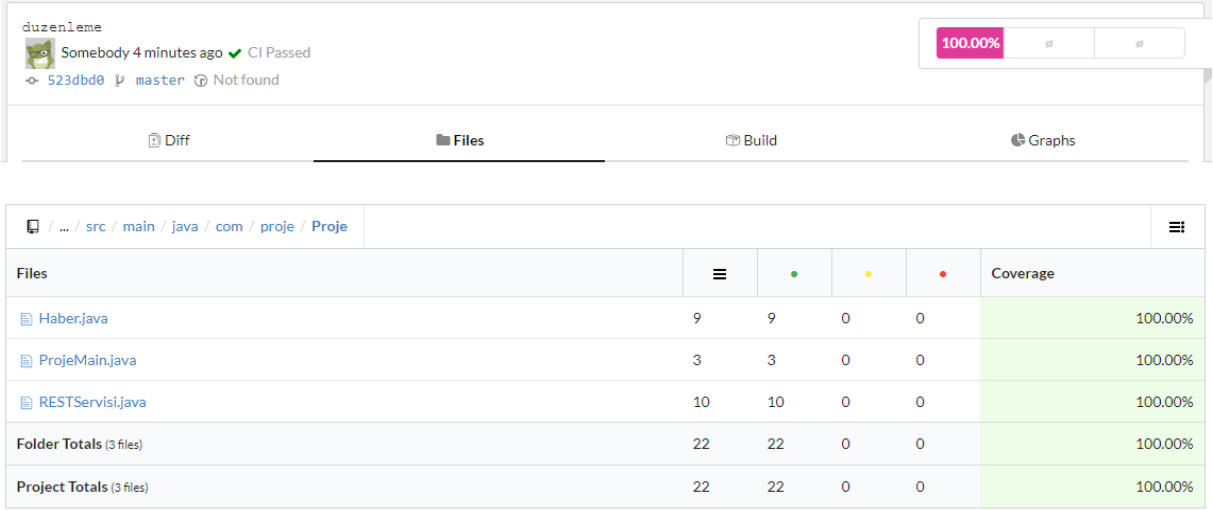
Figur 79

Figur 80 - 81 İçin Adımlar

1. CodeCov için ise “Figur 73”te açılan sayfayı yenilediğimizde ekrana aşağıdaki gibi sonuçlar gelecektir.
2. “Files” kısmından projemize tıklarsak ise “Figur 81” karşımıza gelir.



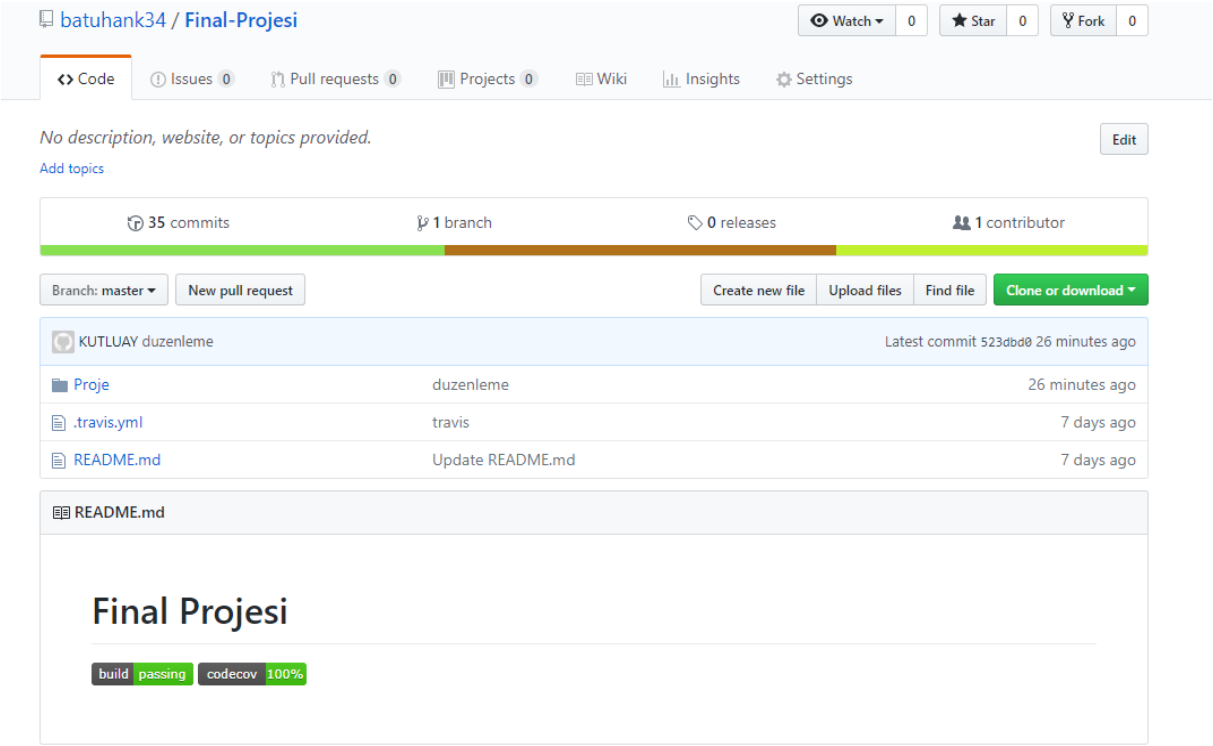
Figur 80



Figur 81

Figur 82 İçin Adımlar

1. Travis ve Codecov işlemleri bittikten sonra otomatik olarak sonuçlar repositoryimizin“README.md” kısmında gözükmeye başladı.



Figur 82

11. POSTMAN VE JMETER

Postman ve Jmeter testleri yapılırken projemizin çalışır durumda olması gerekmektedir.

11.1. POSTMAN

Postman rest clientını kullanabilmek için gerçekleştirlmesi gereken adımlar vardır.

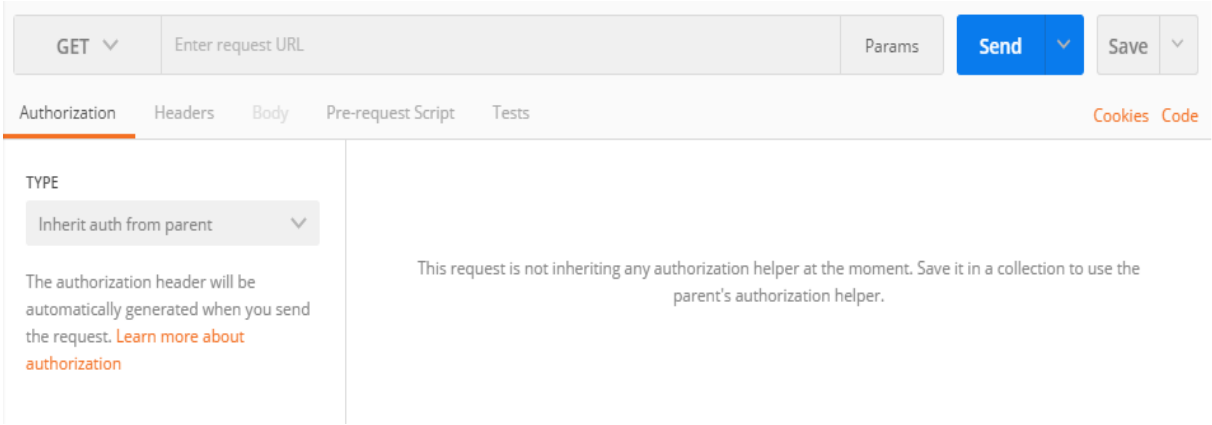
Figur 83 İçin Adımlar

1. https://www.getpostman.com/ sitesine giriyoruz.
2. İşletim sistemimize uygun olan versiyonu indirip kurulumunu yapıyoruz.

Figur 83

Figur 84 İçin Adımlar

1. “Enter request URL” kısmına “RESTServisi” sınıfında oluşturduğumuz Request Mappingleri yazıp ekran çıktılarını görebiliriz.

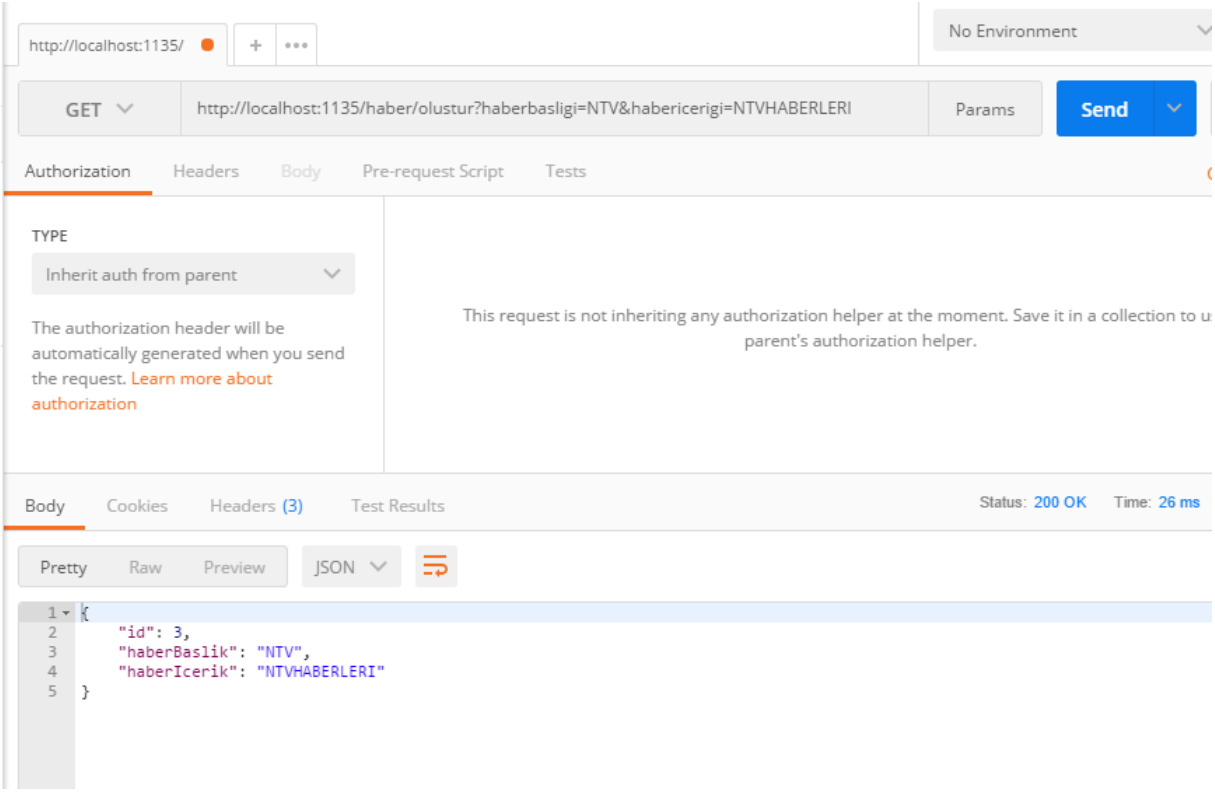


Figur 84

@RequestMapping("/haber/olustur")

Figur 85 İçin Adımlar

1. Urlyi yazıp “SEND” butonuna basıyoruz.3 kere ekleme yaptım.

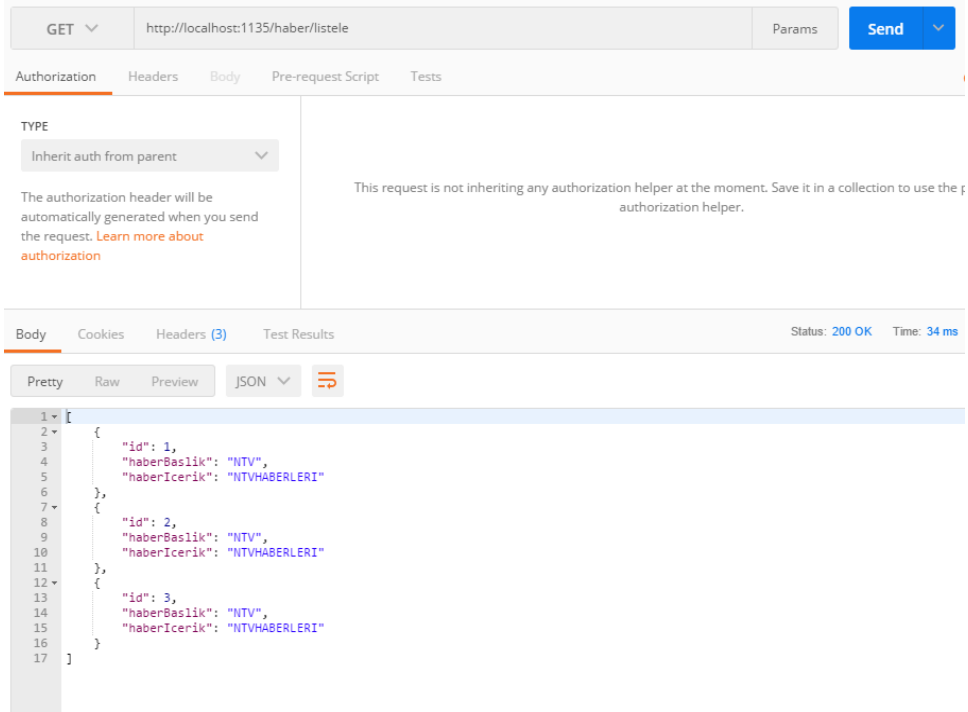


Figur 85

@RequestMapping("/haber/listele")

