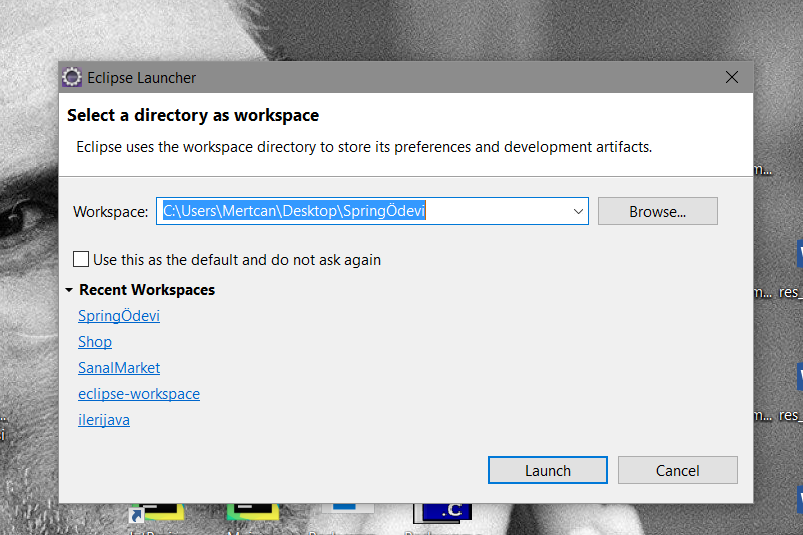
**YAZILIM ORTAM ve ARACLARI RAPORU**

**İçindekiler:**

1. **Projenin yapım aşaması.**
2. **Github’a yeni bir repository oluşturularak yapılan projeyi commitleme.**
3. **Javadoc oluşturulması.**
4. **Travis kurulumu.**
5. **UML diagramı.**

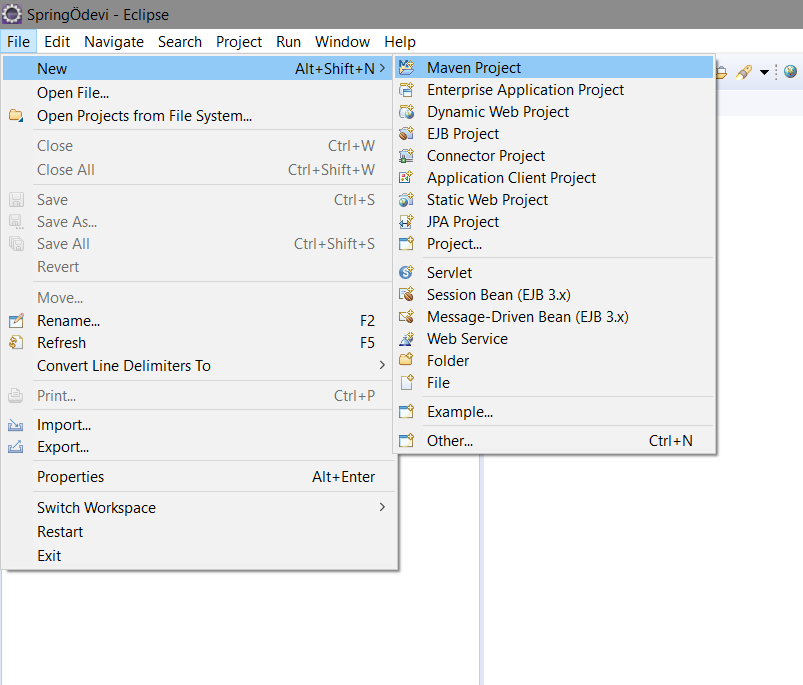
# **spring boot projesinin oluşturulması**

Öncelikle kullandığım eclips programımı açıp gelen kısıma projeye vermek istediğimiz adı yazıp, oluşturmak istediğimiz yeri belirtip programı başlatıyoruz.(Resim 1.1)



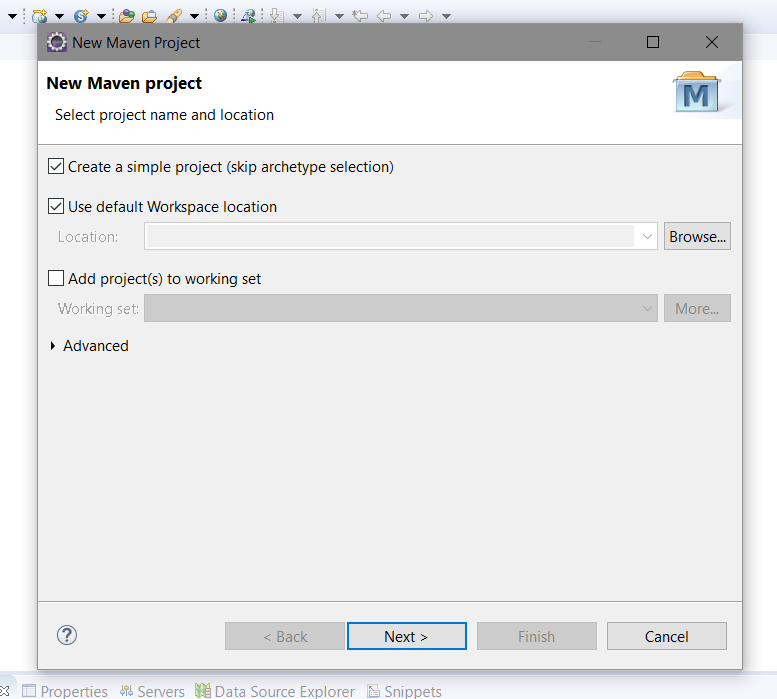
(Resim 1.1)

Daha sonra file sekmesinden maven project i seçip yeni bir maven projesi oluşturuyoruz. (Resim 1.2)



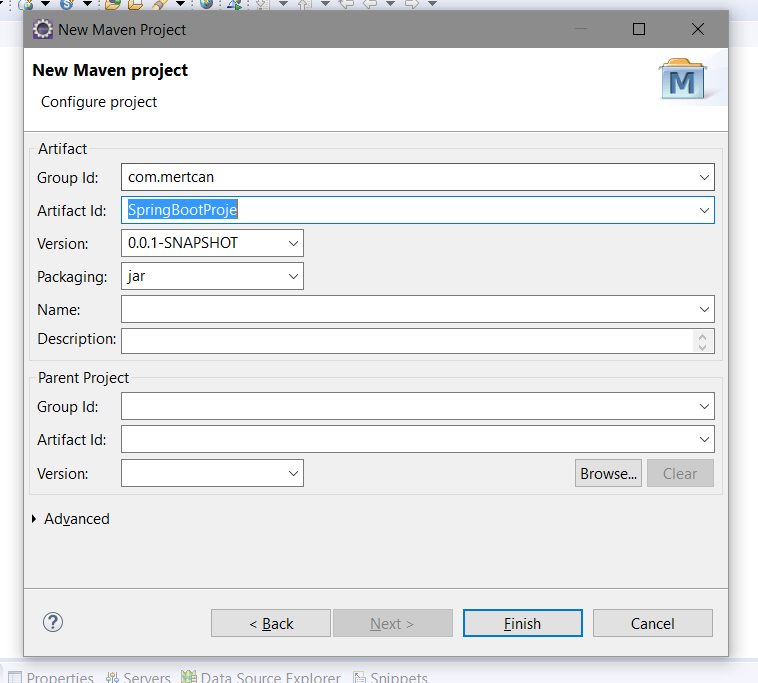
(Resim 1.2)

Oluşturduğumuz yeni maven projesine create a simple project seçeneğini işaretleyip next diyoruz.(Resim 1.3)



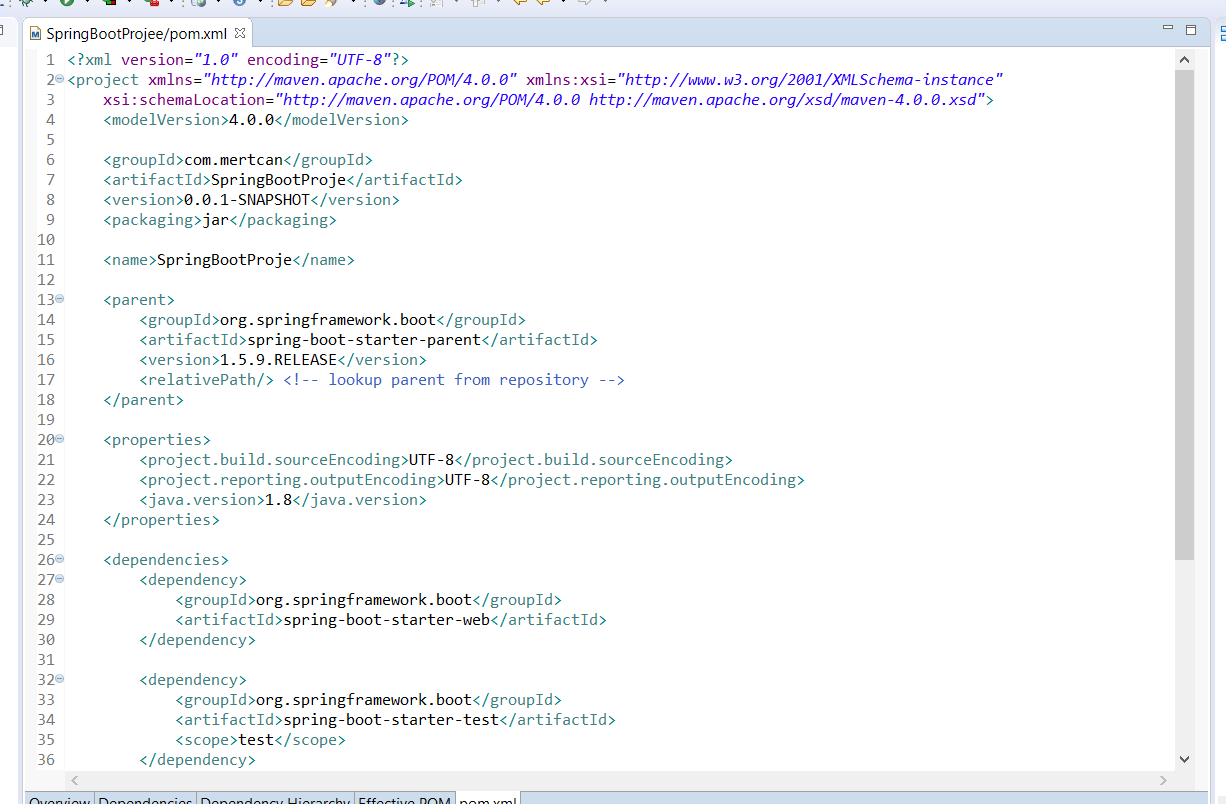
(Resim 1.3)

Daha sonra karşımıza gelen projemizle ilgili ayar sayfasını dolduruyoruz ve ayarlarımızı tamamladıktan sonra finish butonuna basıp ayarlarımızı tamamlayıp projemize başlıyoruz. (Resim 1.4)



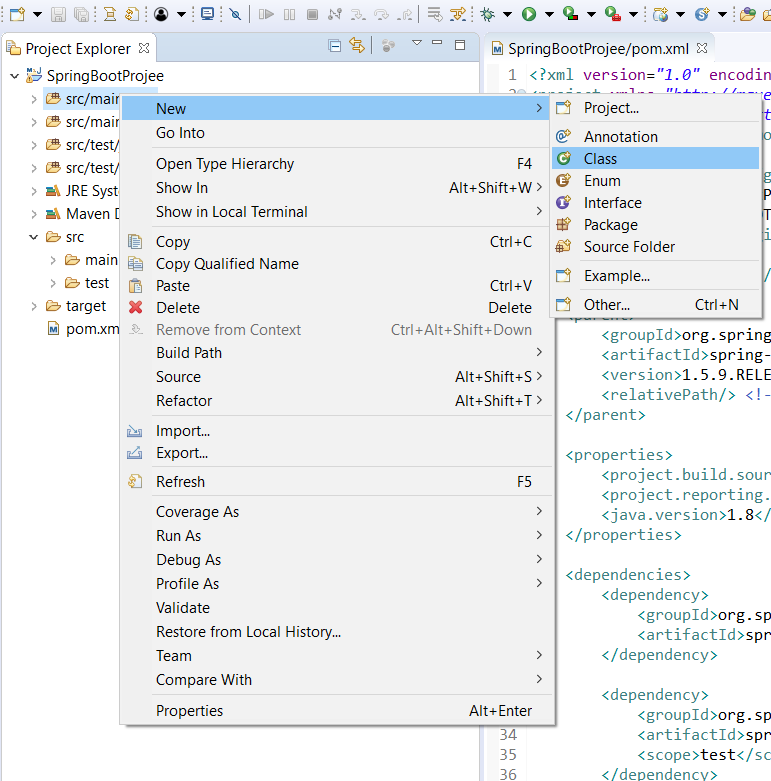
(Resim 1.4)

Açılan projemizde pom.xml sayfamızı açıyoruz. Pom.xml yapacağımız bütün gerekli ayarları yaptığımız sayfamızdır. Bir nevi ayarların yapıldığı sayfa diyebiliriz. Burada gerekli ayalarımızı yapıyoruz.(Resim 1.5)



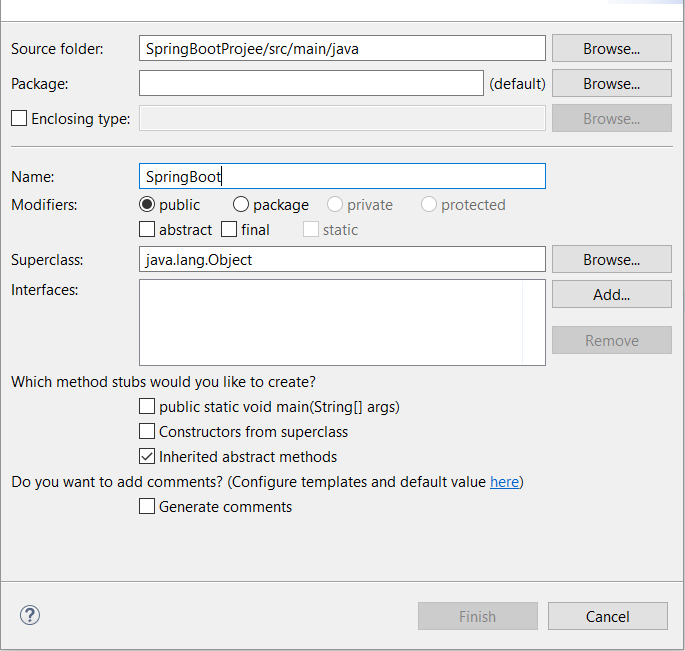
(Resim 1.5)

