Delphi ve Github Entegrasyonu Nasıl Yapılır

Giriş

Konu, RAD Studio (Tokyo) bir Windows ürünü olduğu için doğal olarak Windows temel alınarak anlatılacaktır. Dolayısıyla bu makale Windows dışındaki işletim sistemleri ile ilgili bilgiler içermemektedir fakat bazı adımlar diğer işletim sistemlerinde de benzer şekilde yapılabilir. Özellikle komut satırı ve BASH kullanımı buna örnek verilebilir.
Bu makalede anlatılan konu RAD Studio (C++ açısından) ve AppMethod ile de benzerlik gösterir, dolayısıyla bu teknikleri oralarda da kullanabilirsiniz.
Hedef basit ama süreç karmaşık olduğu için kısa bir makale yazma imkânımız maalesef yok, o nedenle sabırla makaleyi sonuna kadar okumanızı öneririm.

Nedir, Neden?

Git, bilgisayar dosyalarındaki değişiklikleri izlemek ve bunlar üzerindeki çalışmayı birden çok kişi arasında koordine etmek için kullanılan bir sürüm kontrol sistemidir. Esasen Linus TORVALDS tarafından 2005 yılında Linux çekirdeğinin geliştirilmesi sırasında yaşanan koordinasyonsuzlukların bir sonucu olarak üretilmiş ve diğer çekirdek geliştiricileri ile eşgüdümü sağlamak üzere tasarlanmıştır.
Çoğu dağıtılmış (ve genelde ücretli) sürüm kontrol sisteminde olduğu gibi ve çoğu istemci – sunucu sisteminin "tersine" proje klasörlerimizdeki her ".git" dizini, ağ erişiminden veya merkezi bir sunucudan "bağımsız" olarak eksiksiz bir geçmiş bilgisine ve tam sürüm izleme yeteneklerine sahip tam teşekküllü bir havuzdur. Yani en az iki kişinin aynı projede çalışıyor olduğunu varsaydığımızda söz konusu projenin en az 3 adet "tam teşekküllü" (2'si proje katılımcılarında ve 1'i de github sisteminde olmak üzere) birer kopyası vardır demektir. Bu noktada github.com'daki projenin konumu ise proje katılımcıları arasındaki ortak havuz olma rolünü üstlenir. Bu yapı, projeye dahil olan herkesin kendi depolarında her istediğini yapabileceği, aynı zamanda yaptıkları her değişikliğin asıl projeyi etkilemeden, kontrollü bir şekilde birbiriyle paylaşabileceği anlamına gelmektedir. Bunu, yani eşgüdümü de projenin katılımcıları "commit" ve "push" mekanizmalarıyla gerçekleştirirler.

Git ve RAD Studio arasındaki Mekanizma

Yazının devamında da anlatılacağı üzere RAD Studio ve GIT sürüm kontrol sistemi aslında arka tarafta BASH denen bir kabuk uygulamayı kullanır. Bash aslında Unix sistemlerinde kullanılan ve Windows'daki DOS komut sistemine denk gelen bir yapıya "benzer". Aslında yaptığı iş komutları yorumlamak ve çalıştırmaktır. BASH ve diğer Unix kabukları için aşağıdaki linkten daha fazla bilgi edinebilirsiniz. Bizim açımızdan RAD Studio ve git sistemi arasındaki bağlantıyı BASH denen bu kabuğun arka planda yaptığını bilmemiz yeterli olacaktır.

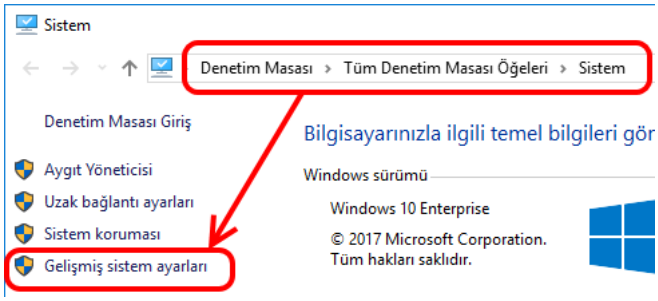
http://web.deu.edu.tr/doc/lis/lis-6.html