Figur 86 İçin Adımlar

1. Urlyi yazıp “SEND” butonuna basıyoruz.Ve Eklediğimiz öğeler listeleniyor.

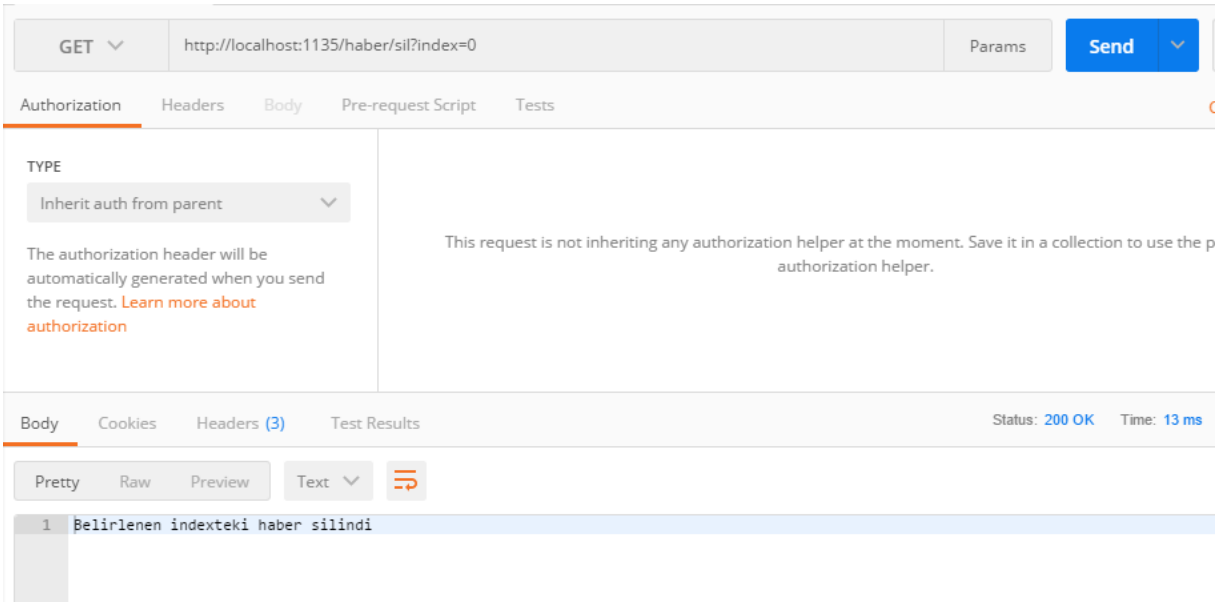


Figur 86

@RequestMapping("/haber/sil")

Figur 87 İçin Adımlar

1. Urlyi yazıp “SEND” butonuna basıyoruz.Belirlediğimiz indexteki haberi siliyor

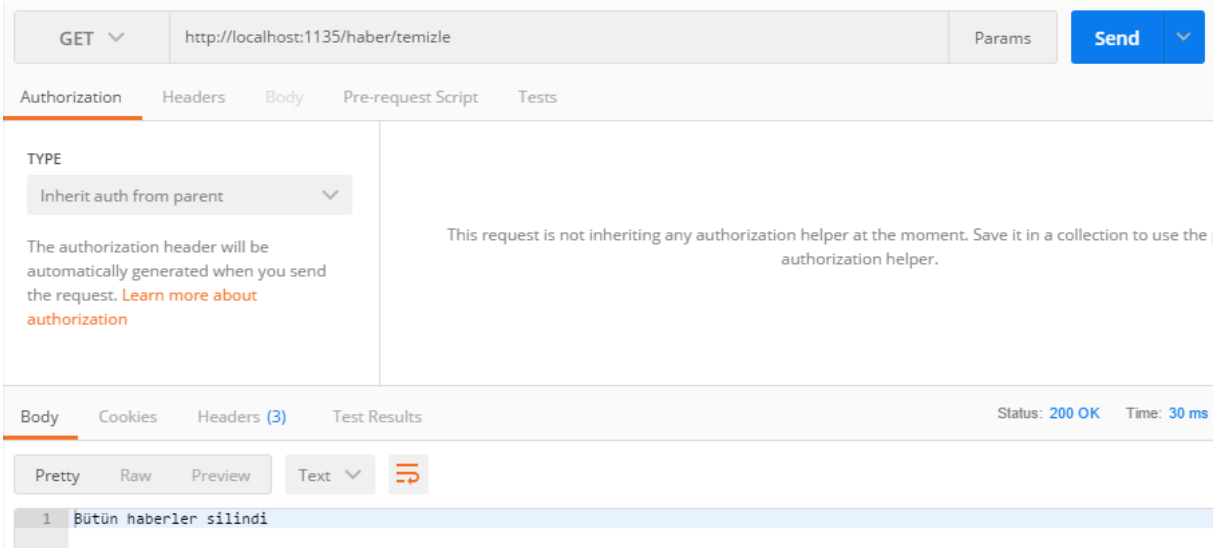


Figur 87

@RequestMapping("/haber/temizle")

Figur 88 İçin Adımlar

1. Urlyi yazıp “SEND” butonuna basıyoruz.Bütün öğeleri temizliyor.



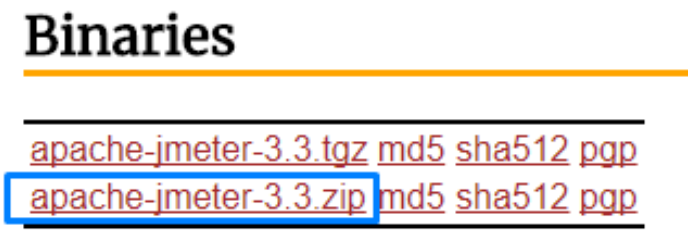
Figur 88

11.2. JMETER

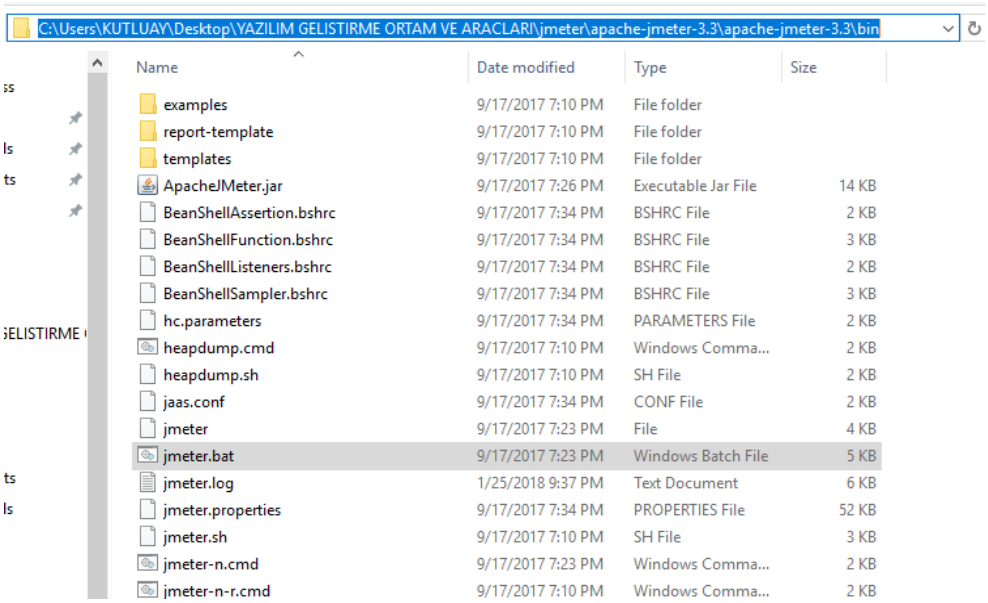
Jmeter yazdığımız REST servislerini test etmemizi sağlar.Jmeter kullanmak için gerekli adımlar vardır.

Figur 89-90 İçin Adımlar

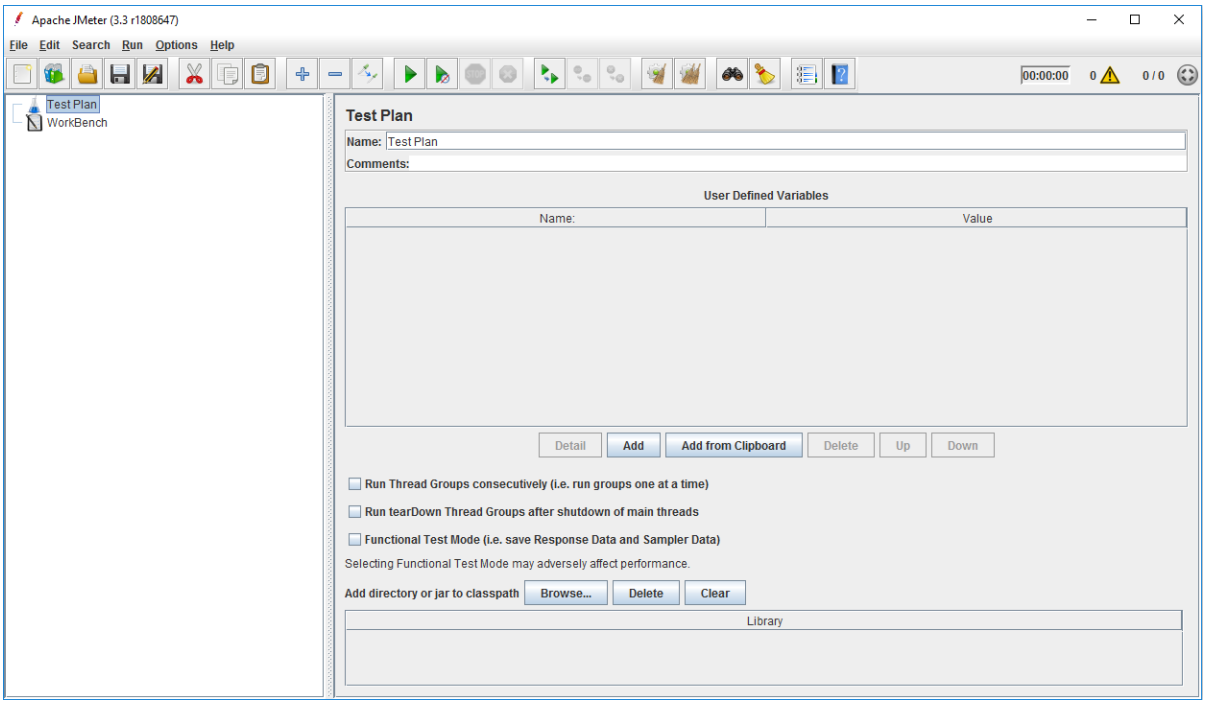
1. http://jmeter.apache.org/download_jmeter.cgi sitesine giriyoruz.
2. “Binaries” kısmından zip olarak programı indiriyoruz.
3. Zipten çıkardıktan sonra “\jmeter\apache-jmeter-3.3\apache-jmeter-3.3\bin” yoluna gidiyoruz(Figur 90) daha sonra “jmeter.bat”ı çalıştırıyoruz ve program açılıyor(Figur 91).



Figur 89



Figur 90

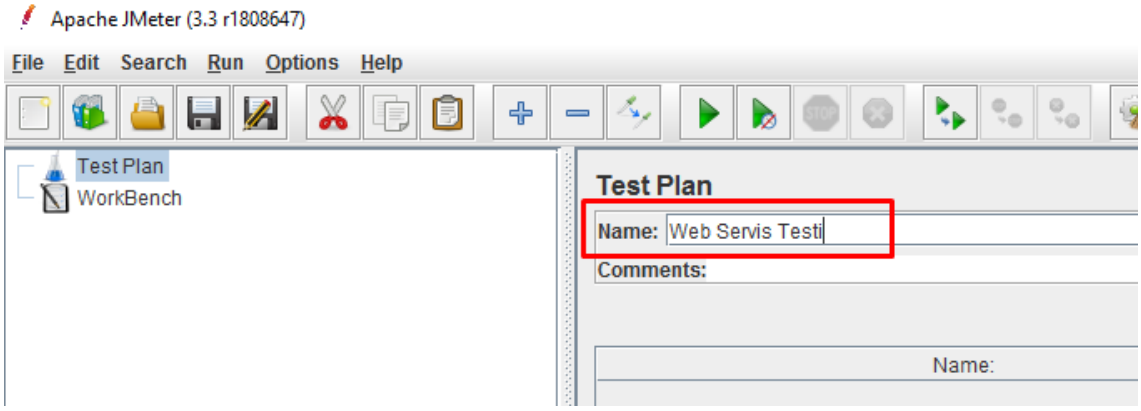


Figur 91

Web Servis Testleri İçin Adımlar

Figur 92 İçin Adımlar

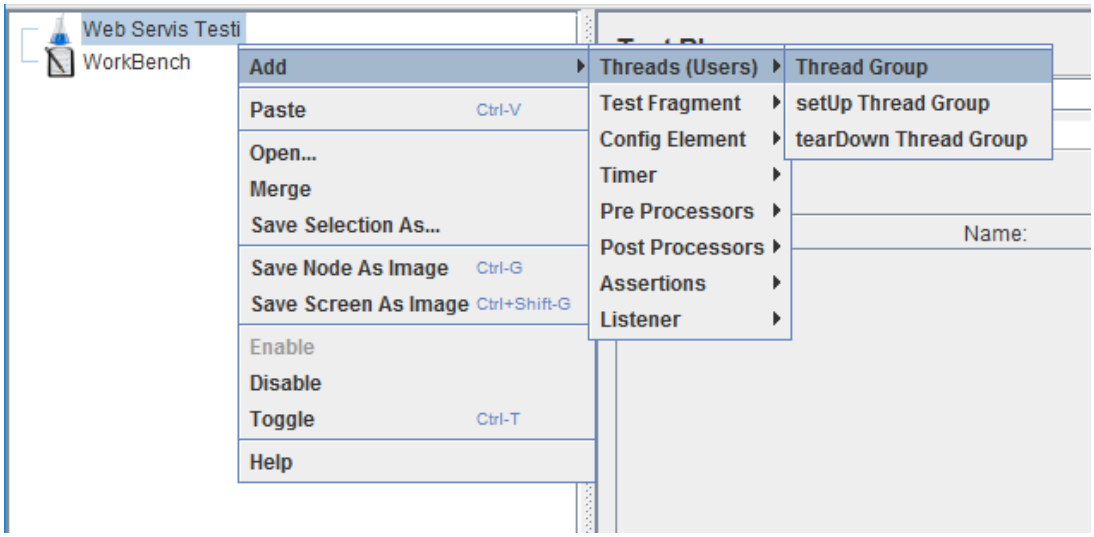
1. Soldaki menüden “TestPlan”a tıklıyoruz.
2. Sağ tarafta TestPlanın ismini değiştirebileceğimiz yer çıkıyor.
3. “Web Servis Testi” olarak isimlendiriyorum.