Pom.xml ayarlarımızı yaptıktan sonra projemizi yapılandırma kısmına geçiyoruz. src/main içerisine bir clas oluşturuyoruz.(Resim 1.6)



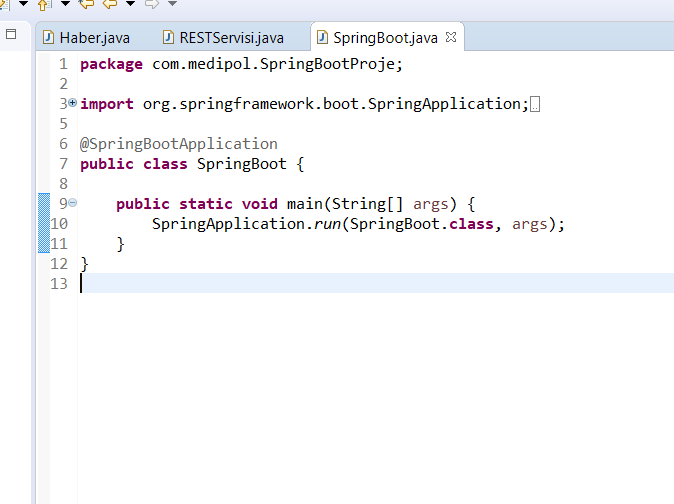
(Resim 1.6)

SpringBoot adında ana sınıfımızı oluşturup, bu sınıfta gerekli bağlantı ve kütüphanelerimizi oluşturup gerekli kodlarımızı yazıp diğer sınıfları oluşturma kısmına geçiyoruz.(Resim1.7)



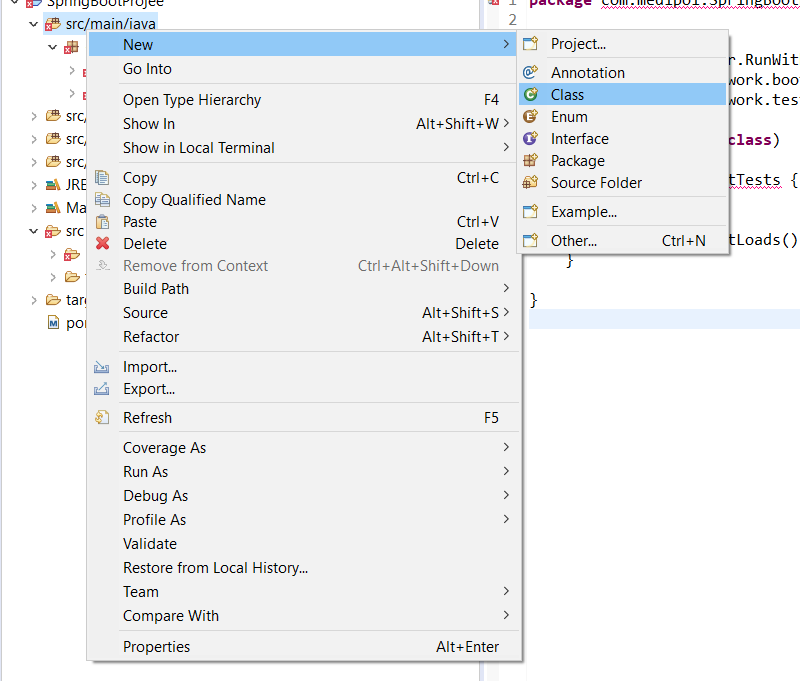
(Resim 1.7)

Gerekli bağlantıların kurulduğu ana sınıfımızın kodlarını tamamlıyoruz.(Resim 1.8)



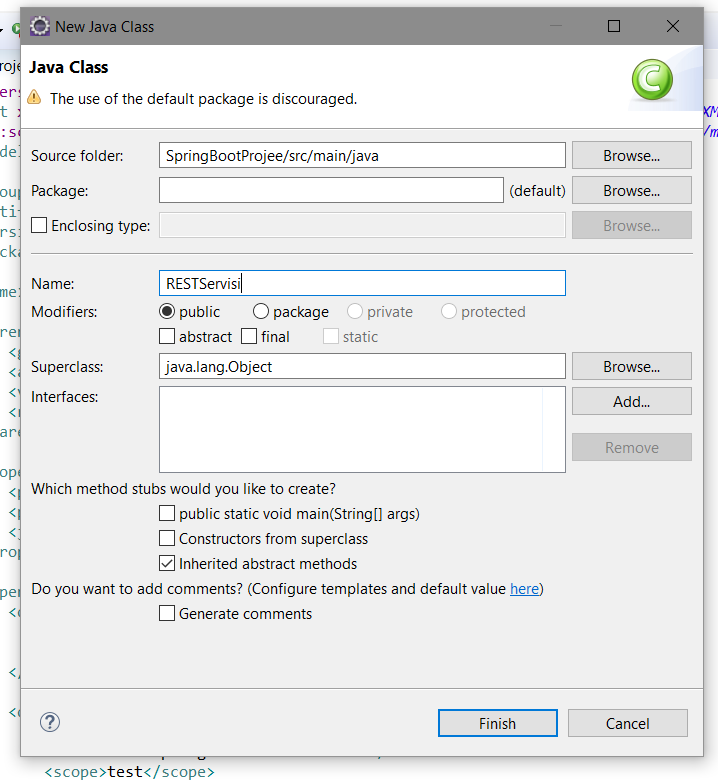
(Resim 1.8)

Daha sonra rest servis kodlarımızı yazmak için yeni bir sınıf oluşturuyoruz.(Resim 1.9)



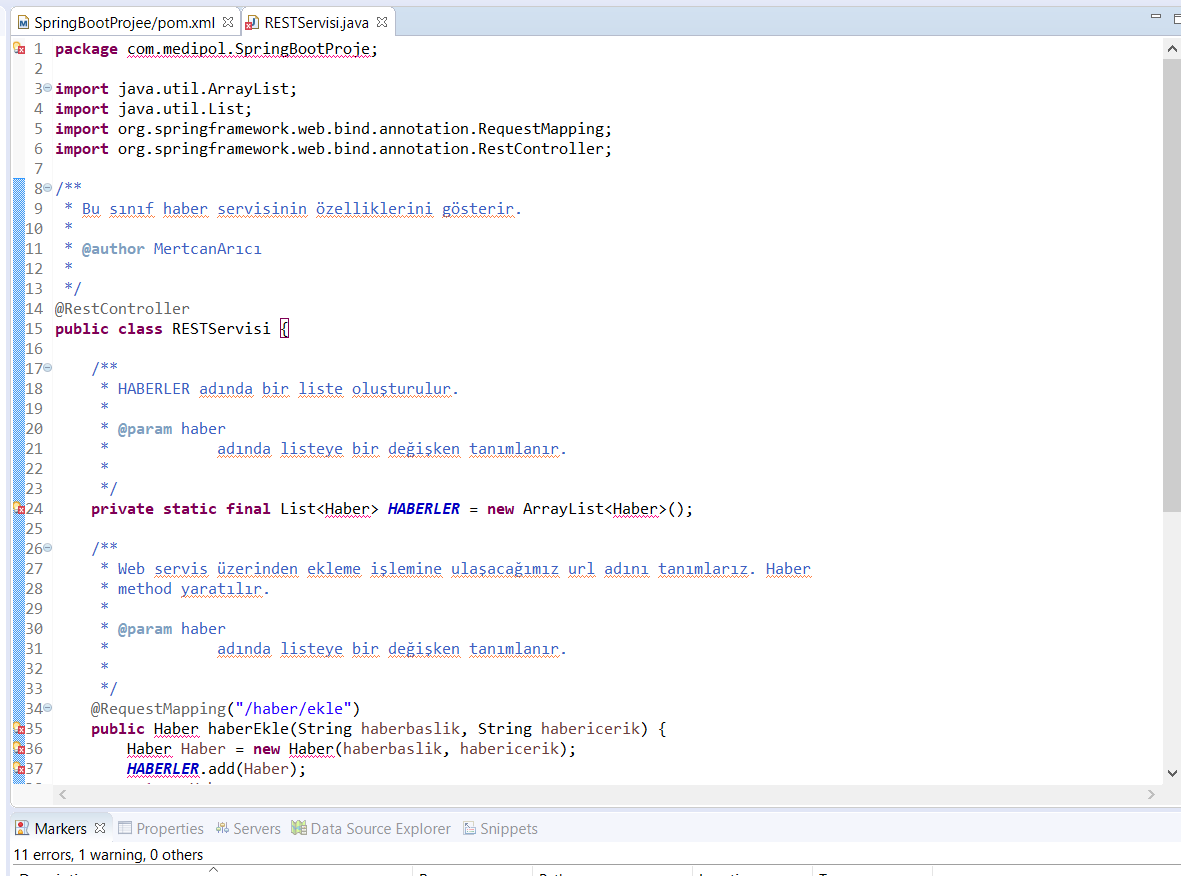
(Resim 1.9)

Karşıma gelen yeni sınıf oluşturma kısmına rest servis adını verip bu kısıma rest servisimizin kodlarını yazacağız. Sınıfın ismine RESTServisi adını verip finish butonuna basıp sınıfımızı oluşturuyoruz.(Resim 1.10)



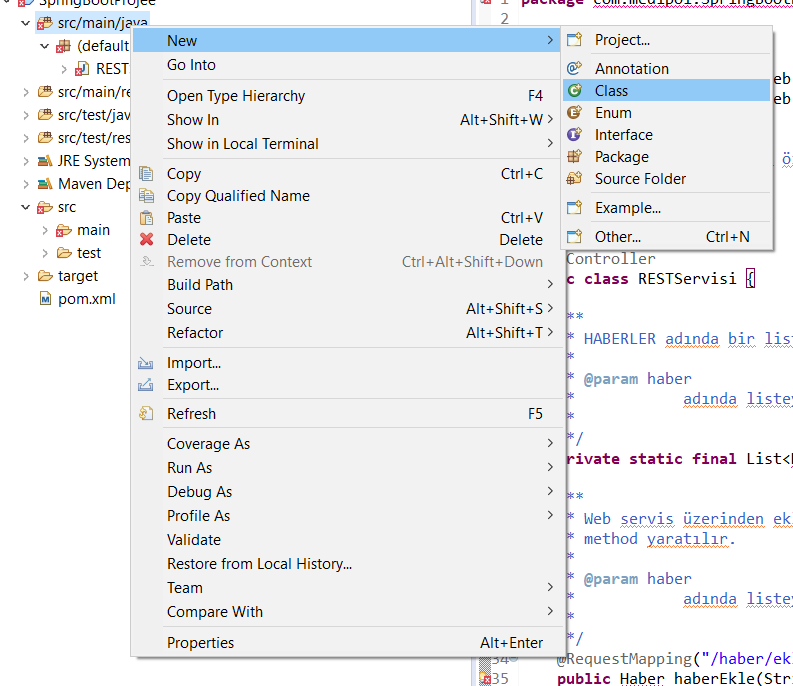
(Resim 1.10)

Oluşturduğumuz rest servisi sınıfımızın içine kodlarımızı yazıyoruz. Bu kısma yazacağımız rest servisi yani web sayfasındaki gerçekleştirdiğimiz işlemlerin kodlarını yazıyoruz.(Resim 1.11)



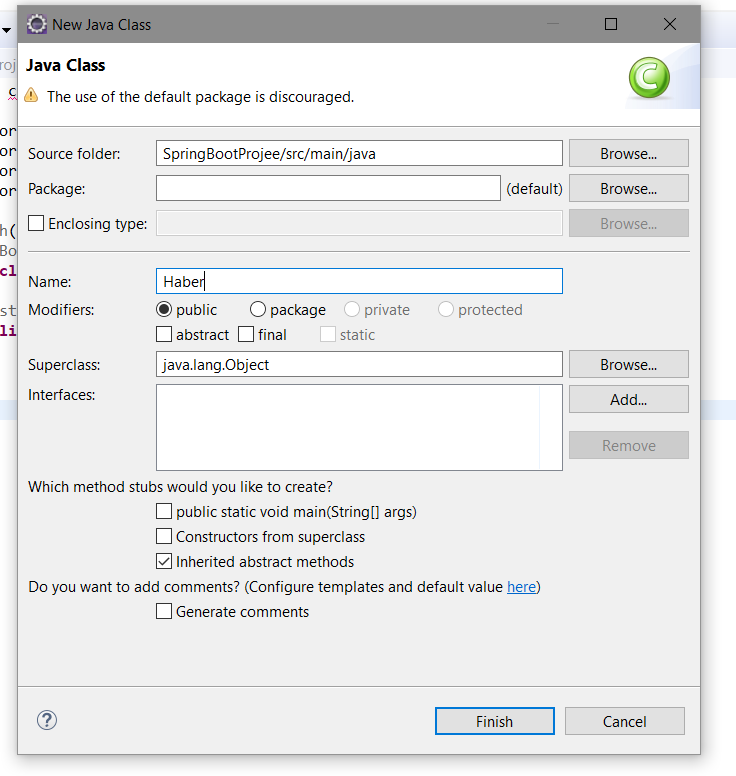
(Resim 1.11)

RESTServisi sınıfı bittikten sonra yeni bir sınıf oluşturup orada haber verilerimizi oluşturacağız. (Resim 1.12)



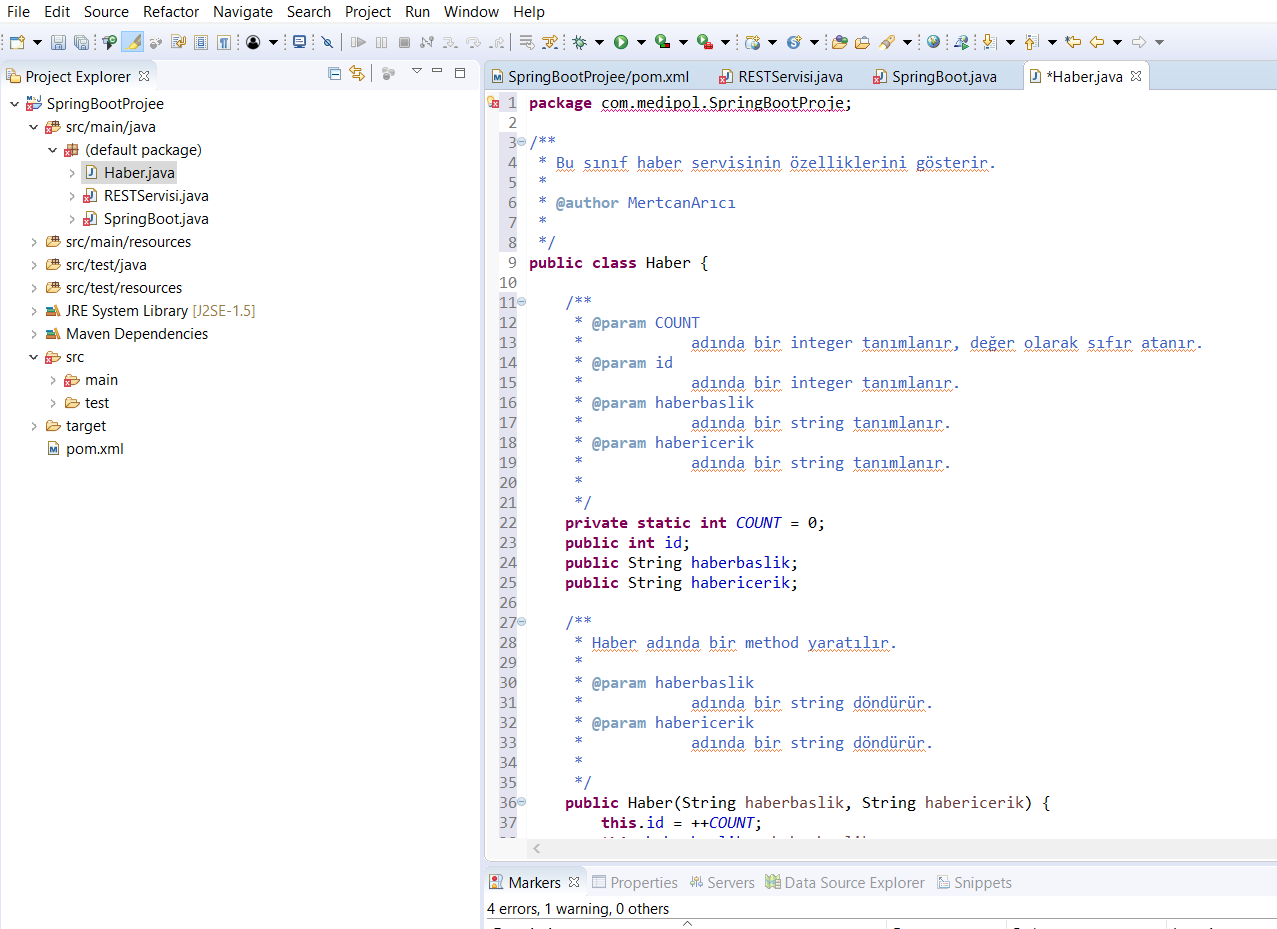
(Resim 1.12)

Sınıfımızın ismini ve ayarlarını yapıp oluşturuyoruz.(Resim 1.13)



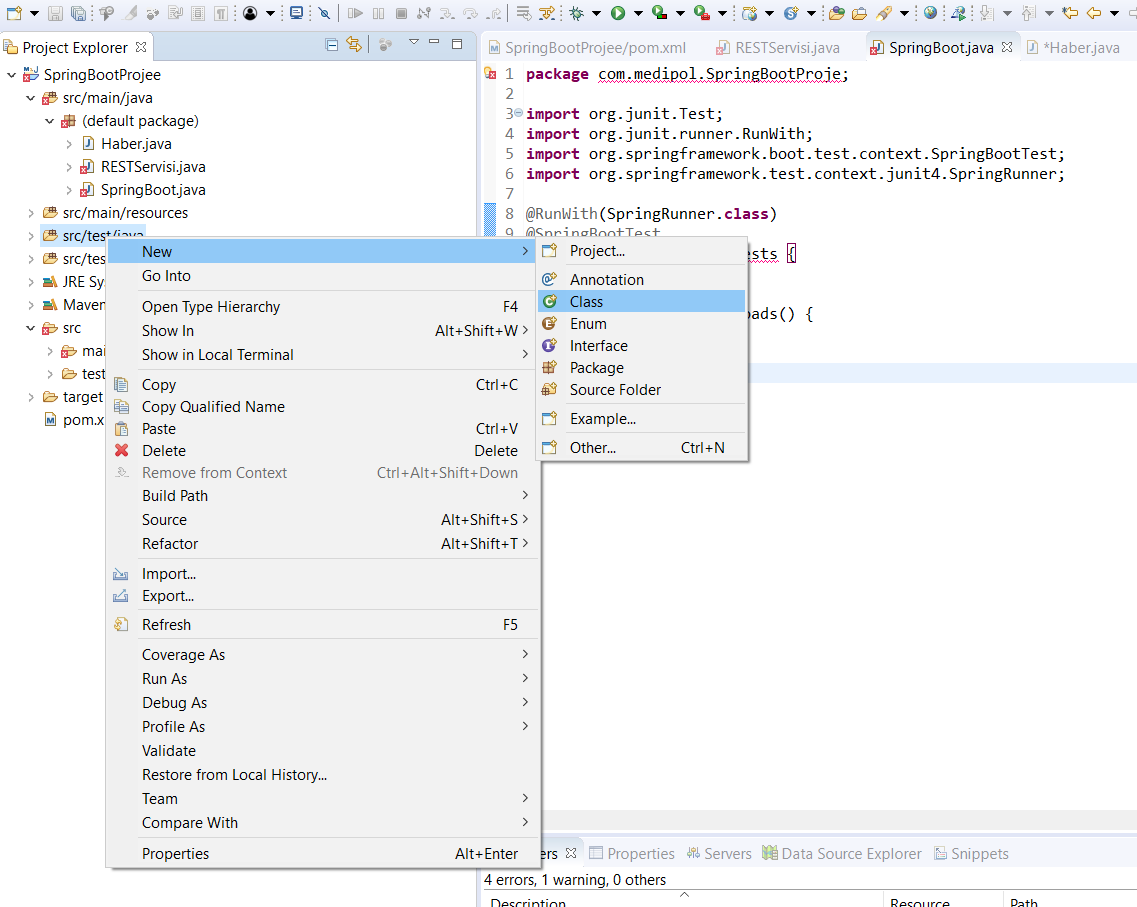
(Resim 1.13)

Haber adlı sınıfımızın değişkenlerini içeriğini oluşturuyoruz.(Resim 1.14)



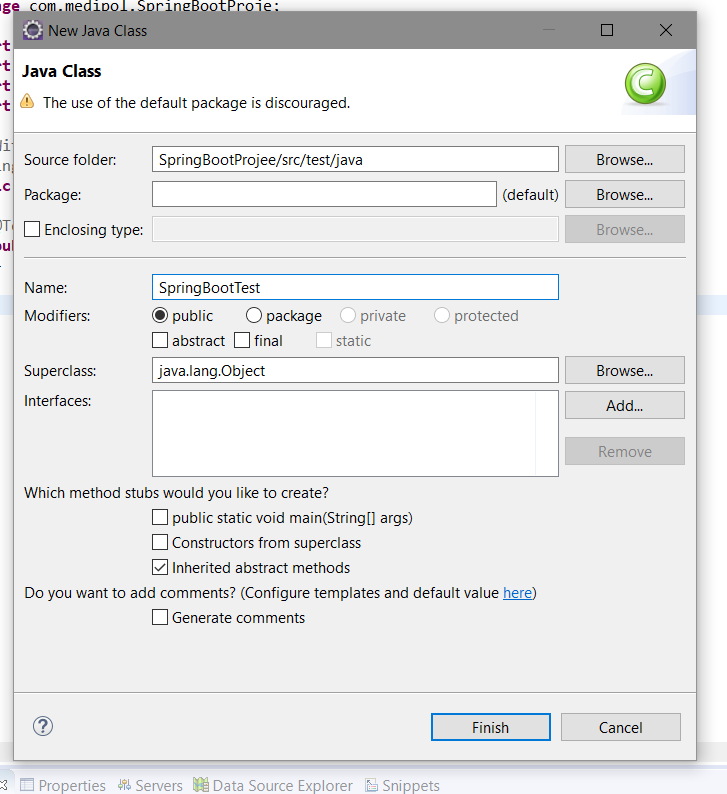
(Resim 1.14)

Yaptığımız kodların test sınıflarını oluşturmak için yeni sınıf oluşturuyoruz. Bu sınıfı src/test/java nın içinde oluşturuyoruz.(Resim 1.15)



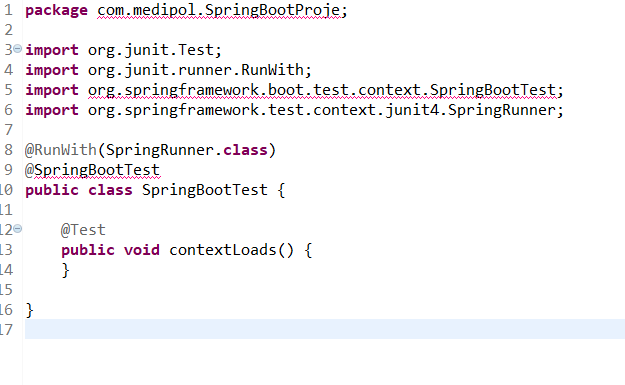
(Resim 1.15)

SpringBootTest adını verip sınıfımızı oluşturuyoruz. (Resim 1.16)



(Resim 1.16)

Oluşturduğumuz test sınıfımız SpringBoot sınıfını test etmek için gerekli kodlama kısmını yapıyoruz. (Resim 1.17)

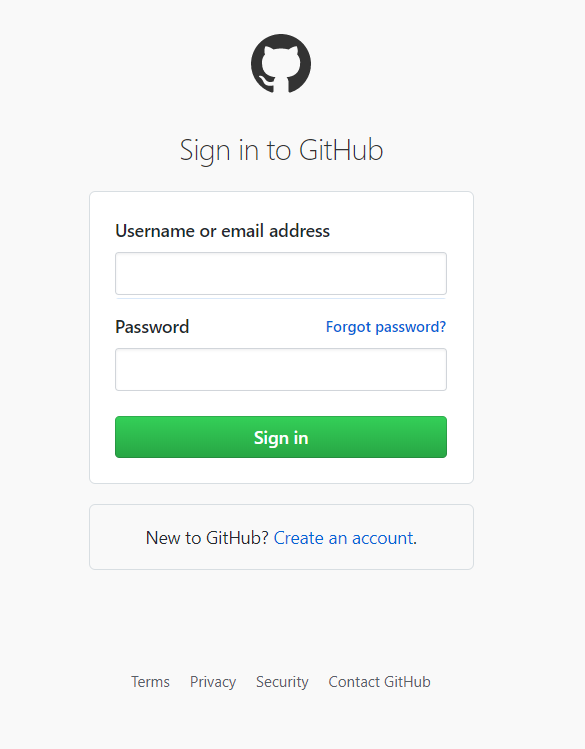


(Resim 1.17)

Projemiz için gerekli olan kodlamayı bitirmiş bulunuyoruz. Gerekli sınıflar oluşturulmuştur.

# **github’a yeni bir repository oluşturularak yapılan projeyi commitleme**

Öncelikle github hesabı oluşturuyoruz, hesabımız varsa kullanıcı adı ve parola girerek hesabımıza giriş yapıyoruz.(Resim 1.1)



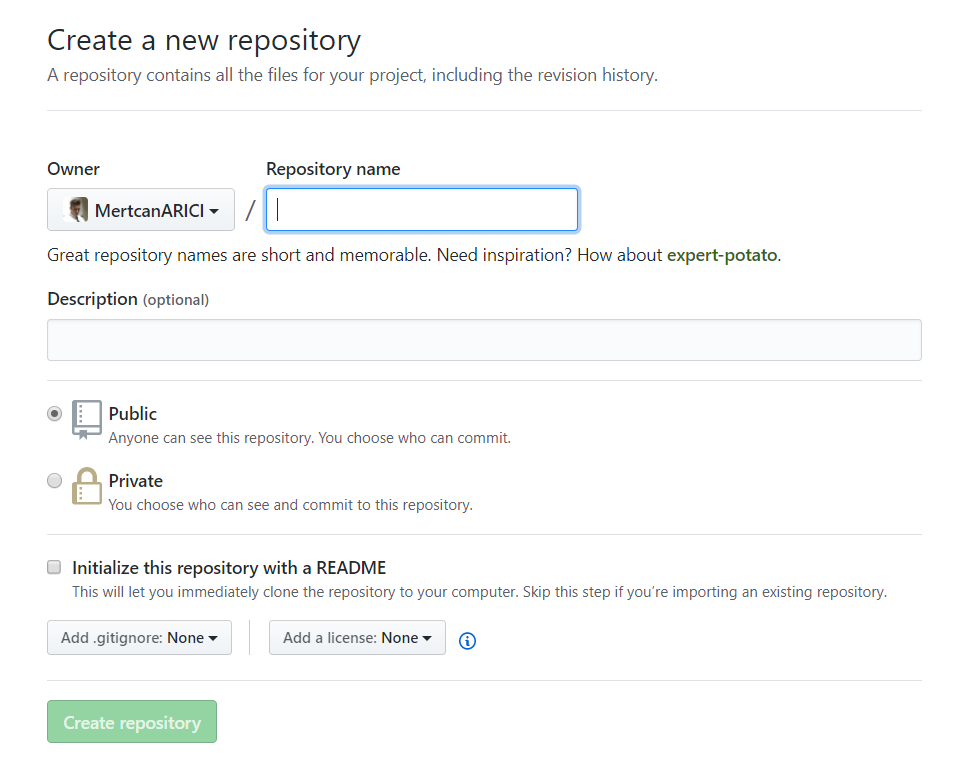
(Resim 1.1)