Figur 92

Figur 92 İçin Adımlar

1. “Web Servis Testi”ne sağ click yapıyoruz.
2. “Add” diyoruz.
3. “Threads(Users)”ı seçiyoruz.
4. “Thread Group”a tıklıyoruz ve 1 adet “Thread Group” oluşturmuş oluyoruz.

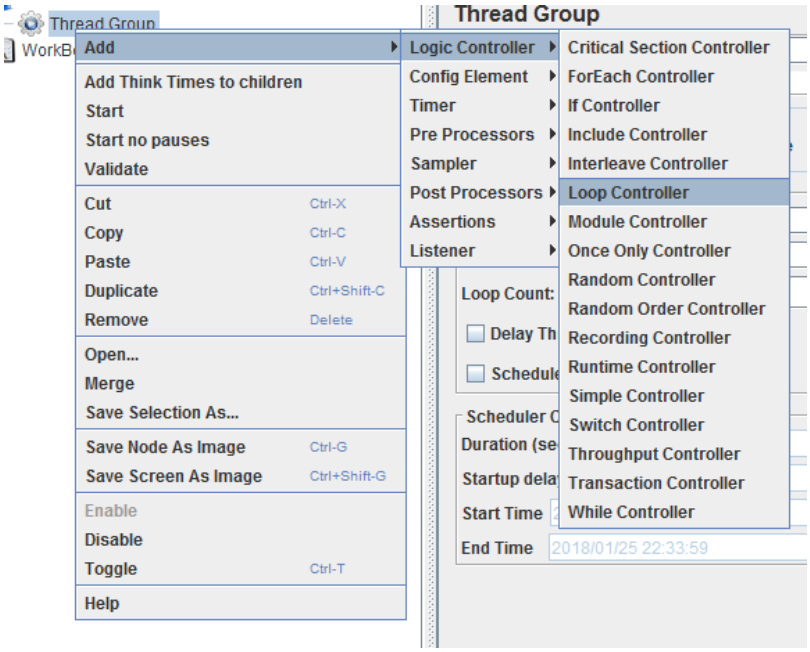


Figur 93

Web Servis Testinde Döngü Oluşturmak,Döngüye Veri Eklemek Ve Göstermek İçin Adımlar

Figur 94 İçin Adımlar

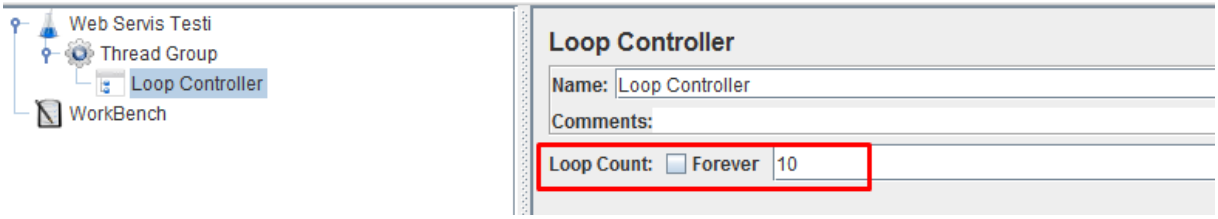
1. “Thread Group”a sağ click yapıyoruz.
2. “Add” diyoruz.
3. “Logic Controller”ı seçiyoruz.
4. “Loop Controller”a tıklıyoruz.Döngümüzü oluşturmuş oluyoruz.



Figur 94

Figur 94 İçin Adımlar

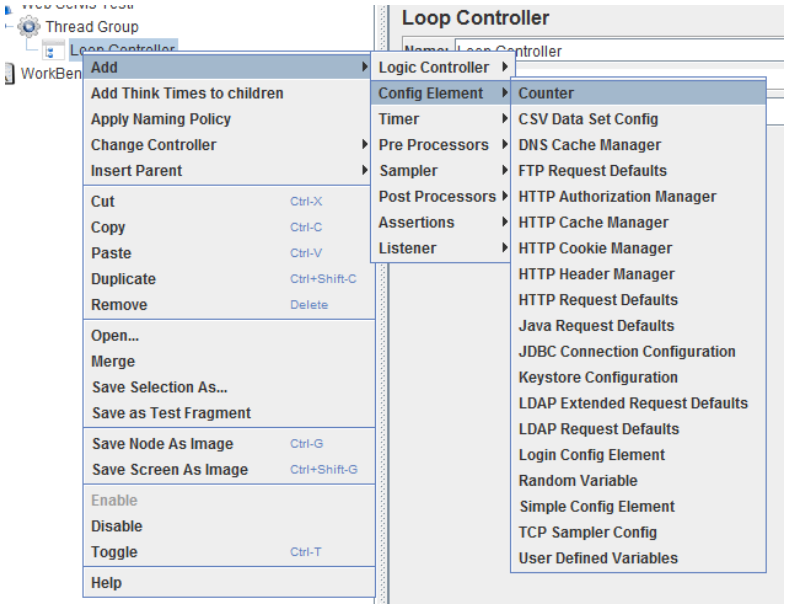
1. “Loop Controller”a tıklıyoruz.
2. Yan tarafta çıkan “Loop Count”u döngümüzün ne kadar dönmesini sağlıyağımız yer.Ben 10 yazdım.



Figur 95

Figur 96 İçin Adımlar

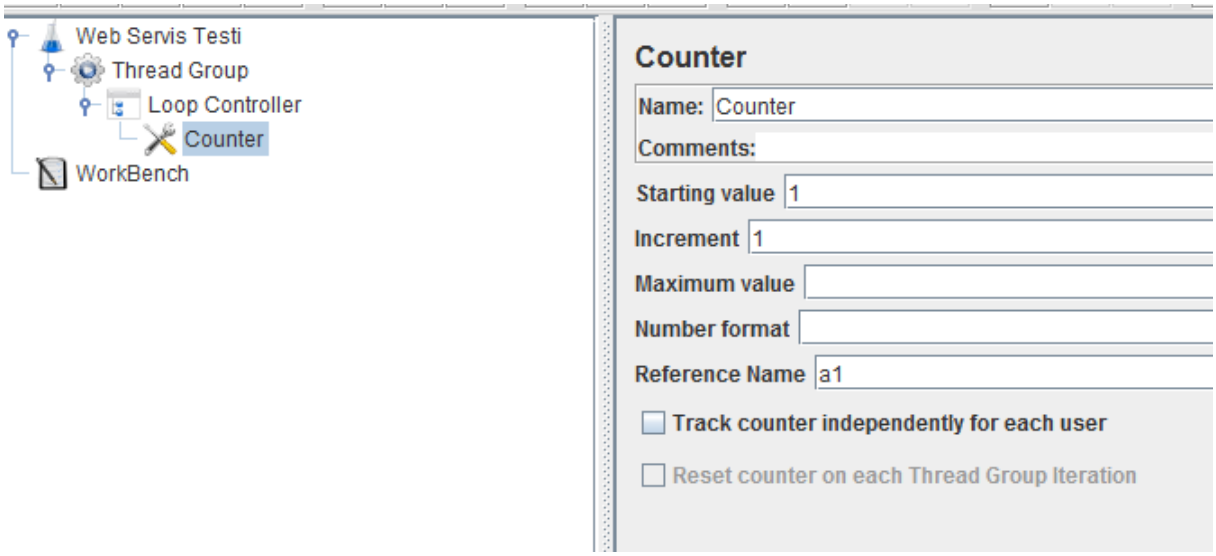
1. “Loop Controller”a “Sağ Click” yapıyoruz.
2. “Add” diyoruz.
3. “Config elementi” seçiyoruz.
4. Son olarak “Counter”ı seçerek döngümüze sayaç eklemiş oluyoruz.



Figur 96

Figur 97 İçin Adımlar

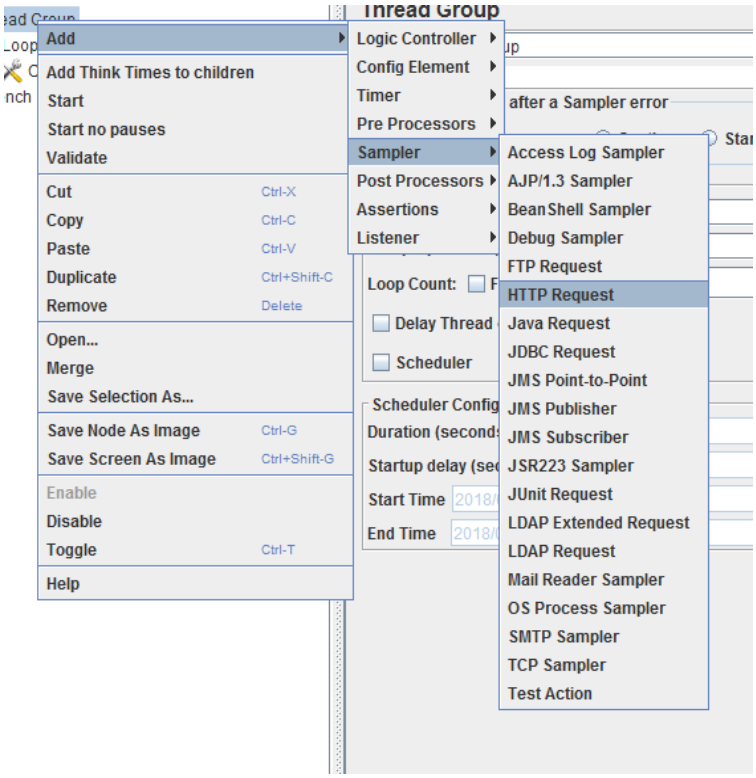
1. “Counter”a tıklıyoruz.
2. Yan tarafta Counterın özelliklerini ayarlıyoruz.
3. “Starting Value”ya sayaçımızın başlangıç değerini veriyoruz.Ben 1 verdim.
4. “Increment”ta sayaçın kaçar kaçar artıcağını ayarlıyoruz.Ben birer birer arttırdım.
5. “Reference Name” ise sayaçın değerini tutacak yerimiz.



Figur 97

Figur 98 İçin Adımlar

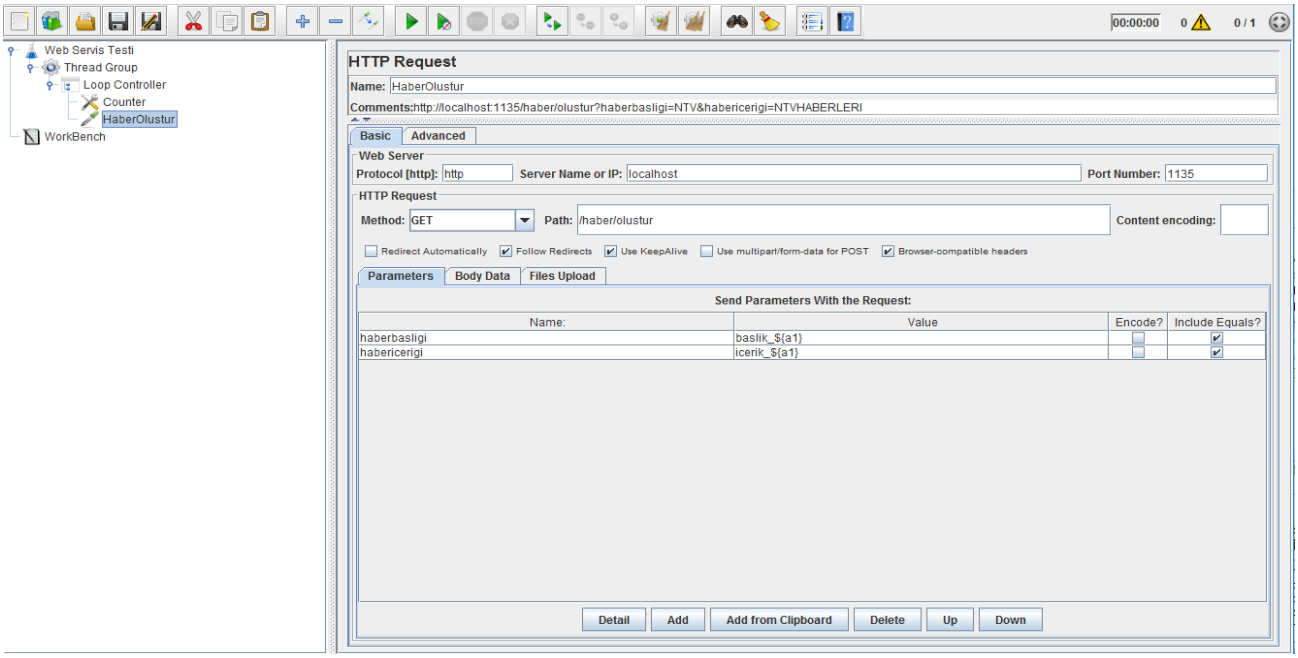
1. “Loop Controller”a sağ click yapıyoruz.
2. “Add” diyoruz.
3. “Sampler” seçeneğini seçiyoruz.
4. Daha sonra “HTTP Request”i seçiyoruz.Bu şekilde HTTP istek formatını eklemiş oluyoruz.



Figur 98

Figur 99 İçin Adımlar

1. “HTTP Request”e tıklıyoruz.
2. Haber oluşturalım bu requestte.
3. Yan tarafta çıkan “HTTP Request”in özelliklerinden ismini değiştirelim.Ben “Haber Olustur” yaptım.
4. “Protocol[http]:” kısmına “http” yazıyoruz.
5. “Server Name or IP” kısmına “localhost” yazıyoruz.
6. “Port Number”a springi hangi porttan başlatıyorsak o portu yazıyoruz.Benim 1135.
7. “Path” kısmına haber oluşturduğumuz Request Mappingi yazıyoruz.
8. “Parameters” kısmında habere eklediğimiz parametleri yazıyoruz.
9. “Add” ile parametre oluşturuyoruz.
10. Şu durumda “haberbasligı” ve ”habericerigi” isimlerini yazdık.
11. “Value” değeri tanımlama kuralı ise “HerhangiBirIsim_${Sayacımıza verdiğimiz reference name}”



Figur 99

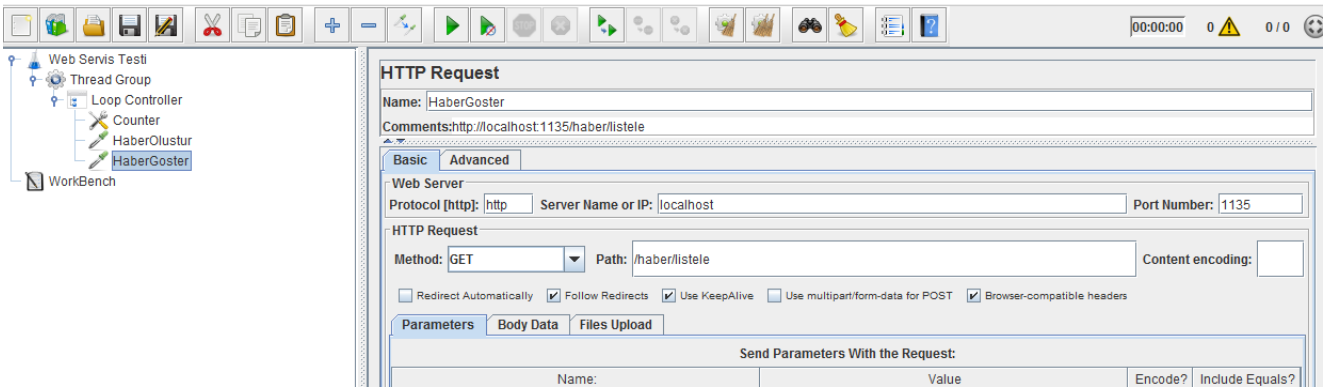
Figur 100 İçin Adımlar

1. “HaberOlustur”a “Sağ Click”yapıyoruz.
2. “Duplicate” seçeneğini seçiyoruz.

Figur 100

Figur 101 İçin Adımlar

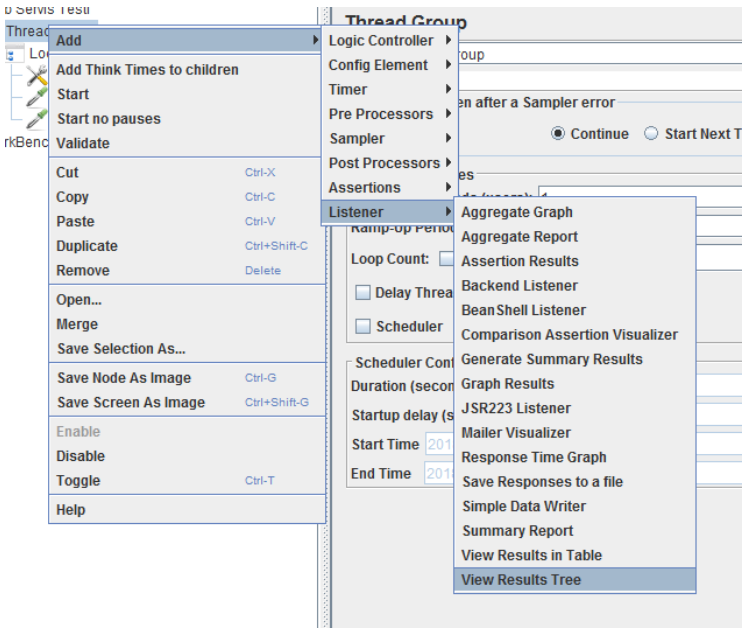
1. Duplicate olan “HaberOlustur”a tıklıyoruz.
2. Yan taraftan ismini “HaberGoster” olarak değiştiriyoruz.
3. Parametreleri siliyoruz.



Figur 101

Figur 102 İçin Adımlar

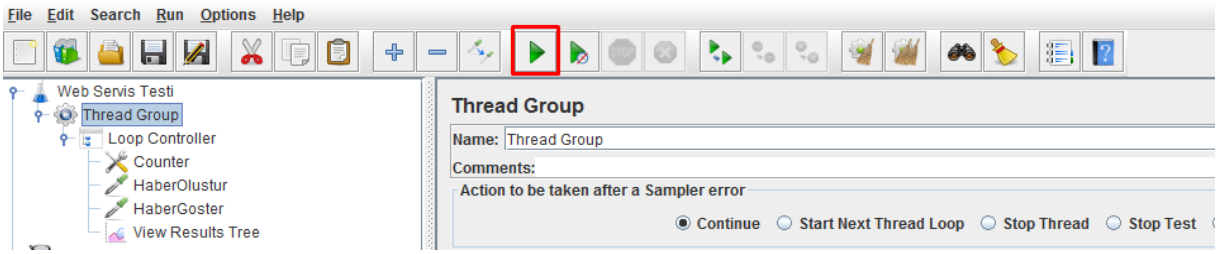
1. “Loop Contoller”a “Sağ Click” yapıyoruz.
2. “Add” diyoruz.
3. “Listener” seçeneğini seçiyoruz
4. Daha sonra “View Result Tree”yi seçerek yaptığımız requestlerin sonucunu gösteren bir olay ekliyoruz.



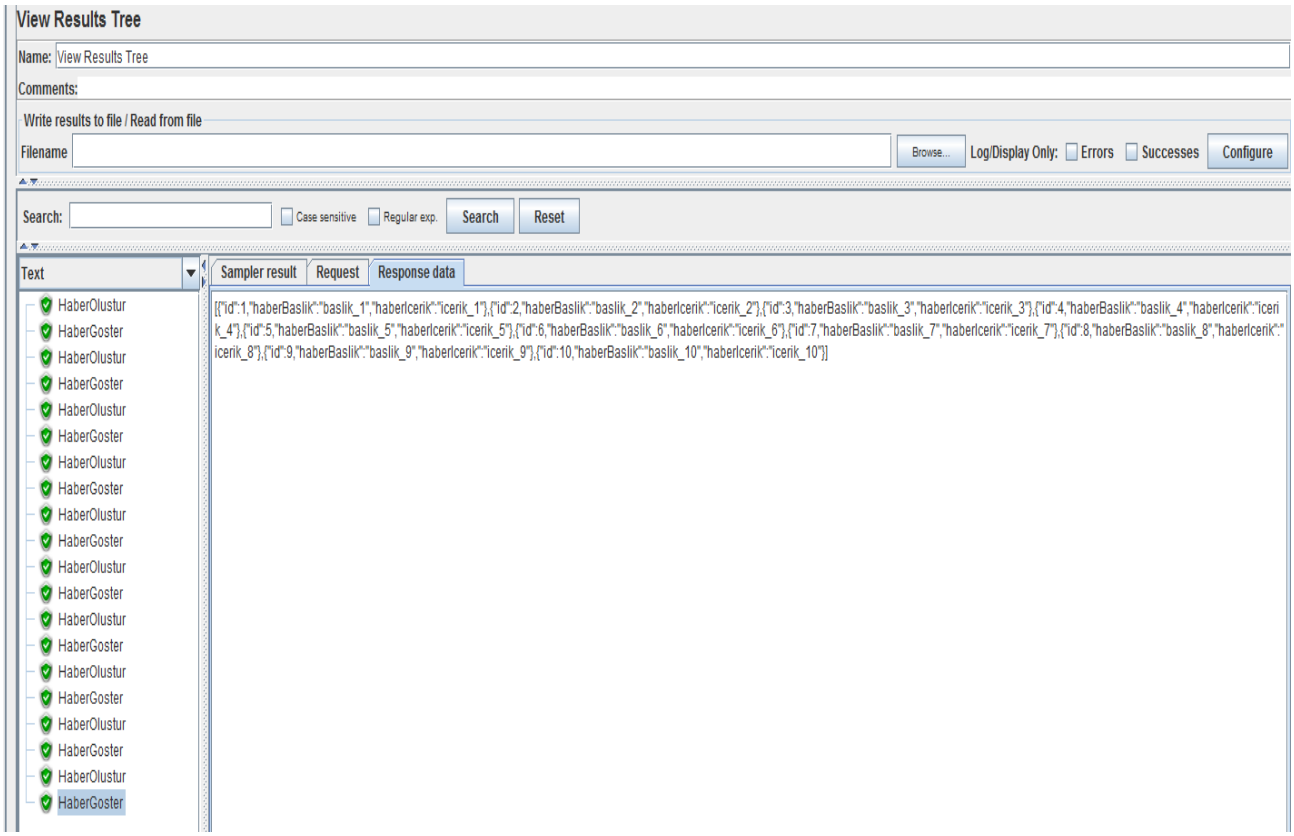
Figur 102

Figur 103-104 İçin Adımlar

1. “Run” diyerek Thread Group’u çalıştırıyoruz.
2. “View Results Tree”ye baktığımız zaman on tane birer birer artmış bir şekilde oluştuğunu görebiliriz(Figur 104).

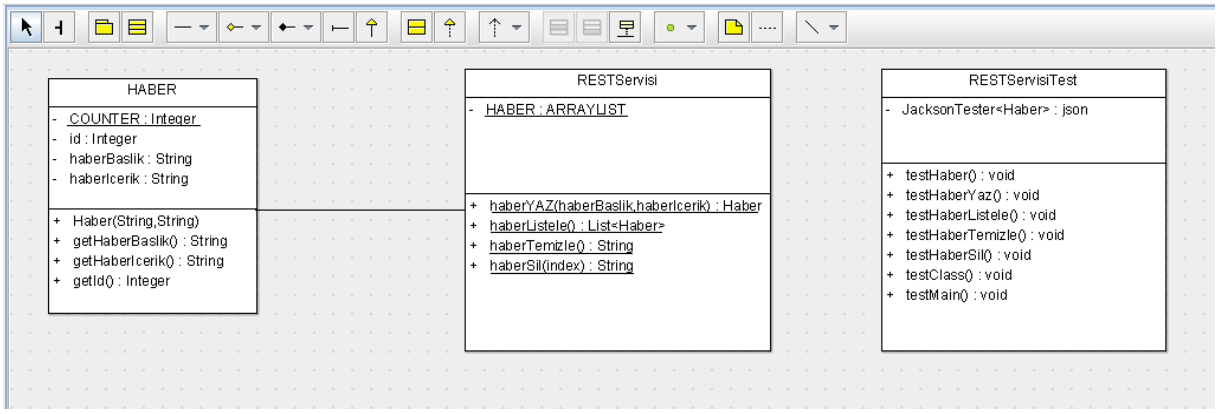


Figur 103


View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Search:
Case sensitive
Regular exp.
Search
Reset
Text
Sampler result
Request
Response data
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
HaberOlustur
HaberGoster
[{"id":1,"haberBaslik":"baslik_1","haberIcerik":"icerik_1"},{"id":2,"haberBaslik":"baslik_2","haberIcerik":"icerik_2"},{"id":3,"haberBaslik":"baslik_3","haberIcerik":"icerik_3"},{"id":4,"haberBaslik":"baslik_4","haberIcerik":"iceri
k_4"},{"id":5,"haberBaslik":"baslik_5","haberIcerik":"icerik_5"},{"id":6,"haberBaslik":"baslik_6","haberIcerik":"icerik_6"},{"id":7,"haberBaslik":"baslik_7","haberIcerik":"icerik_7"},{"id":8,"haberBaslik":"baslik_8","haberIcerik":"
icerik_8"},{"id":9,"haberBaslik":"baslik_9","haberIcerik":"icerik_9"},{"id":10,"haberBaslik":"baslik_10","haberIcerik":"icerik_10"}]

Figur 104

12. UML DİAGRAMI



Figur 105

HABER SINIFI

Değişkenleri tanımladığımız kısımda:"COUNTER", static olduğu için altı çizili ve Integer tipinde ."id" Integer tipinde. "haberBaslik" ve "haberIcerik" String tipinde tanımlıyoruz. Bütün değişkenler private olduğu için de başlarına "-" koyuyoruz.

Operasyonları tanımladığımız kısımda:Sınıfın "Constructor"ını tanımlıyoruz. "haberBaslik" ve "haberIcerik" olarak String tipinde parametreleri olduğu için diagramda String olarak ayarlıyoruz."gethaberBaslik" ve "getHaberIcerik" metotları String tipinde olduğu için onlarıda String olarak yazıyoruz."getId" ise int tipinde bir metot olduğu için onun da tipini in olarak ayarlıyoruz.Ayrıca operasyonların hepsinin başında "+" var çünkü hepsi public bir metot.

RESTServisi SINIFI

Değişkenleri tanımladığımız kısımda:Haber tipinde ArrayListimiz olduğu için onu tanımlıyoruz.Static olduğu için altı çizgili ve private olduğundan dolayı başında "-" mevcut.

Operasyonları tanımladığımız kısımda:"haberYAZ" metotunun aldığı parametreleri yazıyoruz ve tipini beliriyoruz.Haber türünden oluştuğu için tipi Haber."haberListele" metotunun tipi "List<Haber>" olduğu için diagramda belirttim."haberTemizle" metotunun tipini String olduğu için belirtiyoruz."haberSil" metotonun aldığı parametreyi belirtip metotun tipini yazdım.

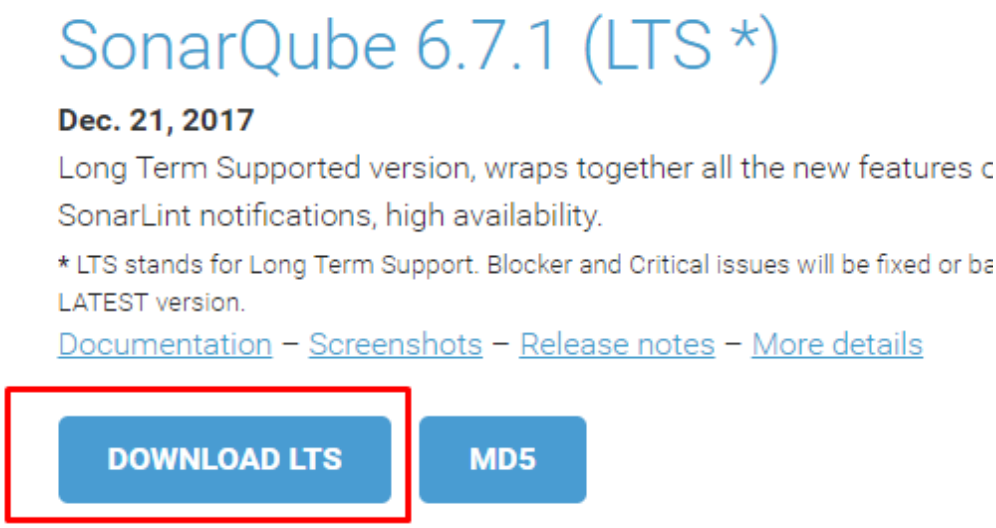
RESTServisiTest Sınıfı

Değişkenleri tanımladığımız kısımda:JacksonTester tipinde Haber sınıfından json tanımlıyoruz.
Operasyonları tanımladığımız kısımda:Bütün sınıfları test ettiğimiz metotları yazıyoruz. Hepsinin tipi void olduğu için diagramda o şekilde belirtiyoruz.Ayrıca başlarına "+" koyuyoruz çünkü hepsi public metotlar.