Oluşturduğumuz hesabın profil buttonunun yanında bulunan + simgesine tıklayarak new repository butonuna basarak yeni bir repository oluştuyoruz. (Resim 1.2)



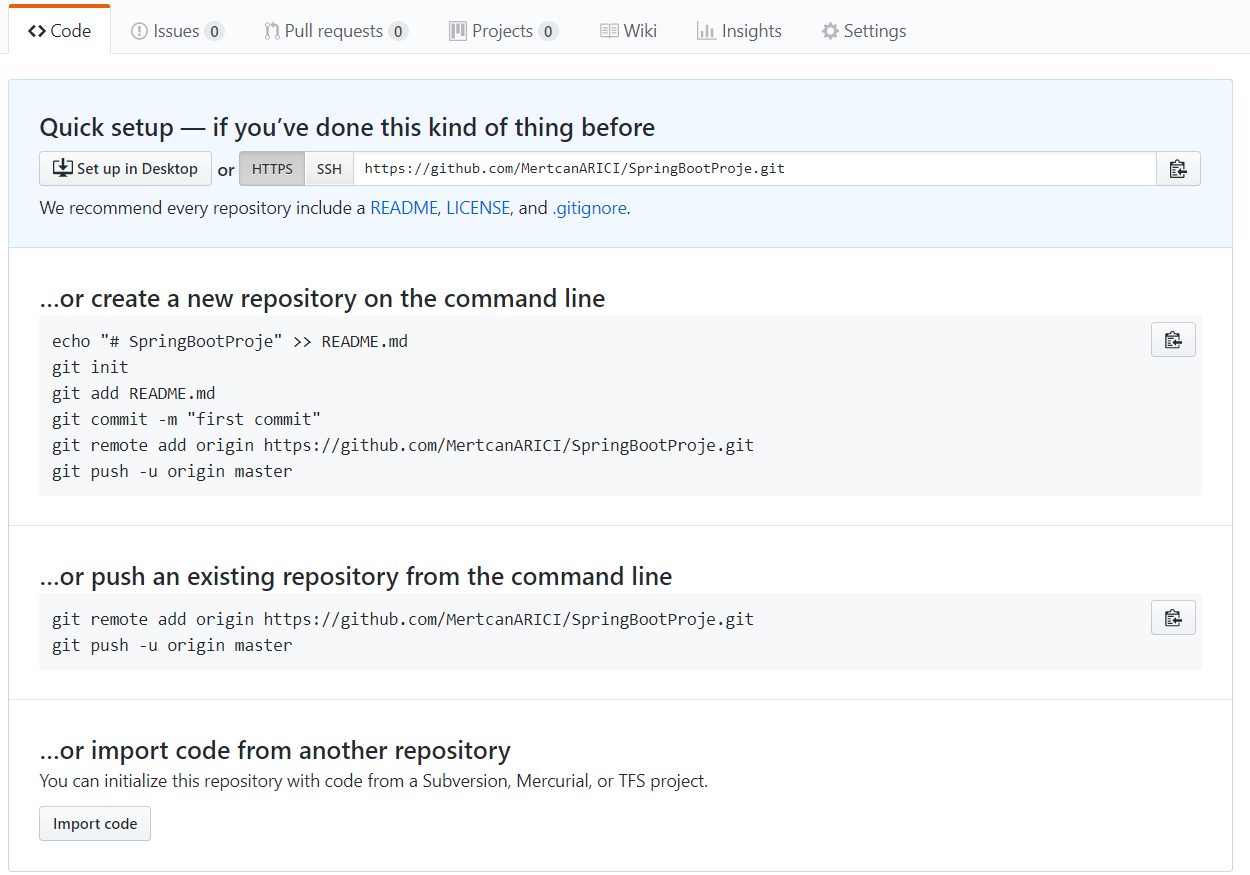
(Resim 1.2)

New repository seçeneğini seçtikten sonra karşımıza oluşturacağımız reponun hangi özellikleri barındırmasını istediğimizi ve isminin ne olmasını arzuluyorsak seçenekleri buna göre dolduruyoruz. Öncelikle repo name kımına projemizin ismini veriyoruz. Daha sonra description kımına alt başlık açıklama cümlesi yazıyoruz, public i seçip paylaşılabilir yapıyoruz ve create repository butonuna basarak repositorimizi oluşturmuş oluyoruz. (Resim 1.3)



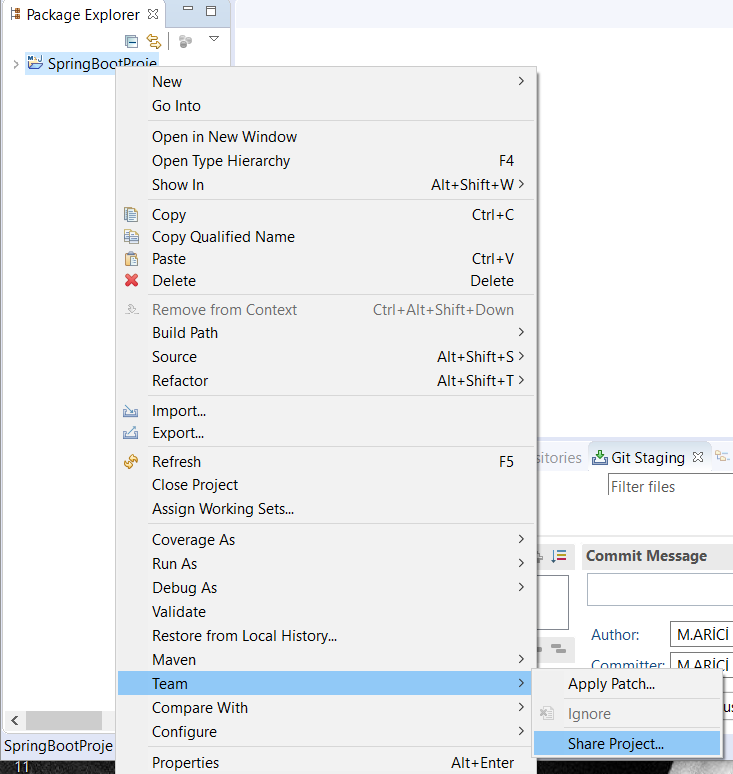
(Resim 1.3)

Bu resimde (1.4) oluşturduğumuz repository hakkında bilgilerimiz bulunmaktadır.



(Resim 1.4)

Repositomuzu oluşturduktan sonra eclipste bulunan projemizi github a aktarma aşamasına geçebiliriz. Projemize sağ tıklayığ Team seçeneğinin alt seçeneği olan Share project i seçiyoruz ve projemizi paylaşıyoruz. (Resim 1.5)



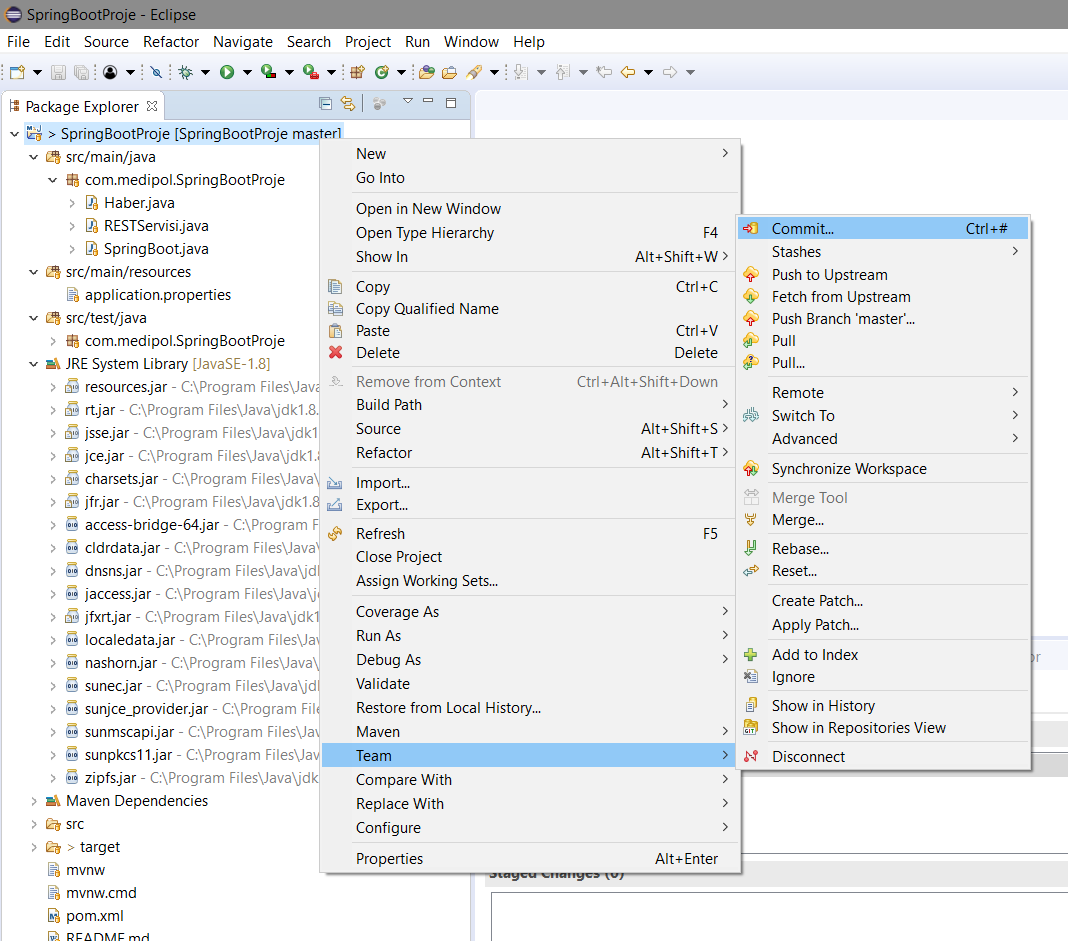
(Resim 1.5)

Çıkan seçeneği işaretliyip, projemizin hangi github reposuyla paylaşacağımızı seçip finish diyoruz. (Resim 1.6)



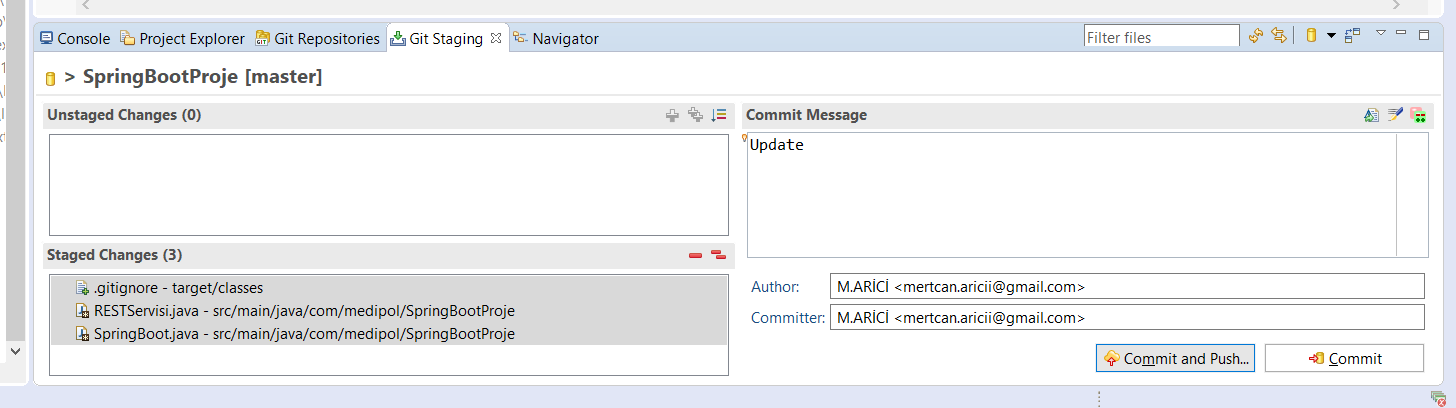
(Resim 1.6)

Projemizi paylaştıktan sonra sıra projemizi repoya commitleme işlemine(gönderme) geldi. Projemize tekrar sağ tık yapıp Team seçeneğini seçip commit seçeneğine tıklayarak projemizi commitleme komutu veriyoruz. (Resim 1.7)



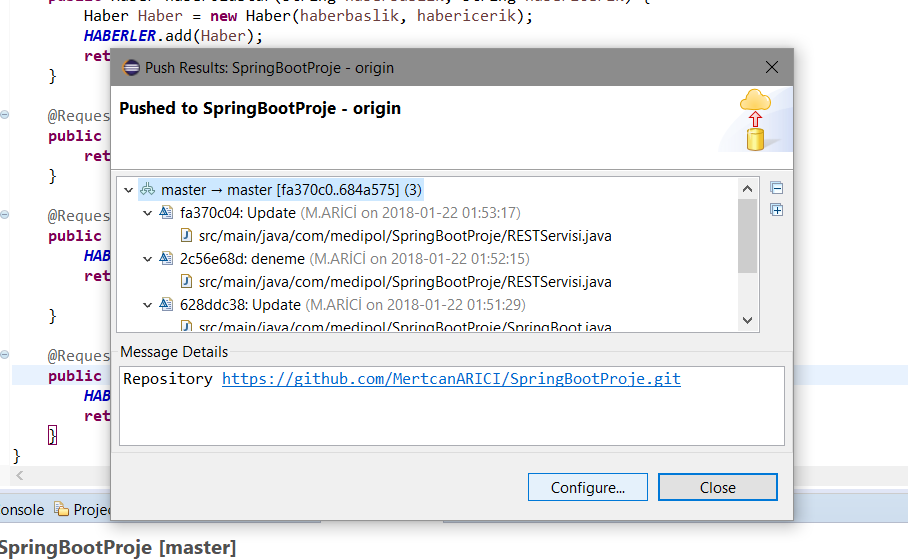
(Resim 1.7)

Commit seçeneğine tıklandıktan sonra karşımıza commitleme tablosu geliyor. Unstaged seçenekleri değişim yapılmış yada commitlenmemiş dosyalar bulunmaktadır. Bu dosyaları göndermek yani commitlemek istediğimiz tabloya sürüklüyoruz yani stagged changes tablosuna. Göndermek istediğimiz dosyaları sürükledikten sonra commit message yani gönderi mesajını yazıp commit and push butonuna tıklayarak projemizi repomuza gönderme işlemini başlatmış oluyoruz. (Resim 1.8)



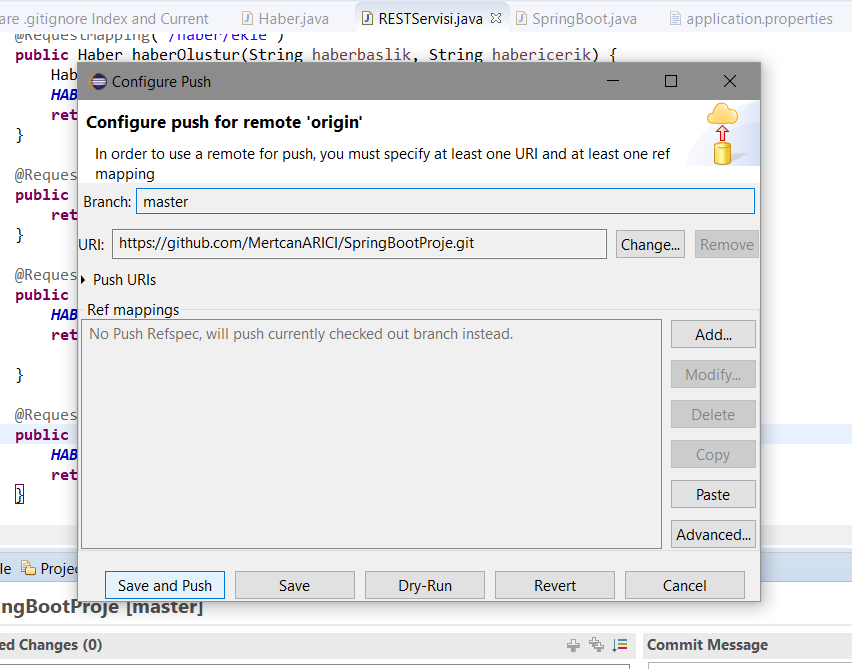
(Resim 1.8)

Commit and push dedikten sonra karşımıza çıkan bu ekranda bizi yapılan değişikliklerdeki dosyaları ve değişiklik ile ilgili bağzı bilgileri bize gösteriyor. Configure diyerek devam ediyoruz işleme. (Resim 1.9)



(Resim 1.9)

Son işlemimiz ise branchimizi seçip hangi url i gönderildiği bilgilerini barındıran tablo işlemleri. Bu bilgiler de onaylandıktan sonra save and push diyerek işlemimizi tamamlıyor ve projemizi github a göndermiş oluyoruz. (Resim 1.10)

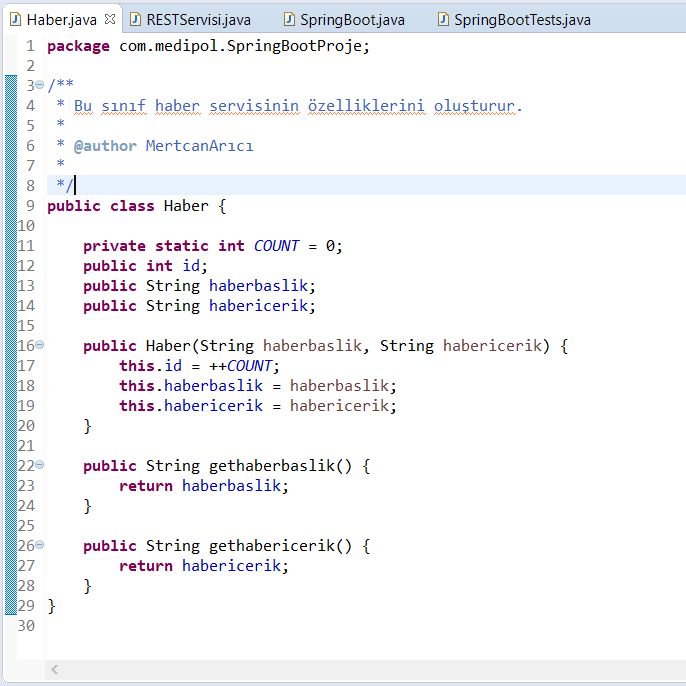


(Resim 1.10)

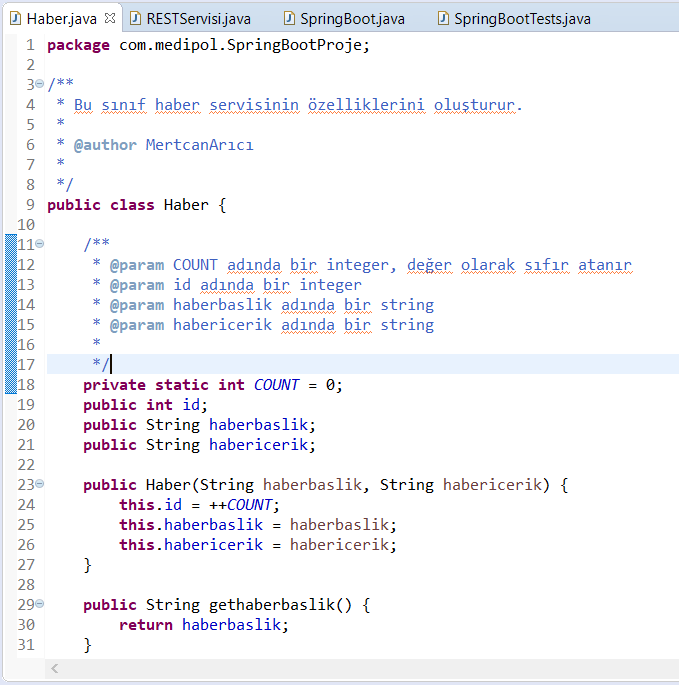
# **javadoc oluşturulması**

* **Haber.java**

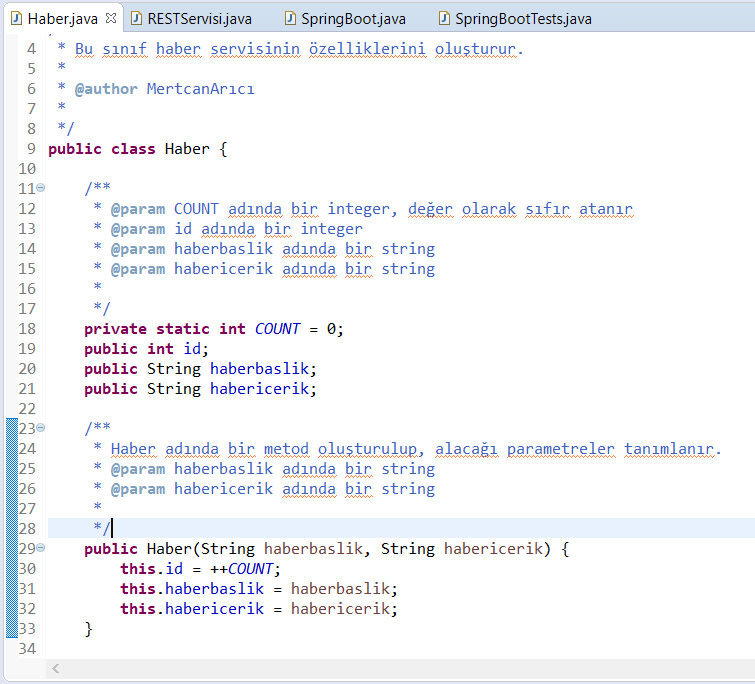
Sayfa için Javadoc hazırlarken ilk olarak Javadoc yorumlarını hazırlıyoruz. İlk olarak sınıfın ne işlem yaptığını ve hazırlayan kişiyi Javadoc etiketi @author ile tanımlıyoruz.



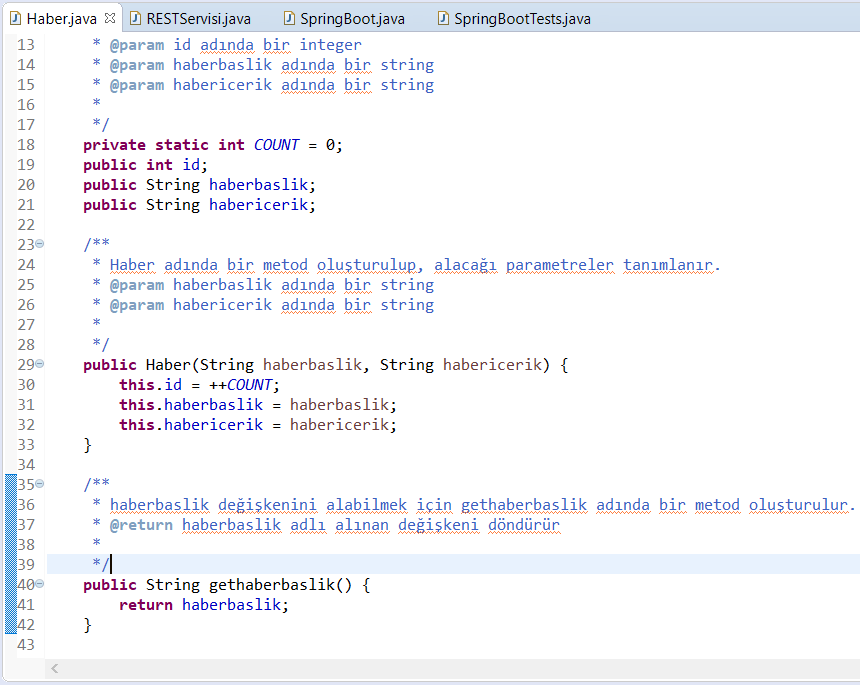
Sonra Javadoc @param etiketi ile bu sınıfın içindeki değişkenleri tanımlıyoruz.



Sonra bu sınıfın içindeki Haber metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını ve Javadoc @param etiketi ile hangi değişkenleri barındırdığını açıklıyoruz.



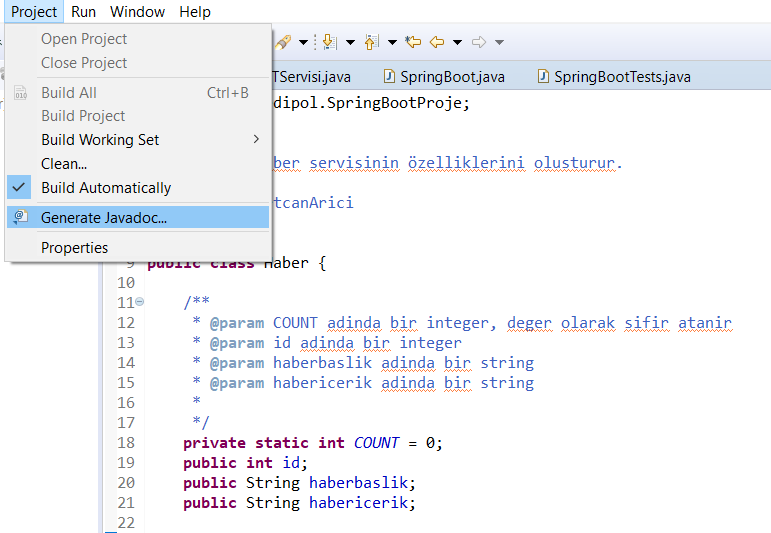
Sonra bu sınıfın içindeki değişken getter metodu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını ve Javadoc @return etiketi ile metodun döndürdüğü değeri açıklıyoruz.



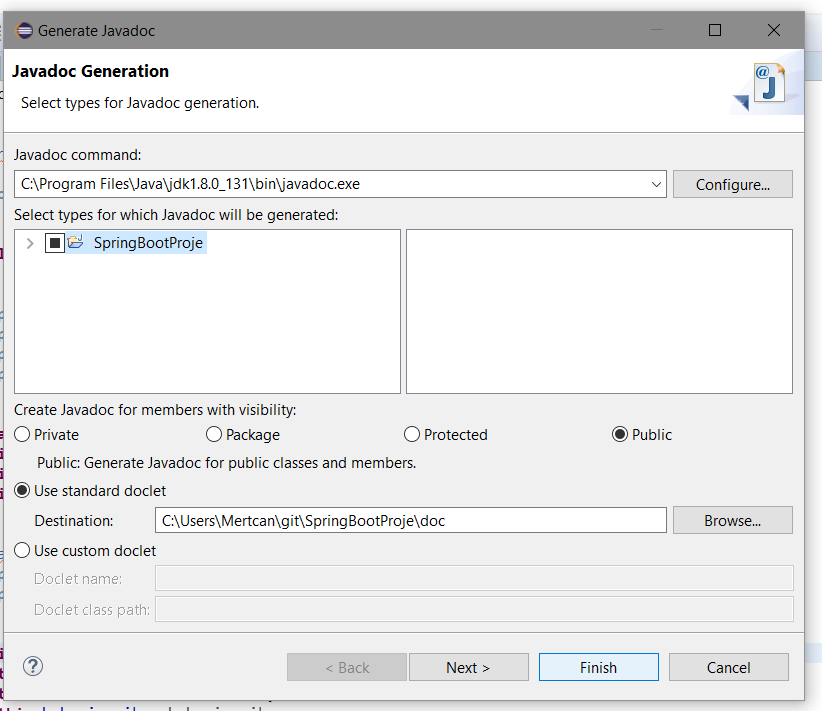
Sonra bu sınıfın içindeki diğer değişken getter metodu tanımlıyoruz. Tanımlarken yine metodun ne işlem yaptığını ve Javadoc @return etiketi ile metodun döndürdüğü değeri açıklıyoruz.



Javadoc yorumlarını hazırladıktan sonra html dökümü olarak almak için Project –> Generate JavaDoc kısmı seçilir.