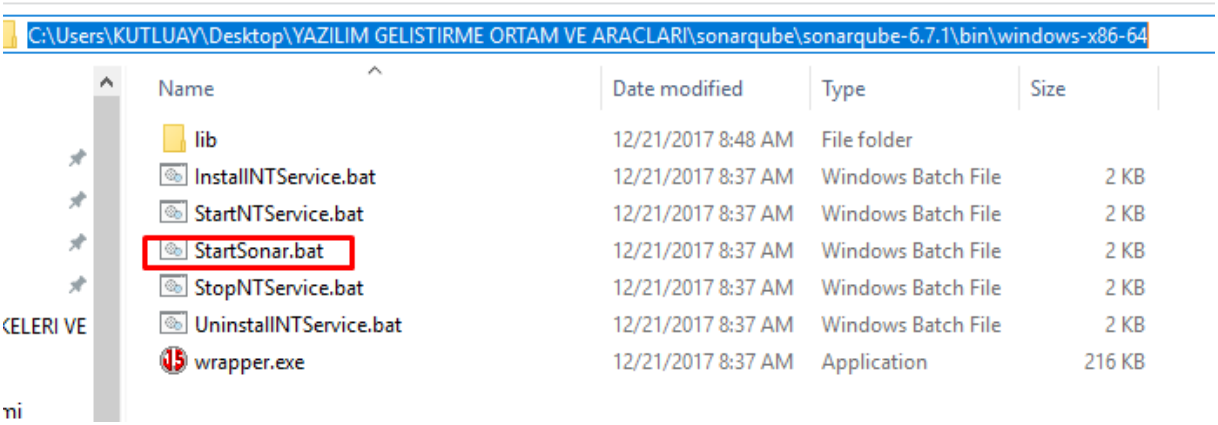
13. SONARQUBE

Figur 106 -107 İçin Adımlar

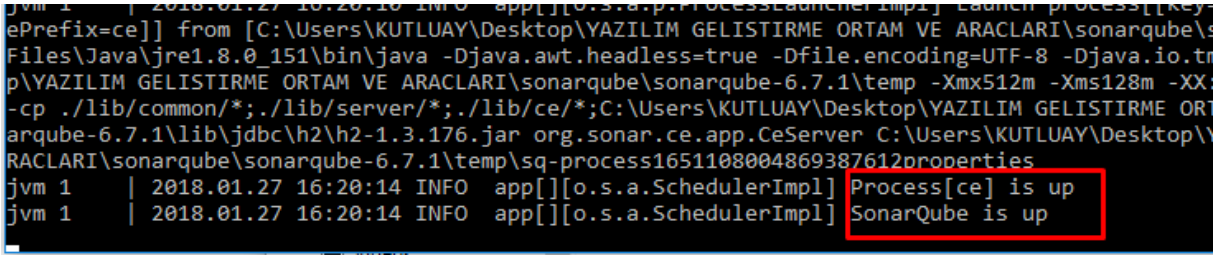
1. https://www.sonarqube.org/downloads/ sitesine girip “SonarQube”ü indiriyoruz.
2. İndirelen dosyayı çıkardıktan sonra “bin” klasörünün altında sistemimiz kaç bit ise o dosyaya giriyoruz
3. “StartSonar.bat”ı çalıştırıyoruz.
4. Yükleme işlemeri bittikten sonra sonarın çalıştığını bize söyliyecektir(Figur 108).Sonar http://localhost:9000/ üzerinden çalışır.



Figur 106



Figur 107



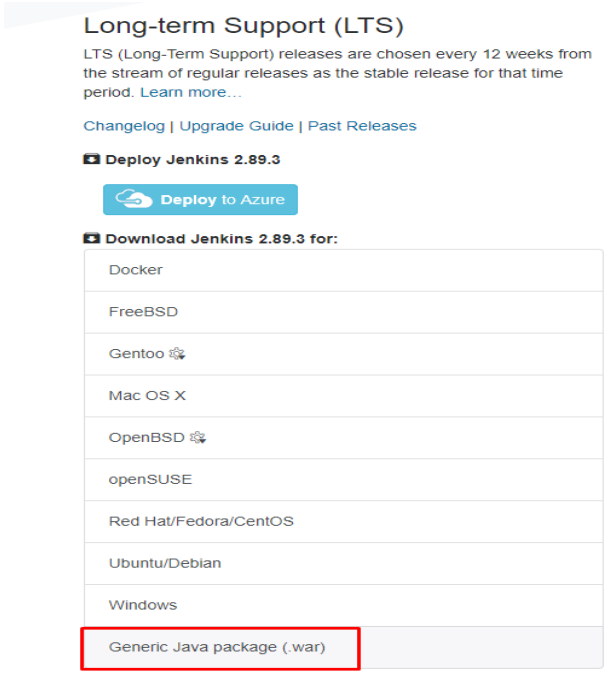
Figur 108

14. JENKINS

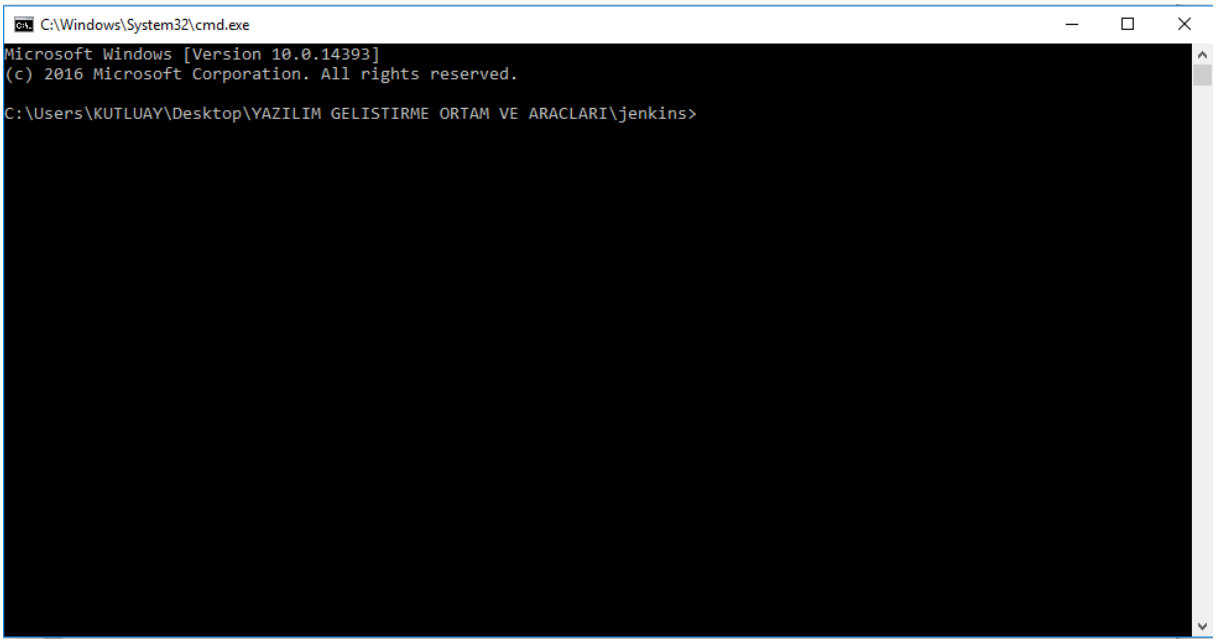
Açık kaynak kodlu bir otomasyon sunucusudur.

Figur 109-110 İçin Adımlar

1. https://jenkins.io/download/ sitesine giriyoruz.
2. ".war" dosyasını indiriyoruz
3. "cmd.exe"yi çalıştırıp "jenkins"i indirdiğimiz yere gidiyoruz.



Figur 109



Figur 110

Figur 111-112 İçin Adımlar

1. Konsolda “java –jar jenkins.war httpPort =49001” yazıp jenkinsi yükleme işlemini başlatıyoruz.Ayrıca hangi porttan çalışaçağını da belirledik.
2. Yükleme bittikten sonra bize daha sonraki adımlarda isteyeceği bir şifre veriyor.Onu kaydediyoruz(Figur 112).

```
C:\Users\KUTLUAY\Desktop\YAZILIM GELISTIRME ORTAM VE ARACLARI\jenkins>java -jar jenkins.war --httpPort=49001
```

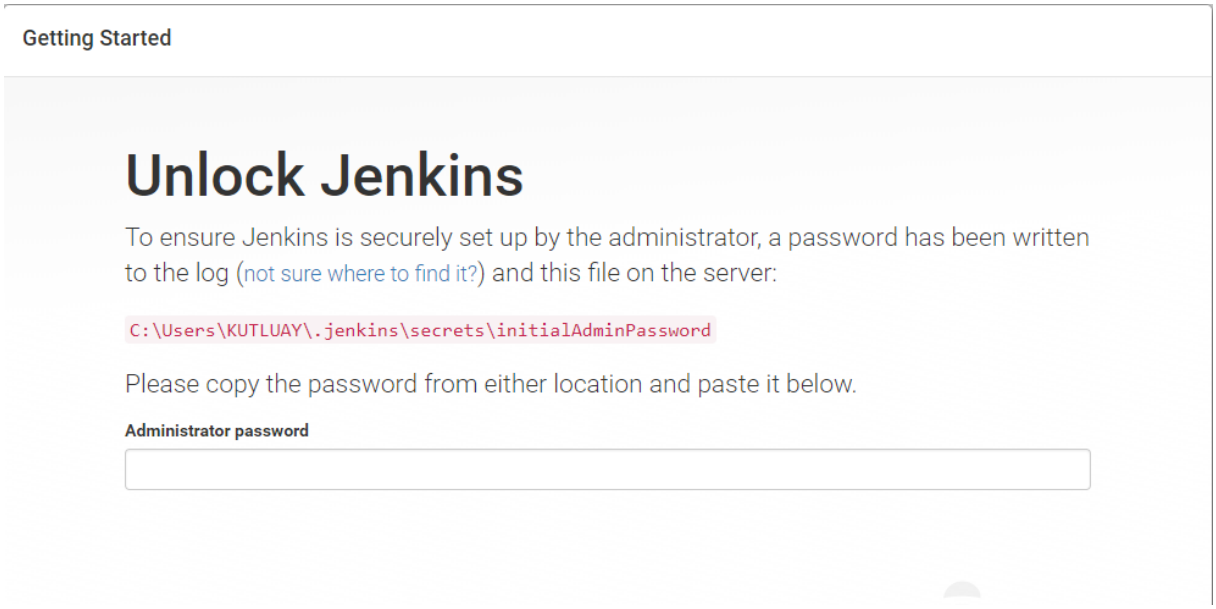
Figur 111

```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

1831f88c4d90481693de82a63adae59a

This may also be found at: C:\Users\KUTLUAY\.jenkins\secrets\initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

Jan 27, 2018 3:58:05 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Jan 27, 2018 3:58:05 PM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
```