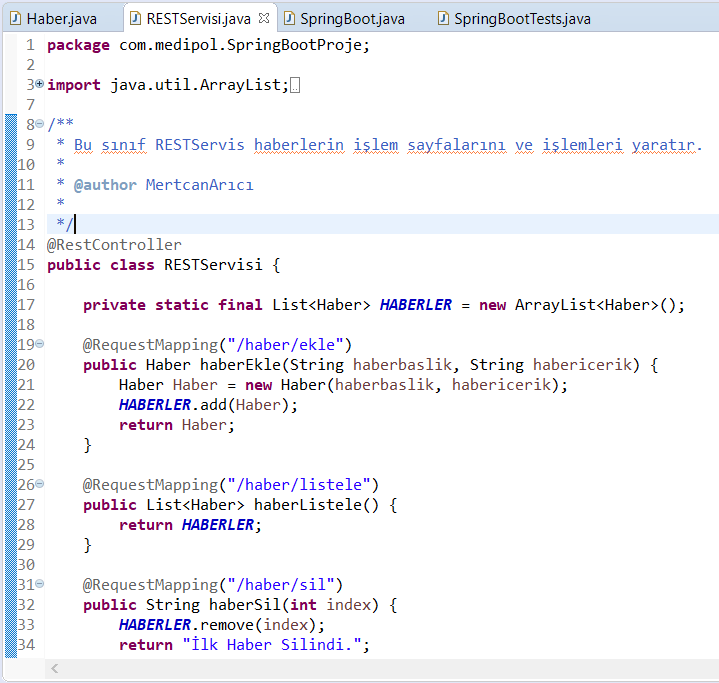


Son olarak da finish butonuna tıklarız.

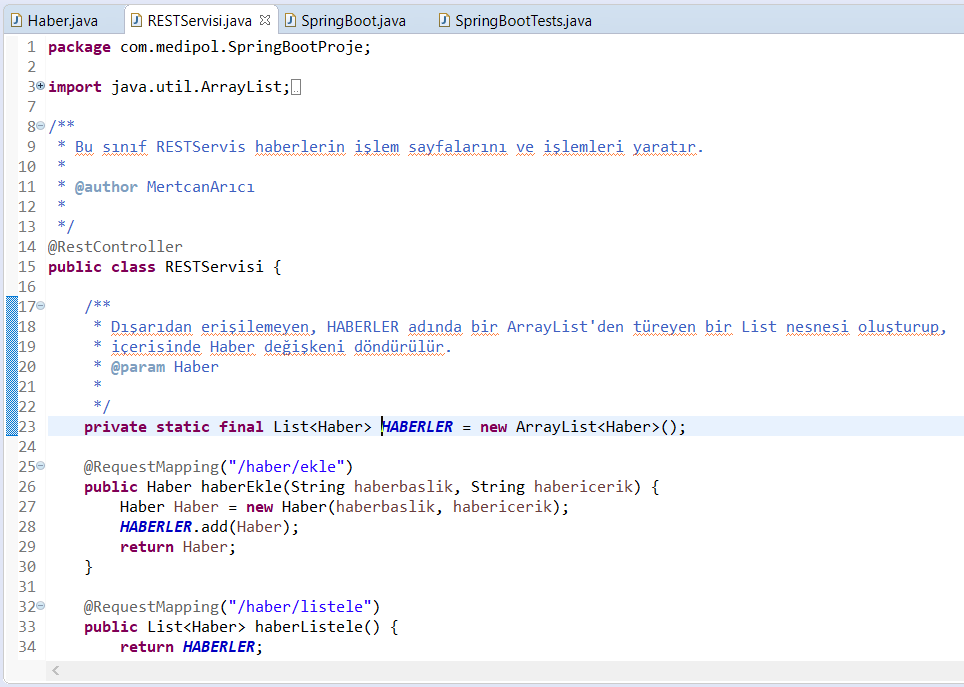


* **RESTServisi.java**

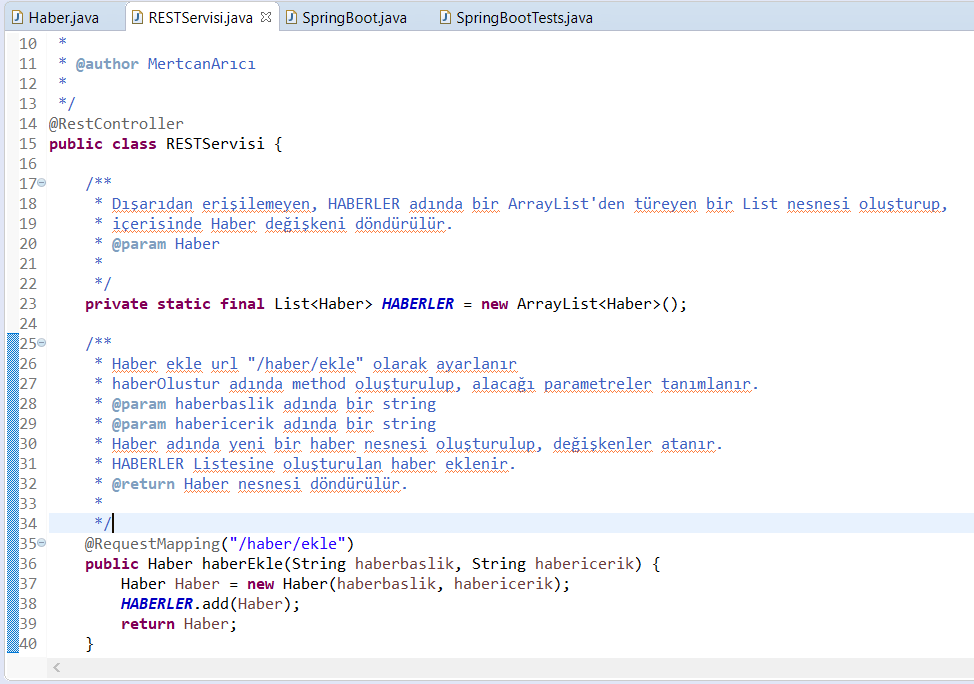
Sayfa için Javadoc hazırlarken ilk olarak Javadoc yorumlarını hazırlıyoruz. İlk olarak sınıfın ne işlem yaptığını ve hazırlayan kişiyi Javadoc etiketi @author ile tanımlıyoruz.



Sonra Javadoc @param etiketi ile bu sınıfın içindeki değişkenleri tanımlıyoruz.



Sonra bu sınıfın içindeki haberEkle metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını, Javadoc @param etiketi ile hangi değişkenleri barındırdığını ve @return etiketi ile metodun döndürdüğü değeri açıklıyoruz.



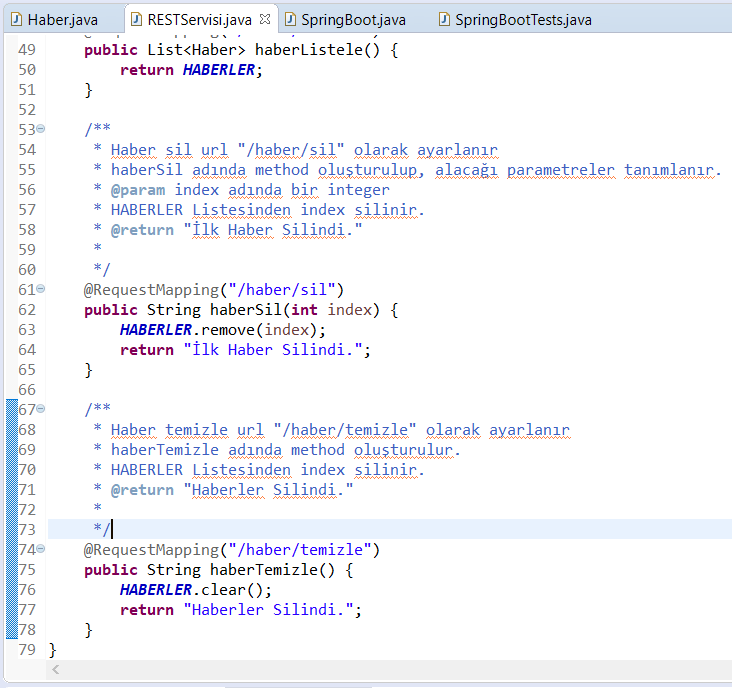
Sonra bu sınıfın içindeki haberListele metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını ve Javadoc @return etiketi ile metodun döndürdüğü değeri açıklıyoruz.



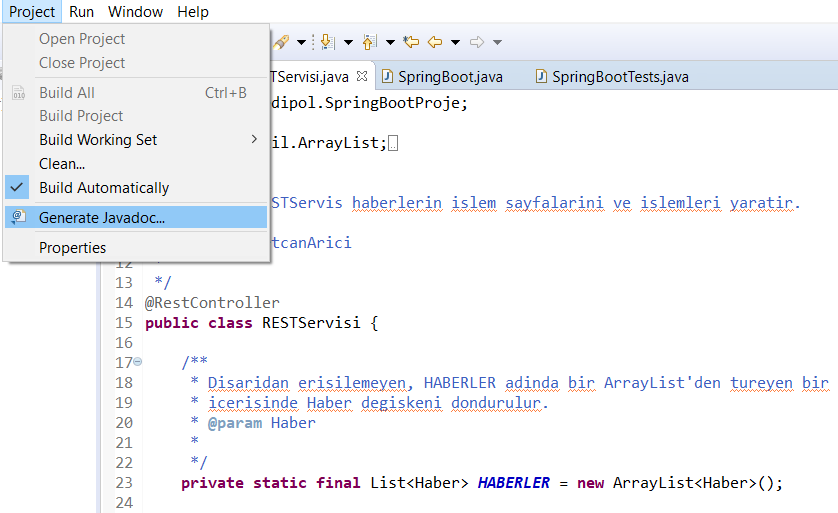
Sonra bu sınıfın içindeki haberSil metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını, Javadoc @param etiketi ile hangi değişkenleri barındırdığını ve @return etiketi ile metodun döndürdüğü değeri açıklıyoruz.



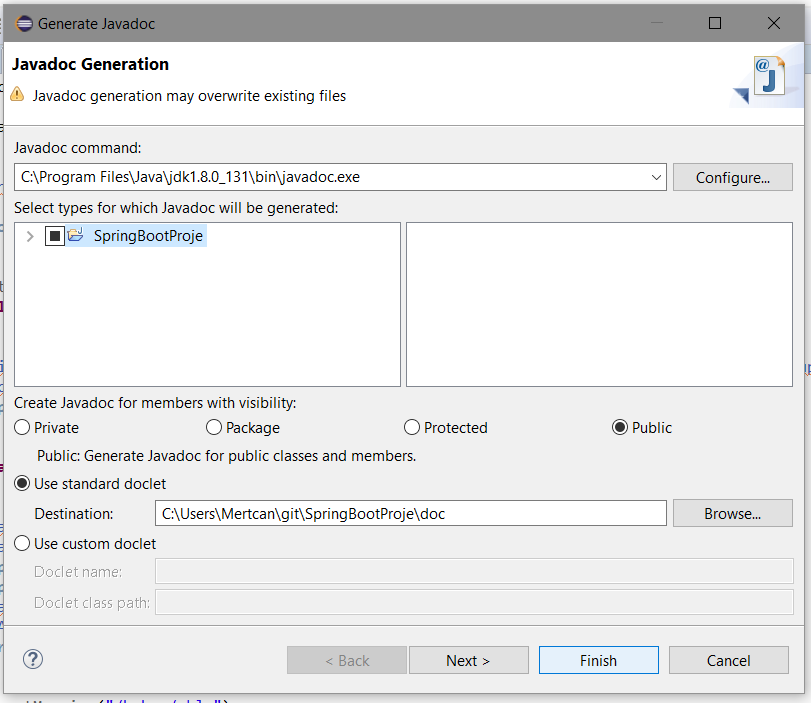
Sonra bu sınıfın içindeki haberTemizle metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını ve Javadoc @return etiketi ile metodun döndürdüğü değeri açıklıyoruz.



Javadoc yorumlarını hazırladıktan sonra html dökümü olarak almak için Project –> Generate JavaDoc kısmı seçilir.

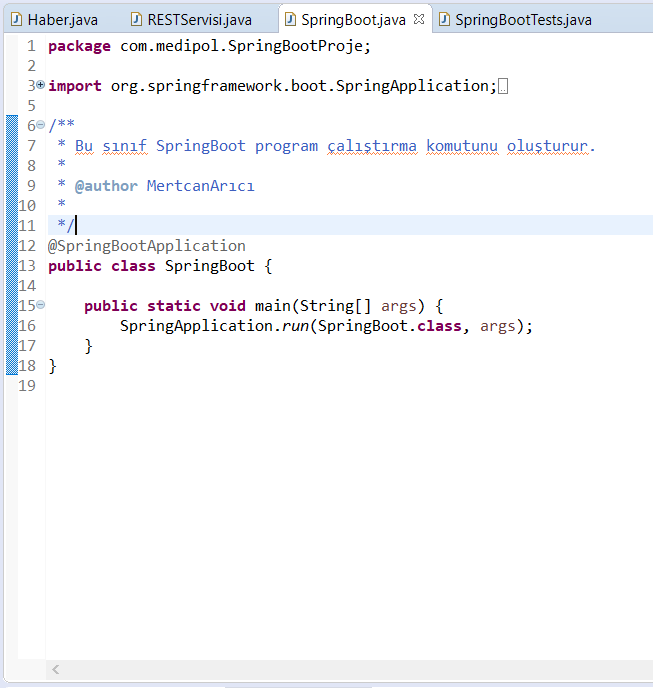


Son olarak da finish butonuna tıklarız.