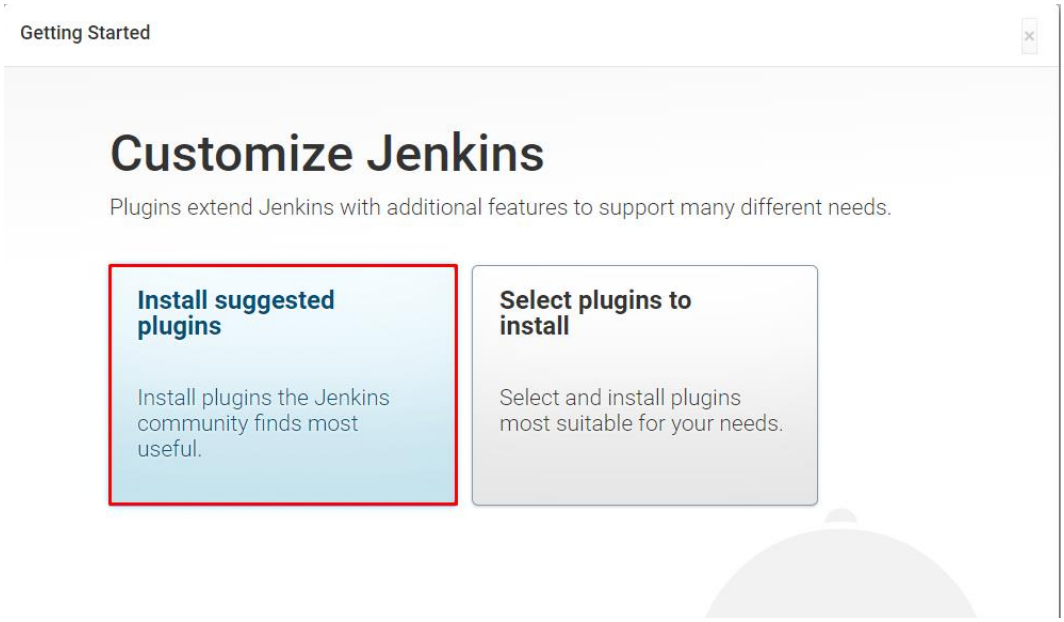
Figur 112

Figur 113-114 İçin Adımlar

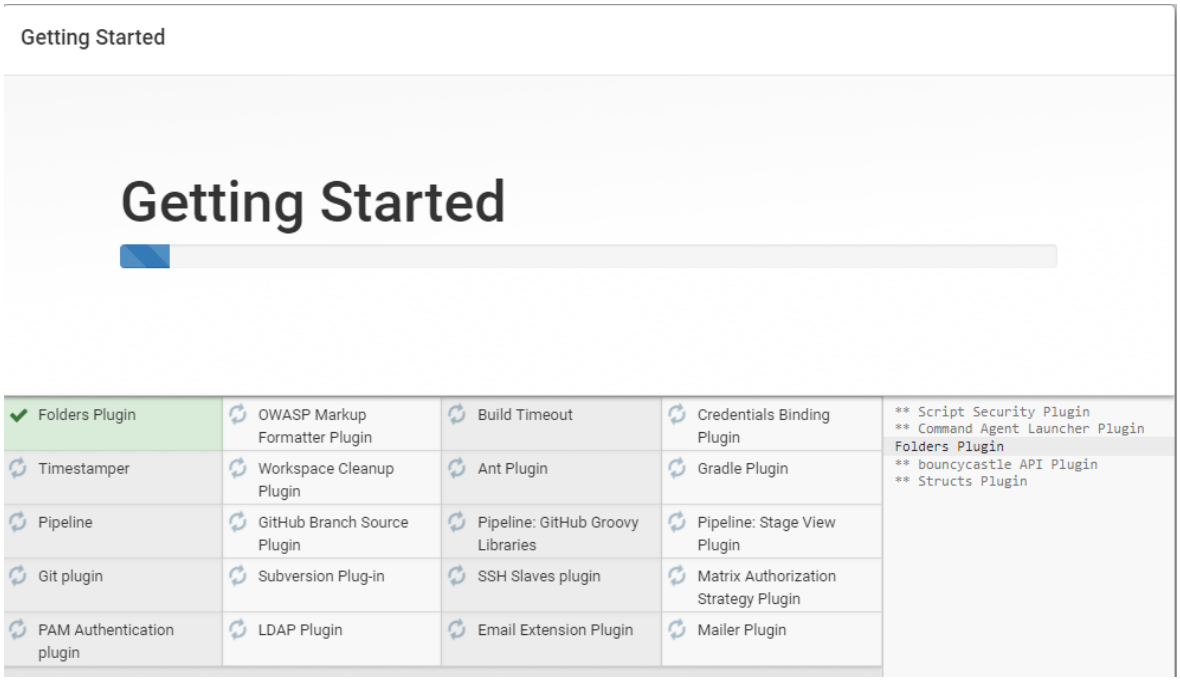
1. localhost:49001 adresine girdiğimiz zaman karşımıza böyle bir ekran gelecektir.
2. Bir önceki adımda kaydettğimiz şifreyi buraya yazıyoruz.
3. Daha sonra gelen ekranda (Figur 114) “Install suggested plugins seçeneğini seçiyoruz” ve yüklemeye başlıyor.(Figur 115).
4. Yükleme işlemi bittikten sonra karşımıza “Figur 116” gelecektir.



Figur 113



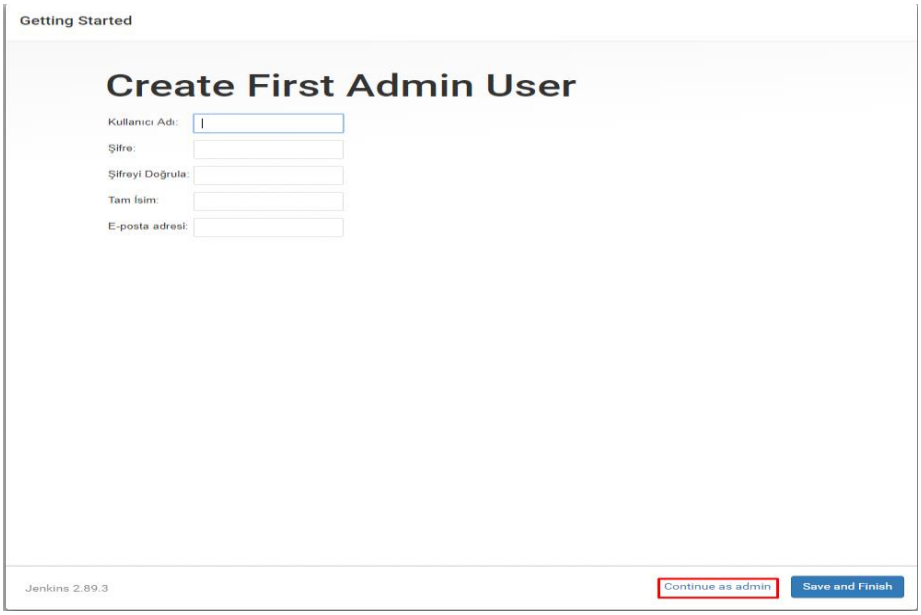
Figur 114



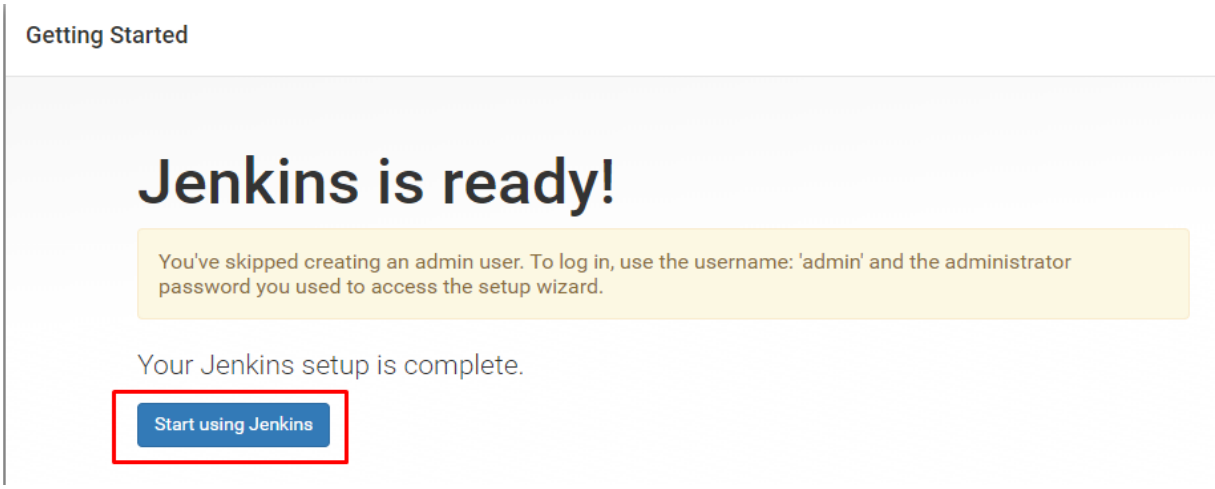
Figur 115

Figur 116 -117 İçin Adımlar

1. Jenkins bizden hesap oluşturmamızı istiyor.”Continue as admin” diyerek burayı geçiyoruz.
2. Karşımıza “Figur 117” gelecektir.”Start Using Jenkins” diyerek ilerliyoruz.



Figur 116

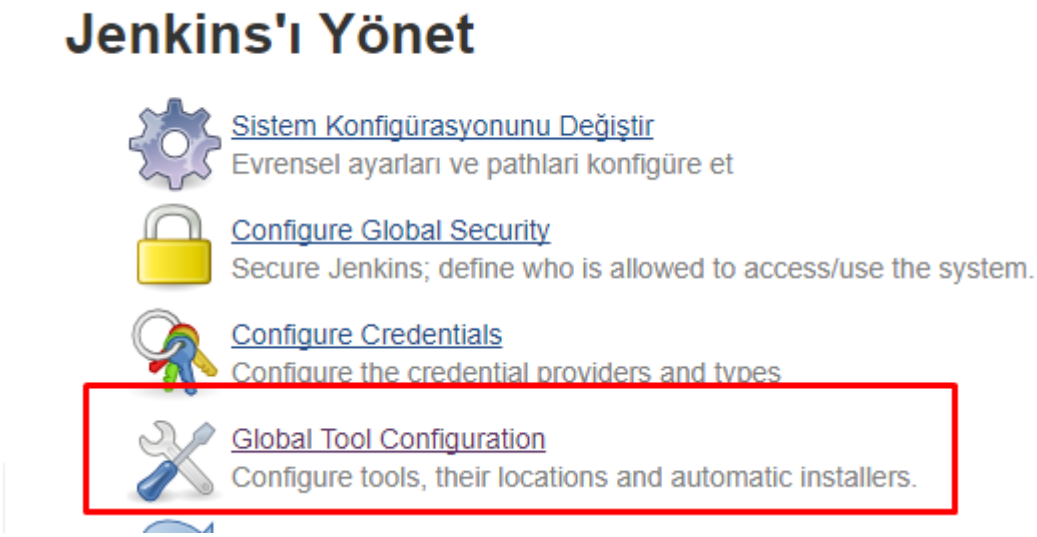


Figur 117

Figur 118-119 İçin Adımlar

1. Jenkins sayfası gelecektir karşımıza soldaki menüden “Jenkinsi’i Yönet” seçeneğini seçiyoruz.
2. Daha sonra karşımıza çıkan sayfda (Figur 119) “Global Tool Configuration” seçeneğini seçiyoruz.

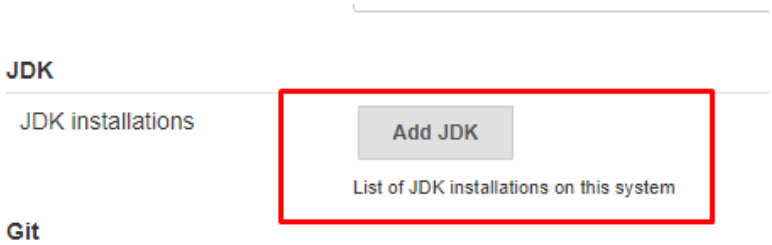
Figur 118



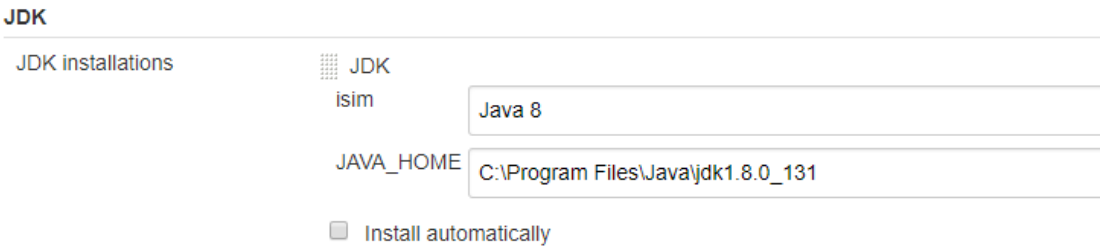
Figur 119

Figur 120 -121 İçin Adımlar

1. “JDK” kısmında “Add JDK” seçenğini seçiyoruz.
2. “Install automatically” seçeneğini kaldırıyoruz
3. İsim olarak “Java 8” “JAVA_HOME” için ise lokalimizde kurulu jdk dizinini oraya yazıyoruz.



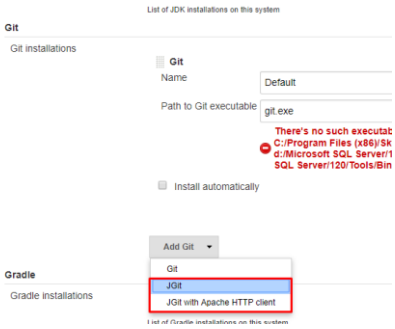
Figur 120



Figur 121

Figur 122 İçin Adımlar

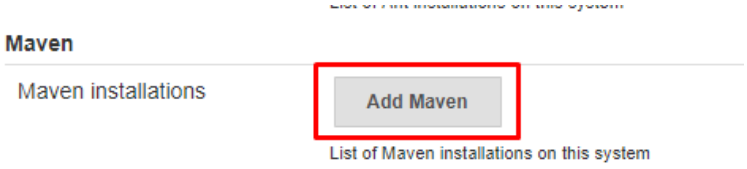
1. “Git” kısmında “Add Git” seçenğini seçiyoruz.
2. “JGit” ve “Jgit with Apache HTTP client” seçeneklerini ekliyoruz



Figur 122

Figur 123 - 124 İçin Adımlar

1. “Maven” kısmında “Add Maven” seçenğini seçiyoruz.
2. İsim olarak “Maven 3.5.2” yazıyoruz.”Install automatically” seçeneğini seçiyoruz.
3. Version kısmını “3.5.2” olarak ayarlıyoruz.

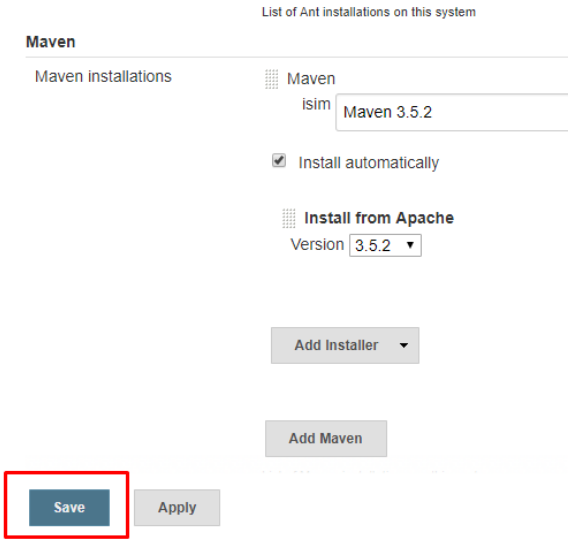


Figur 123

Figur 124

Figur 125 İçin Adımlar

1. Bütün bu ayarları yaptıktan sonra “Save” ediyoruz.

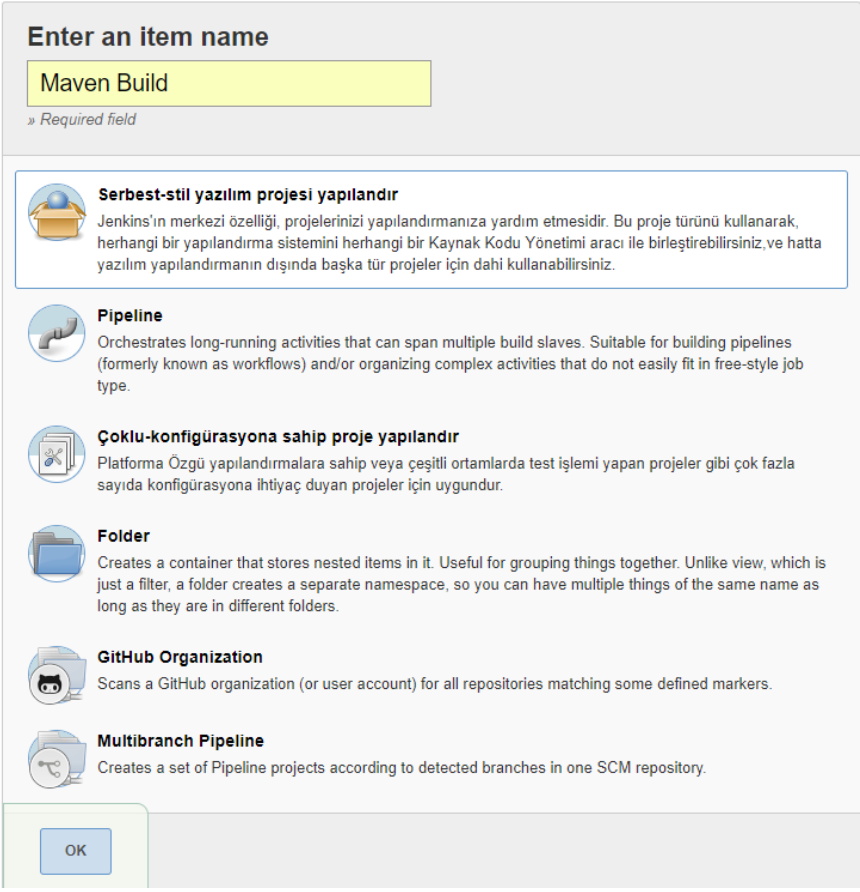


Figur 125

Figur 126 - 127 İçin Adımlar

1. Anasayfada sol taraftan “Yeni Item” seçeneğini seçiyoruz.Karşımıza “Figur 127” gelir.
2. İsim olarak herhangi bir şey girebiliriz ben “Maven Build” girdim.
3. “Serbest-stil yazılım projesi yapılandır” seçeneğini seçip “OK” butonuna basıyoruz.
4. Karşımıza “Figur 128” gelir.

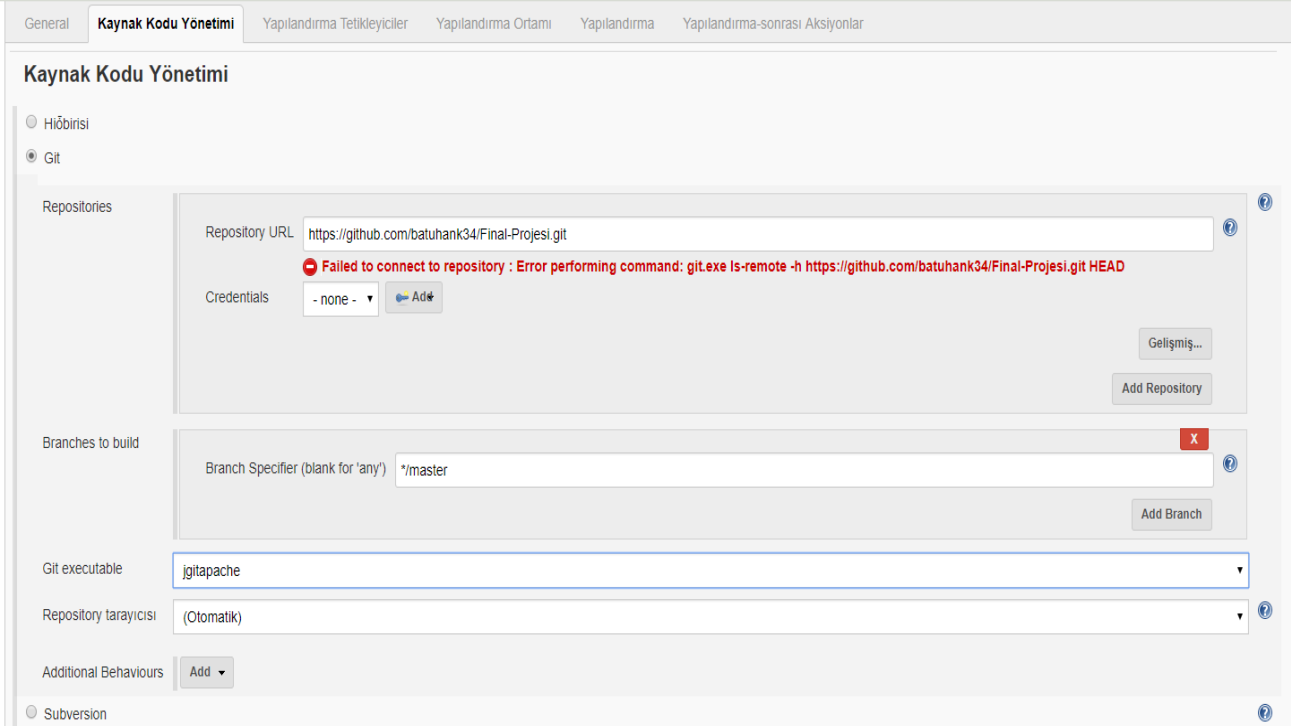
Figur 126



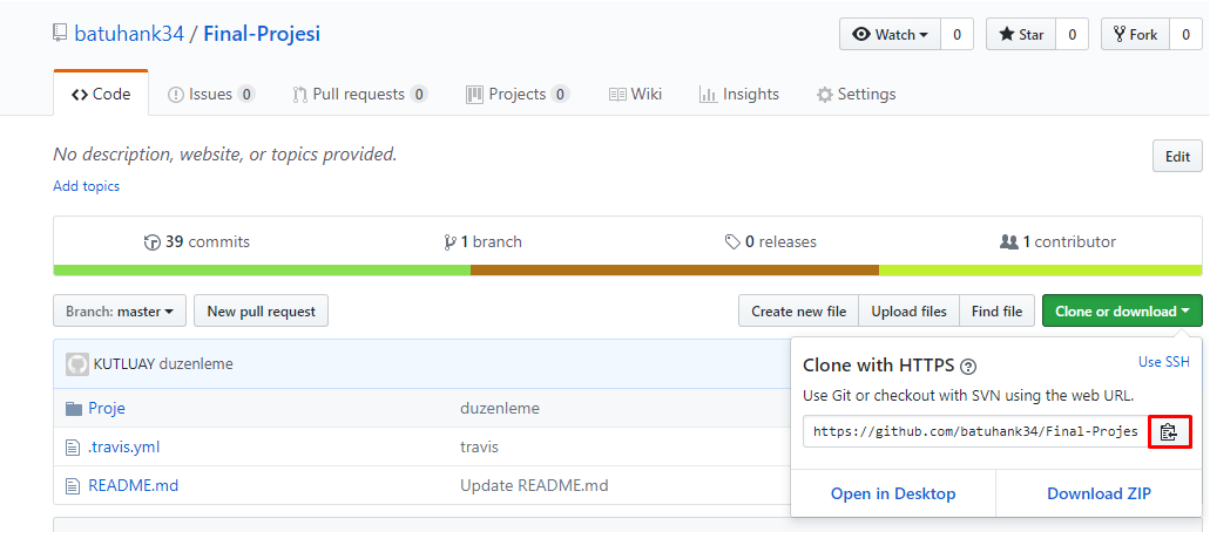
Figur 127

Figur 128-129 İçin Adımlar

1. Sekmelerden “Kaynak Kodu Yönetimi”ni seçiyoruz.
2. “Git” seçeneğini seçiyoruz.
3. Repomuzun URL’sini bizden istiyor.Repomuza gidip URL’yi alıyoruz(Figur 129).
4. “Git executable” olarak “jgitapache” seçeneğini seçiyoruz.



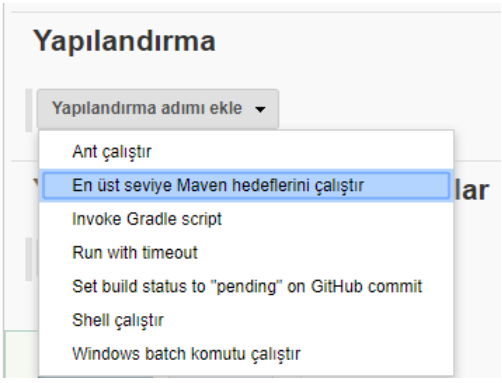
Figur 128



Figur 129

Figur 130 İçin Adımlar

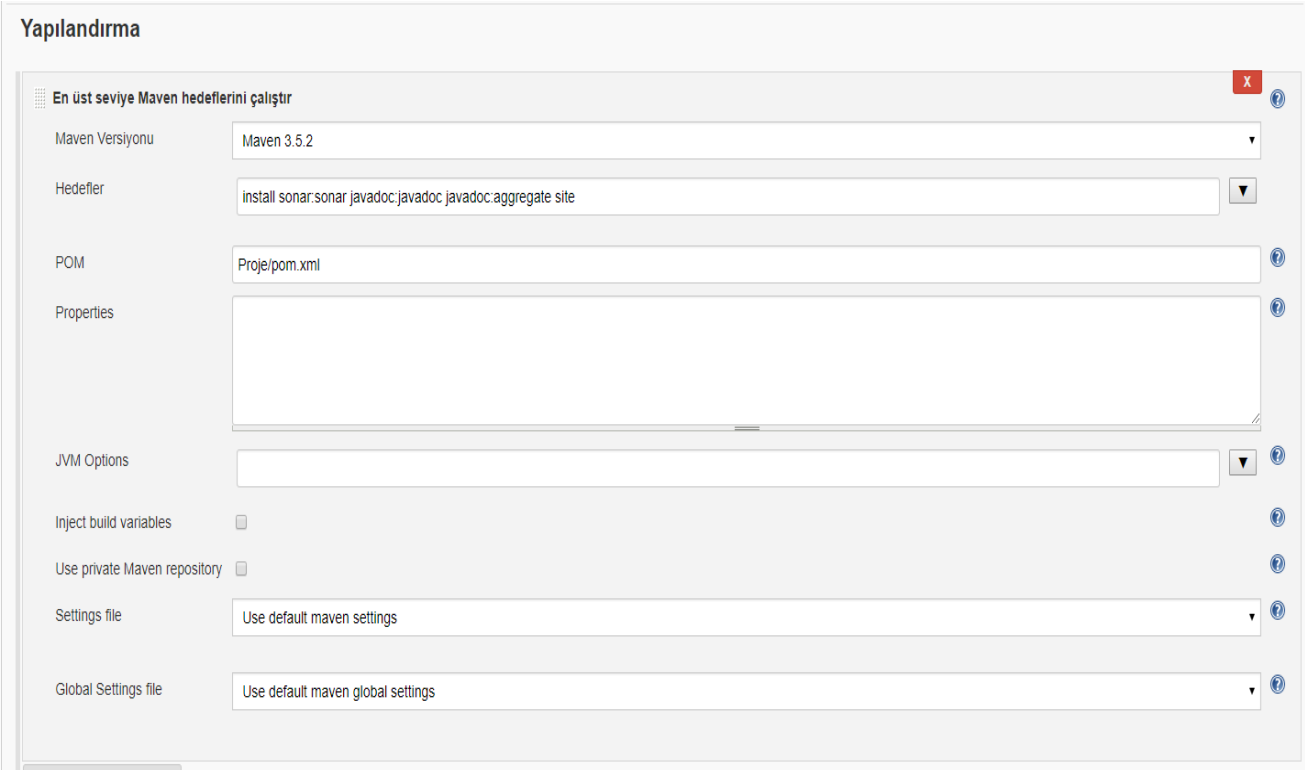
1. Sekmelerden “Yapılandırma” seçeneğini seçiyoruz.
2. Yapılandırma Adımı Ekle diyip “En üst seviye Maven hedeflerini çalıştır” seçenğini seçiyoruz.



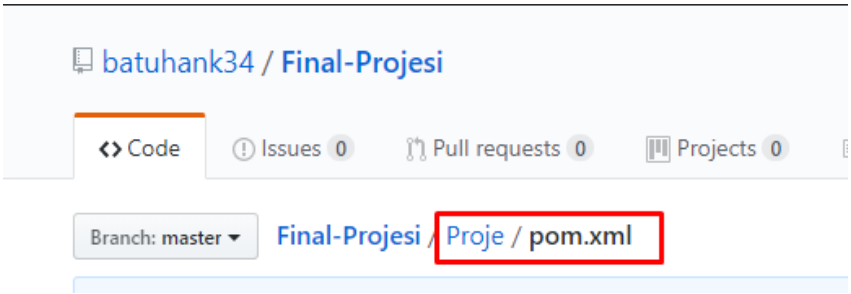
Figur 130

Figur 131-132-133 İçin Adımlar

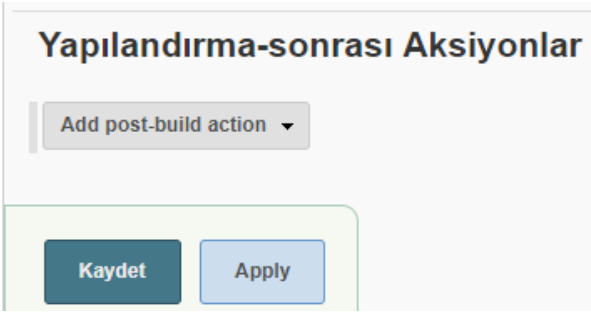
1. Yapılandırmanın gelişmiş ayarlarını açtığımız zaman karşımıza böyle bir ekran gelecektir.
2. “Maven Versiyonu” “Maven 3.5.5” olarak ayarlıyoruz.
3. Maven Goal olarak;SonarQube’ün çalışması,Javadoc ve Maven Site’in oluşması için hedefi Figurdeki gibi belirliyoruz.
4. “POM” kısmına projemizin pom yolunu veriyoruz.(Figur 132)
5. “Yapılandırma-sonrası Aksiyonlar” sekmesini seçiyoruz.”Kaydet” butonunu seçiyoruz.(Figur 133)



Figur 131



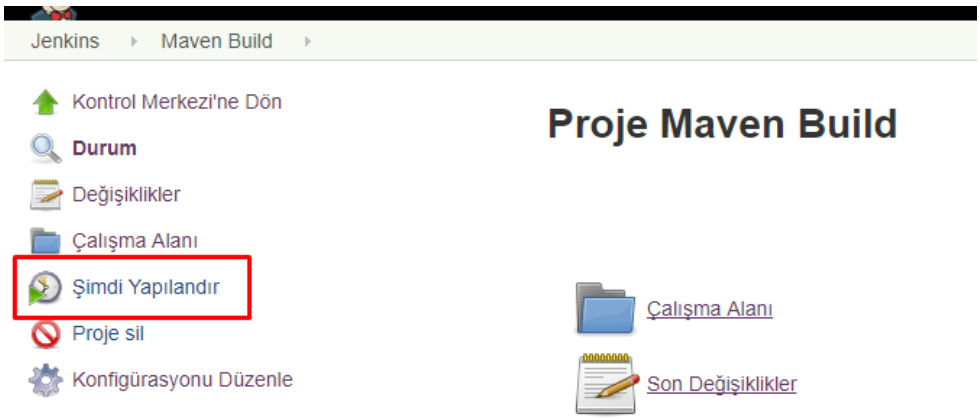
Figur 132



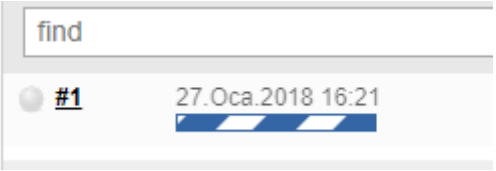
Figur 133

Figur 134 -135-136 İçin Adımlar

1. Oluşturduğumuz işe tıkladığımız zaman karşımıza böyle bir ekran gelir.Sol taraftan "Şimdi Yapılandır" seçeneğini seçiyoruz.
2. İşlem başlar.(Figur 135)
3. İşlem bittiğinde "Figur 136" karşımıza gelir."#1"e tıklarsak karşımıza "Figur 137" gelir.