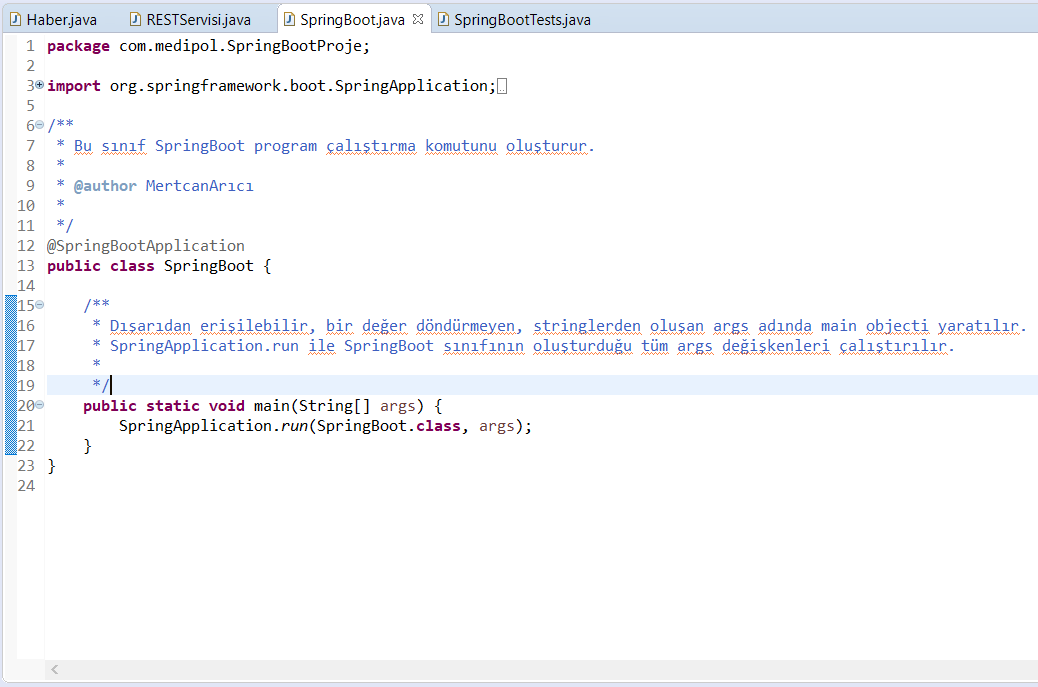


* **SpringBoot.java**

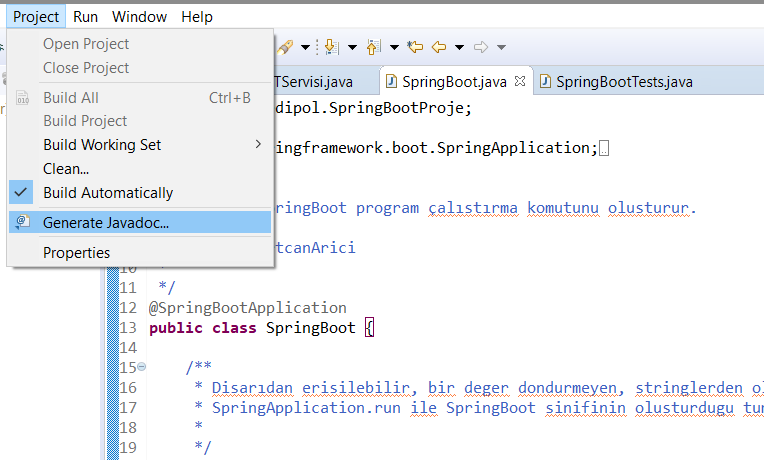
Sayfa için Javadoc hazırlarken ilk olarak Javadoc yorumlarını hazırlıyoruz. İlk olarak sınıfın ne işlem yaptığını ve hazırlayan kişiyi Javadoc etiketi @author ile tanımlıyoruz.



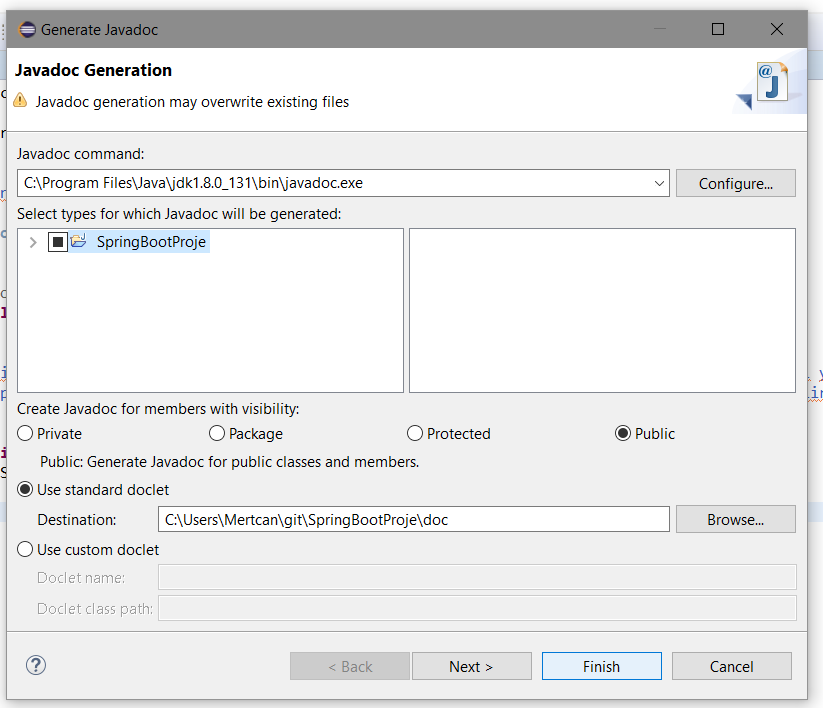
Sonra bu sınıfın içindeki main metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını açıklıyoruz.



Javadoc yorumlarını hazırladıktan sonra html dökümü olarak almak için Project –> Generate JavaDoc kısmı seçilir.

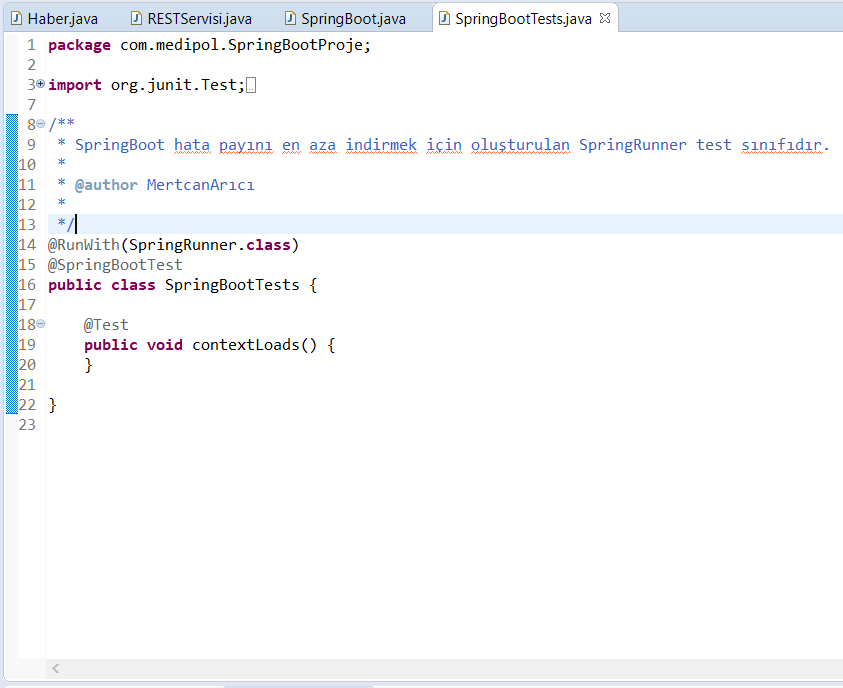


Son olarak da finish butonuna tıklarız.

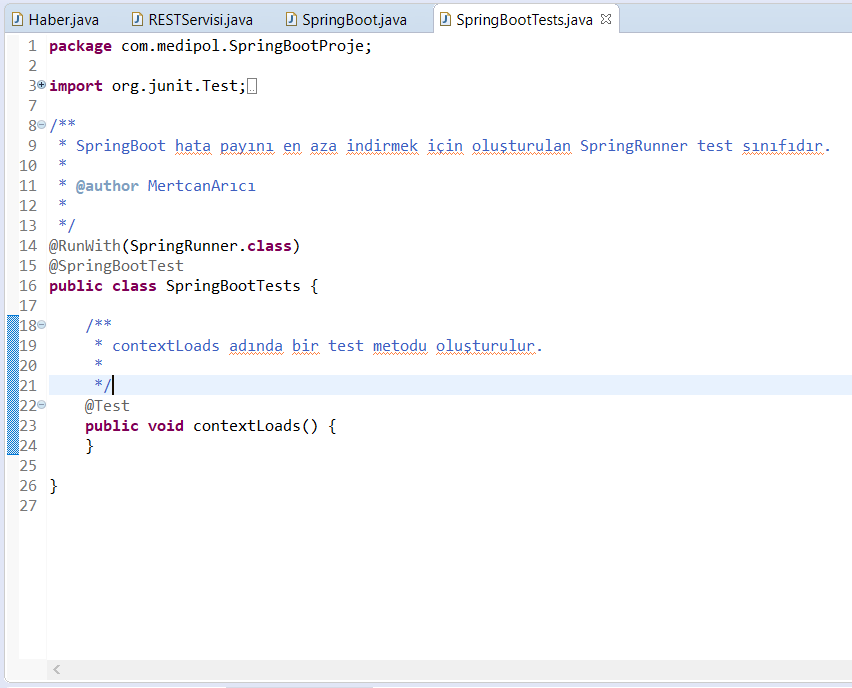


* **SpringBootTests.java**

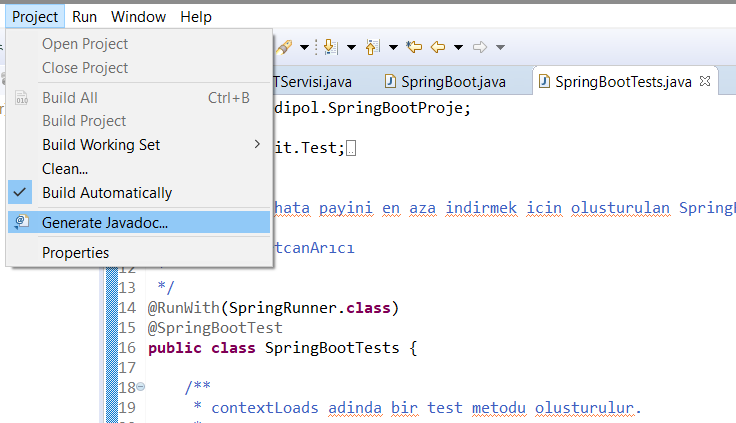
Sayfa için Javadoc hazırlarken ilk olarak Javadoc yorumlarını hazırlıyoruz. İlk olarak sınıfın ne işlem yaptığını ve hazırlayan kişiyi Javadoc etiketi @author ile tanımlıyoruz.



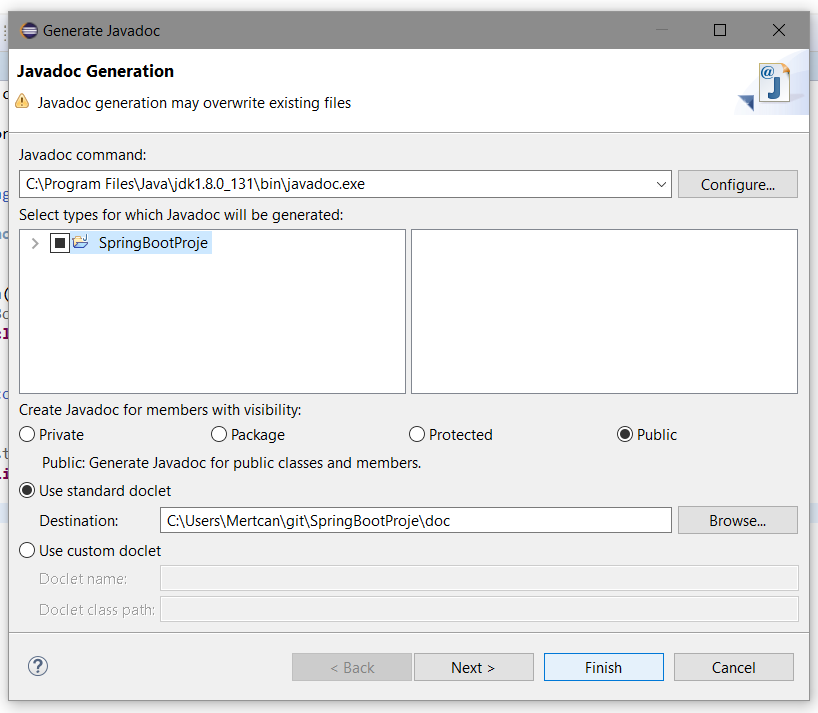
Sonra bu sınıfın içindeki contextLoads metodunu tanımlıyoruz. Tanımlarken metodun ne işlem yaptığını açıklıyoruz.

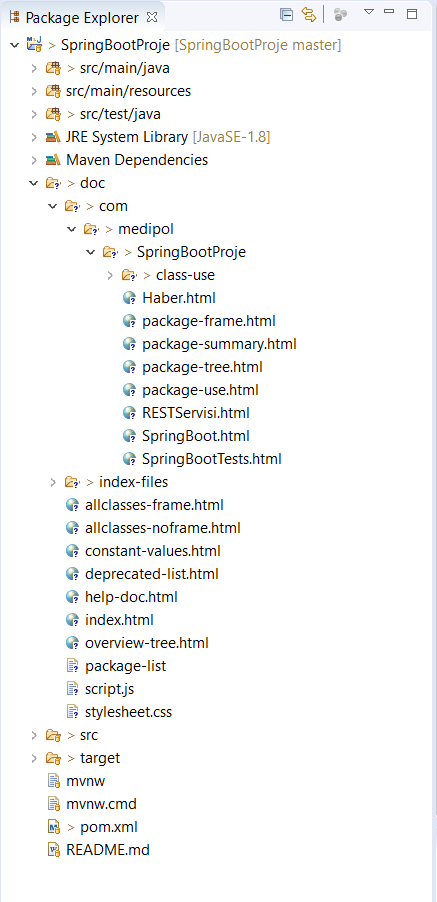


Javadoc yorumlarını hazırladıktan sonra html dökümü olarak almak için Project –> Generate JavaDoc kısmı seçilir.



Son olarak da finish butonuna tıklarız.