Figur 134

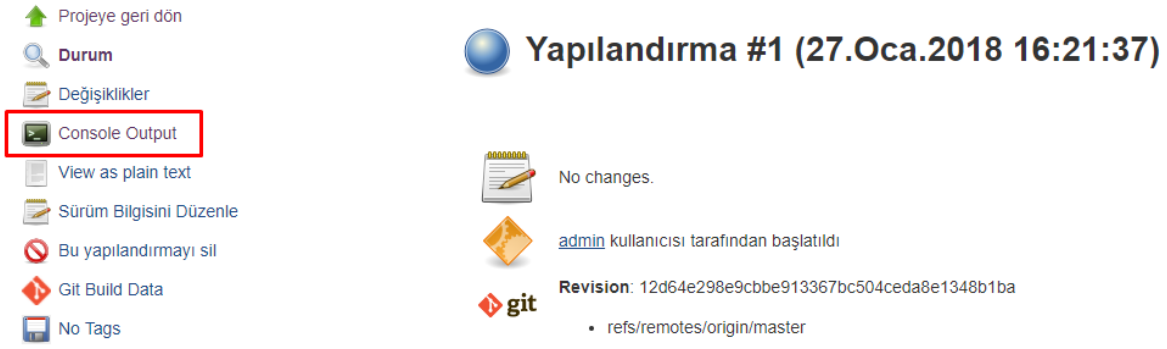


Figur 135

Figur 136

Figur 137-138 İçin Adımlar

1. “Soldaki menüden “Console Output” seçeneğini seçiyoruz.
2. Karşımıza konsol çıktısı gelir.Bu çıktının en sonuna gittiğimiz de “BUILD SUCCES” yazısını görebiliriz.(Figur 138)



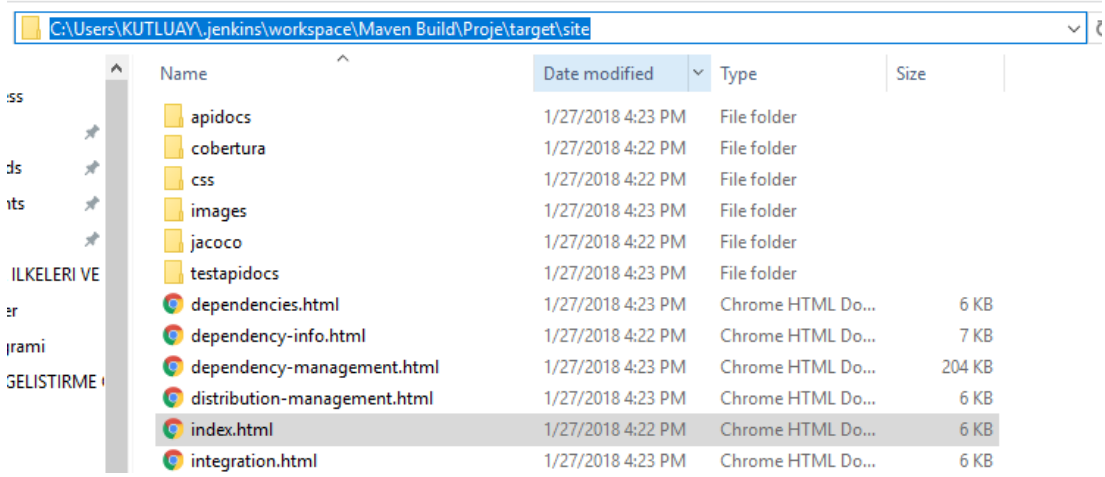
Figur 137

```
Generating C:\Users\KUTLUAY\.jenkins\workspace\Maven Build\Proje\target\site\testapidocs\help-doc.html..
[INFO] Generating "JaCoCo Test" report         --- jacoco-maven-plugin:0.7.6.201602180812:report
[INFO] Analyzed bundle 'Proje' with 3 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 46.189 s
[INFO] Finished at: 2018-01-27T16:23:13+02:00
[INFO] Final Memory: 76M/827M
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

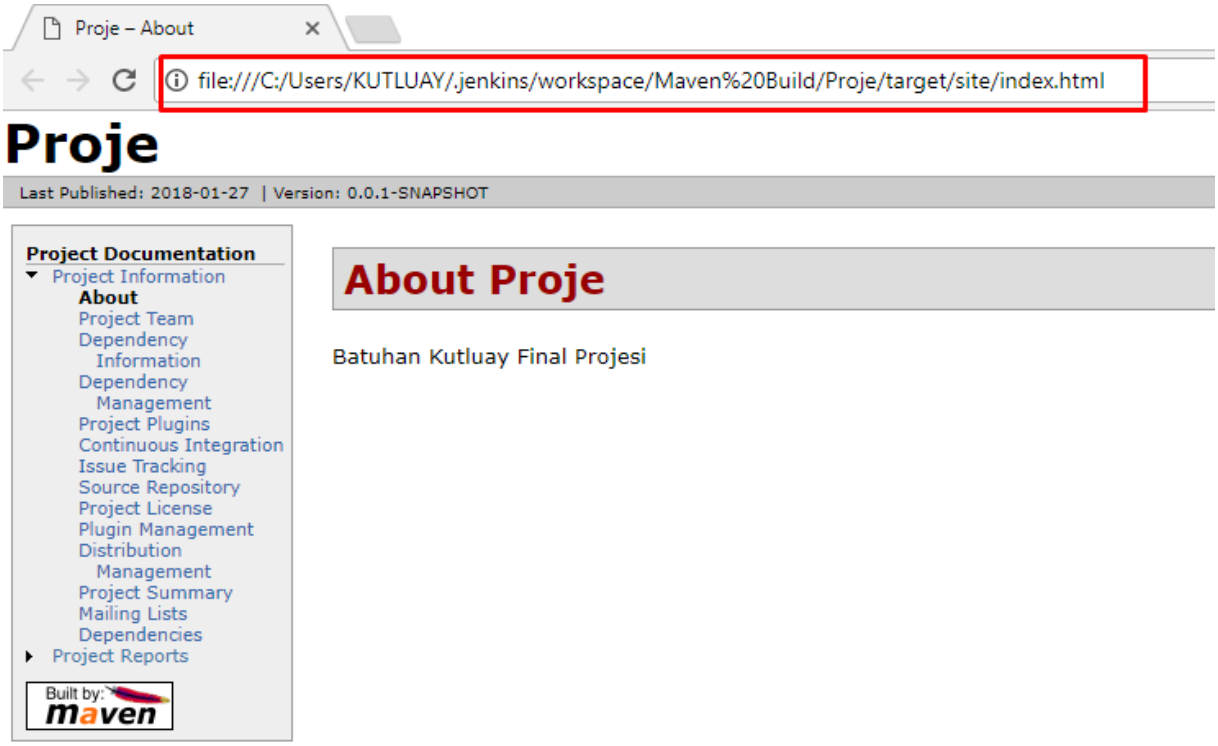
Figur 138

Figur 139 İçin Adımlar

1. Oluşturulan Maven Site lokalimizde bulunur.
2. “index.html”i açtığımız zaman karşımıza Figur 140 gelir.



Figur 139



Figur 140

14.1. SonarQube Test Sonuçları

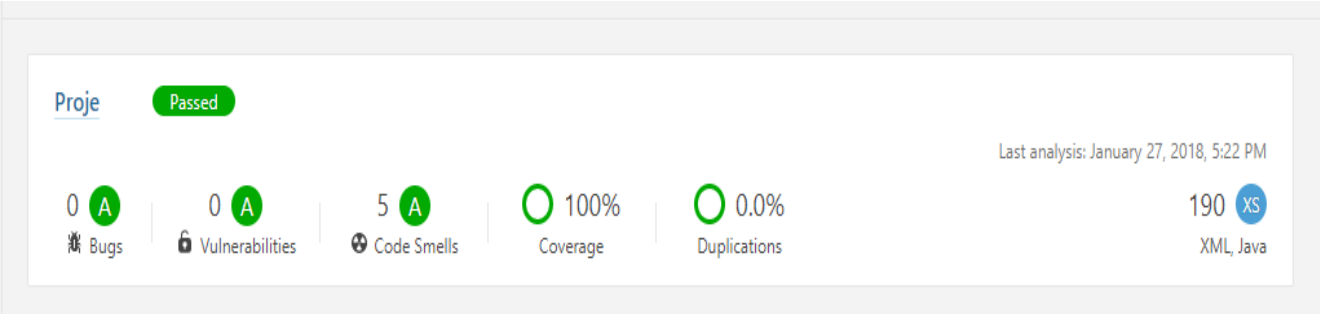
Figur 141 İçin Adımlar

1. localhost:9000 sitesine giriyoruz.
2. 1 Projenin analiz edildiğini görebiliyoruz.
3. “1” yazan yere tıkladığımız zaman karşımıza Figur “142” gelir.

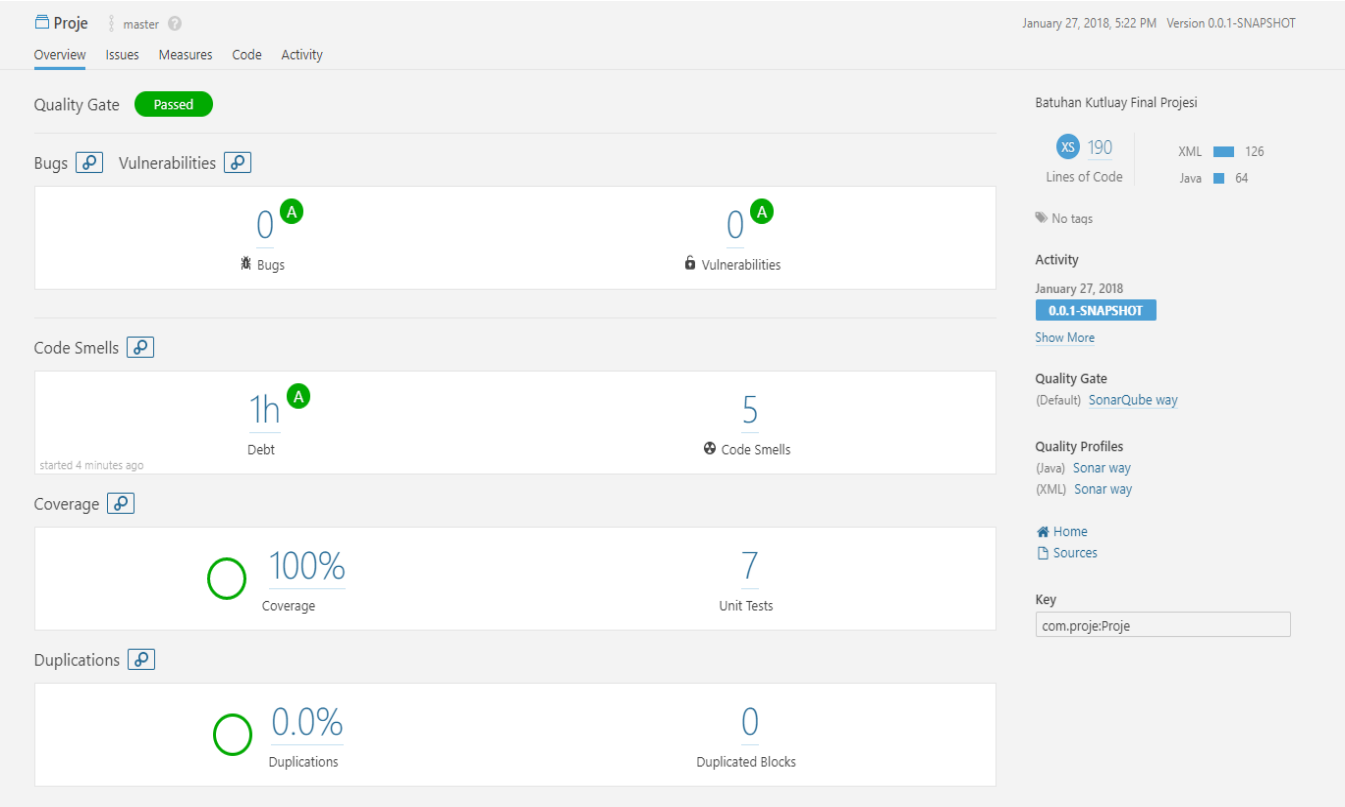
Figur 141

Figur 142 İçin Adımlar

1. “Proje” ye tıkladığımız zaman bize kodumuzun kalitesini göstercektir.(Figur 143)



Figur 142



Figur 143