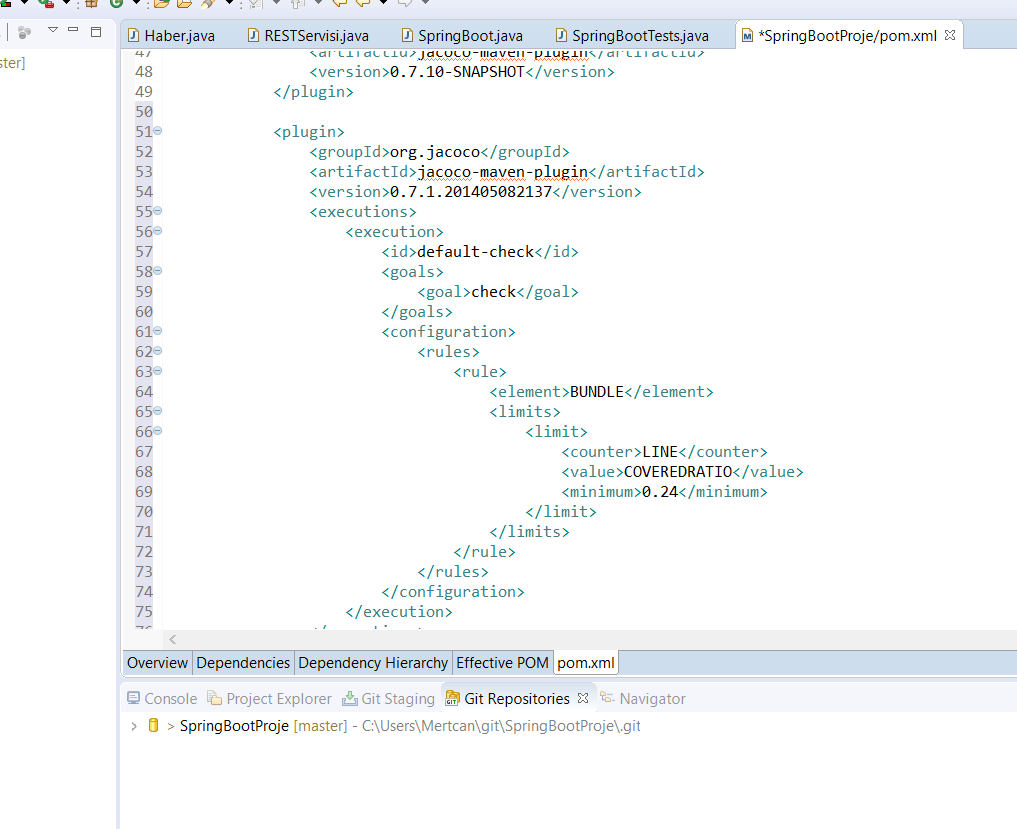




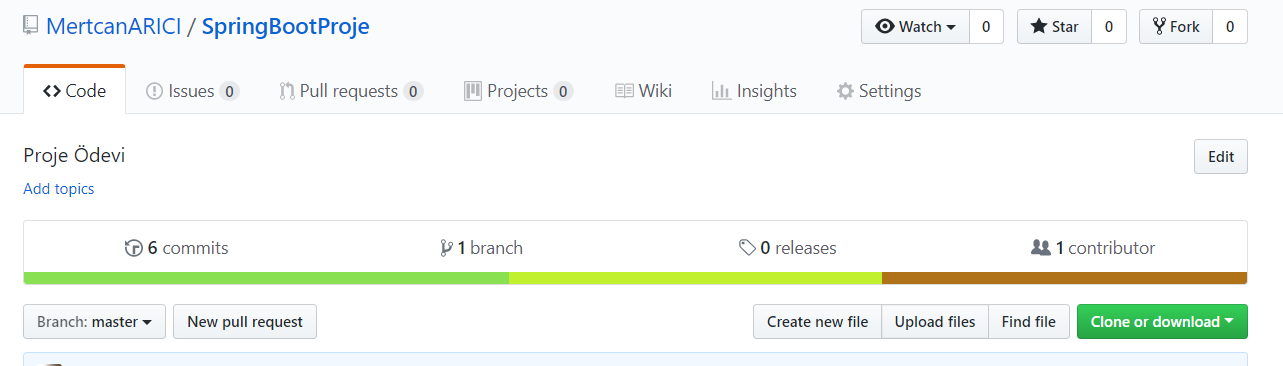
Sayfalarımıza oluşturduğumuz tüm Javadoclar projemizin içine eklenmiştir.

# **travis kurulumu**

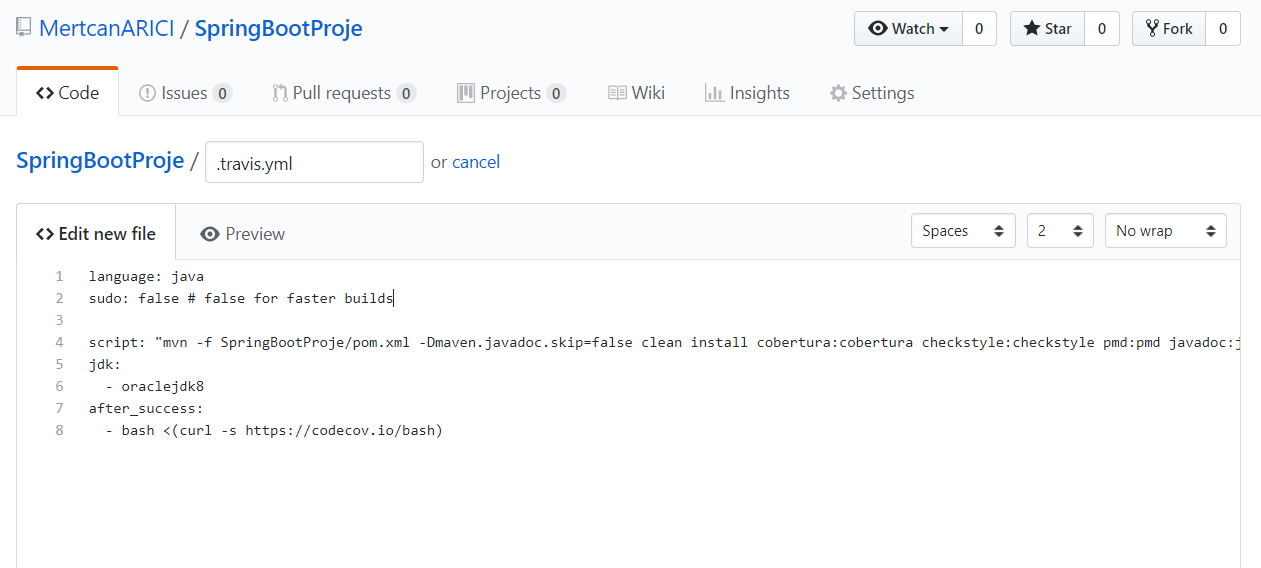
Projemizin ayarlarının yapıldığı pom.xml kısmında travis ile entegre olması için gerekli kütüphaneleri ve gereken kodları yazıyoruz.



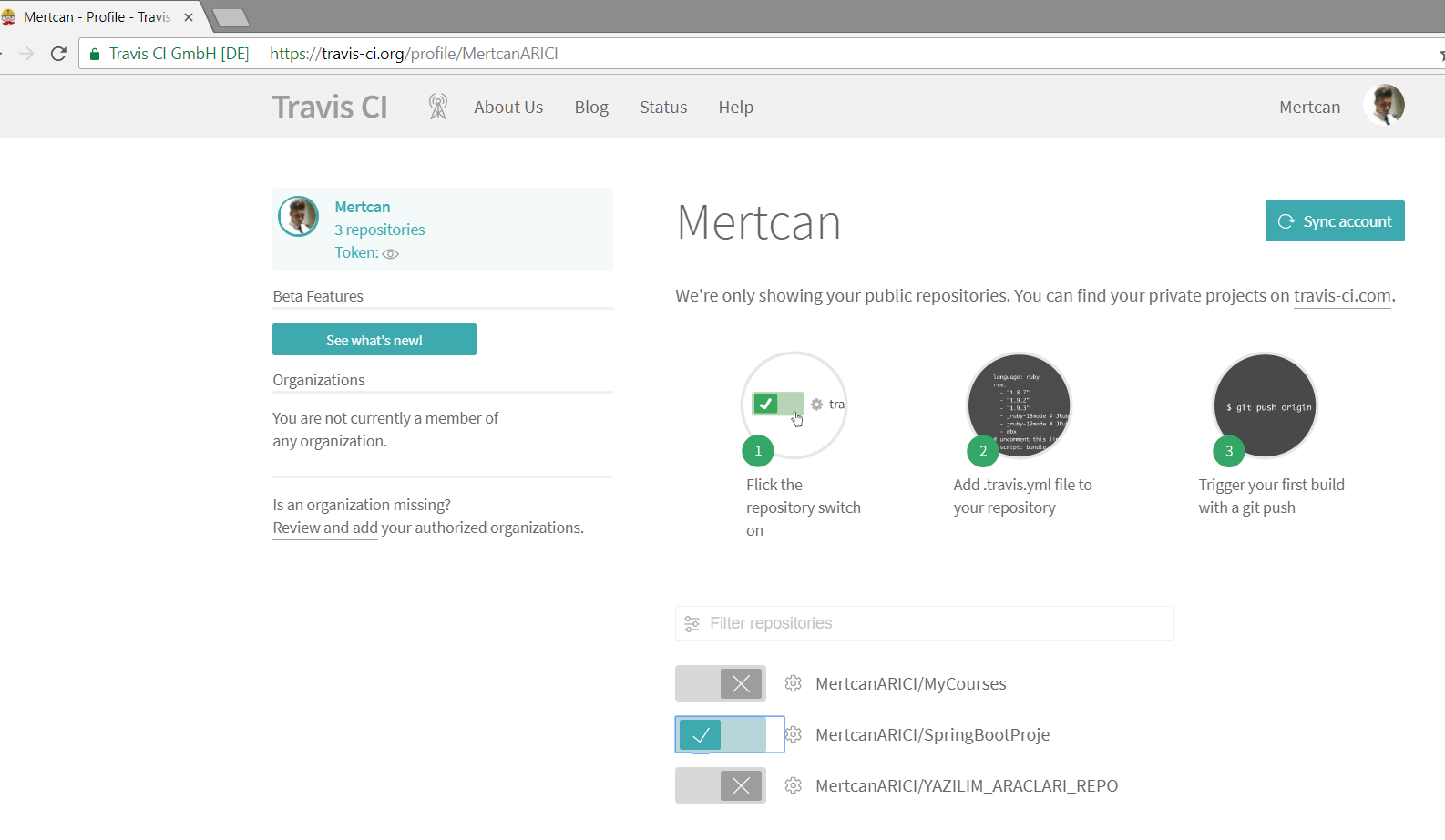
Daha sonra github hesabımıza gidip, repomuza girdiğimizde create new file butotnuna basıyoruz.



Daha sonra travis ile bağlantı kurulması için gerekli olan travis.yml dosyasını oluşturup içine entegre şekilde çalışması için gerekli kodu yazıp kaydediyoruz.

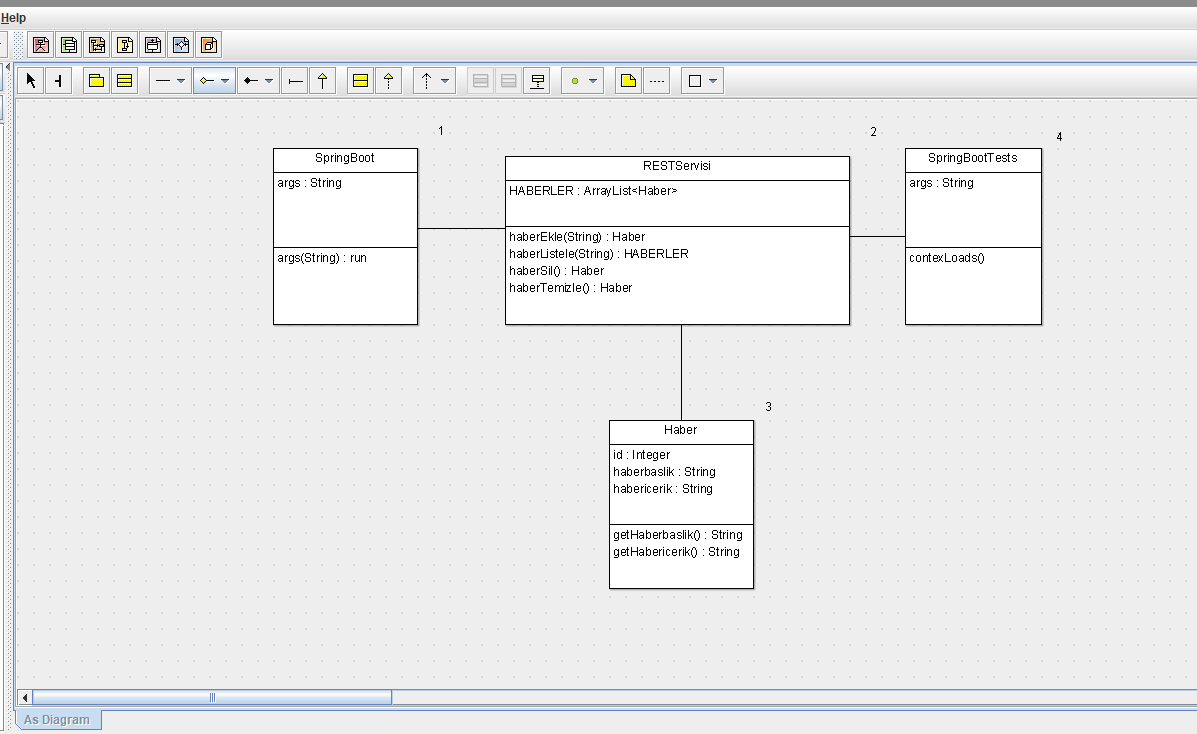


Daha sonra tavis e giriş yapıp github hesabımızdan yüklediğimiz repoların hangilerinin çalıştırılmasını istiyorsak onları açıyoruz. Ve bağlantımız mamamlanmış oluyor.



# **uml diagragramı**

Argo UML diagramı oluşturmak için ArgoUML programını kullandım. Projem dört sınıftan oluştuğu için 4 sınıf oluşturdum. Sınıf isimlerini en baştaki kısıma yazdım daha sonra orta kısıma niteliklerini yazdım ve son olarak işlevleri de en attaki kısıma yazdım. Aralarında işlem olduğunu göstermek için tabloları birbirine bağlayan çizgileri çektim. Sınıflar private olmadığı için – koymadım genel örneklere baktığımda public olduklarında + koyanlarda vardı koymayanlarda bu yüzden sade şeklinde bıraktım public oldukları için. Sınıflarda extends yani yani kalıtım olmadığı için tablolar arasında ok işareti birleştirmesi yapmadım çünkü her bir sınıf ayrı amaçları içeriyor ama hepsi birbiriyle bağlı olduğu için sadece çizgi çektim. Kaynak olarak github a yüklemiş olduğunuz  ini okuyarak anlayarak anladığım kadarıyla UML kısmını bitirmiş oldum. Kullandığım kaynak dosyada UML Kaynak adı altında mevcuttur.

****