Git Sisteminin Kurulması

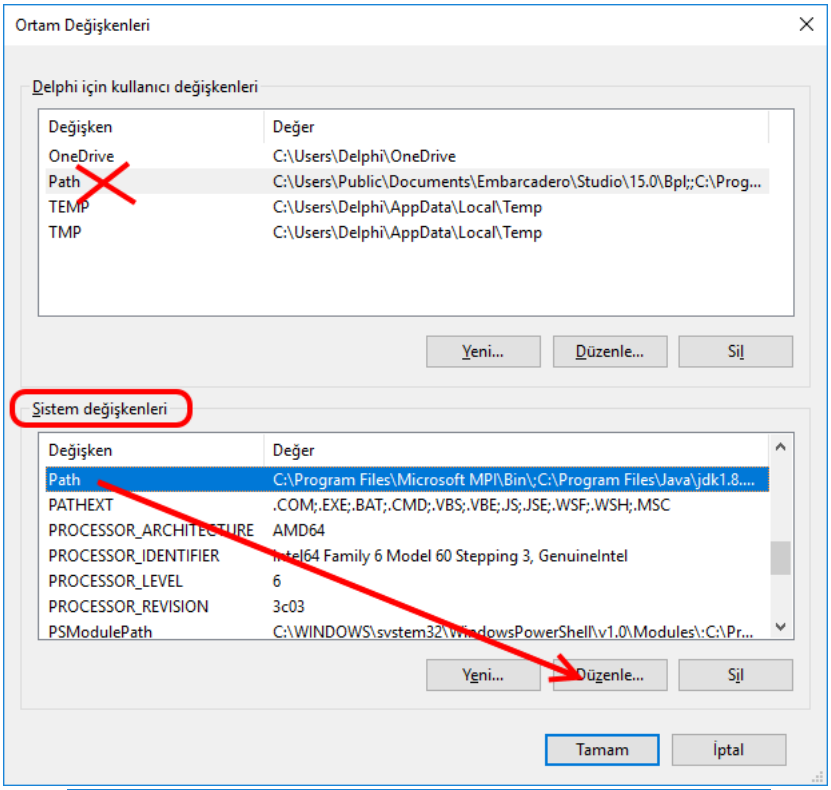
Öncelikle şu (https://git-scm.com/download/win) bağlantıdan GIT uygulaması indirip kurulum programının yönergelerini hiç sorgulamadan (konuyu dağıtmamak adına) NEXT, NEXT, NEXT şeklinde hızlıca kuruyoruz. Kurulum BASH kabuğu ile birlikte GIT için gerekli olan tüm altyapısal yazılımları sisteminize kuracaktır.
CMD dizininde git-gui.exe aracı da yer alır. Biz anlatımımızda herhangi bir GUI arabirimini işin içine katmayacağız ve komutlarımızı konsol üzerinden gerçekleştireceğiz.
Kurulumdan sonra GIT uygulamasının kurulu olduğu klasörü tespit ediyoruz ve bir kenara not ediyoruz. Varsayılan olarak (kurulum sırasında aksini belirtmediyseniz) sizin sisteminizde de aşağıdaki gibi bir yere kurulmuştur.

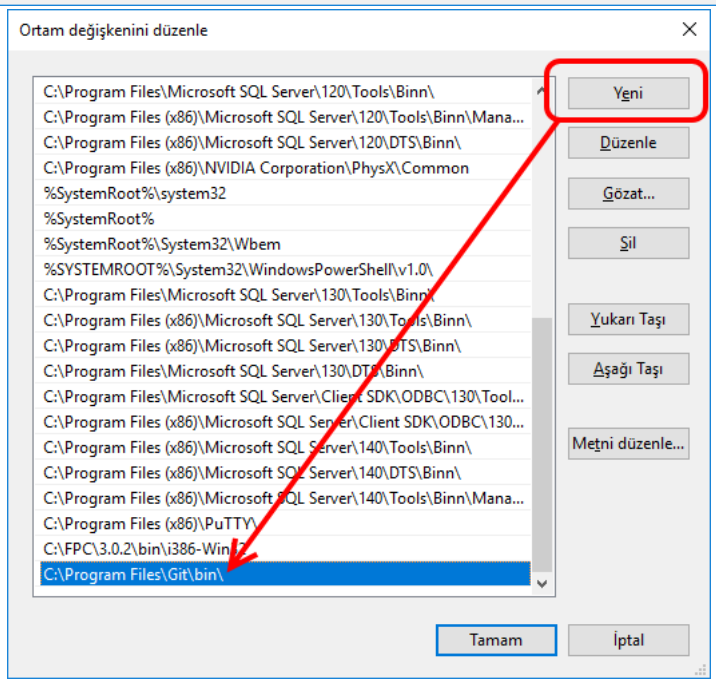
- C:\Program Files\Git\bin\git.exe

Bu noktada DOS komut satırındaki kullanımlarımız için bu klasörün adresini Windows'un PATH listesine eklememiz gerekiyor, bunun için Denetim masası \ Sistem \ Gelişmiş Sistem Ayarları bağlantısını açıyoruz.



Ardından karşımıza çıkan "Ortam Değişkenleri" başlıklı pencerenin alt bölmesindeki listeden PATH değişkenini bulup aşağıdaki düzenle butonuna basıyoruz. Akabinde açılan "Ortam Değişkenini Düzenle" penceresinde yeni butonuna basıp listenin en sonuna aşağıdaki ekran görüntüsündeki gibi git.exe'nin bulunduğu klasörü ekliyoruz ve TAMAM ile pencereleri kapatıyoruz.



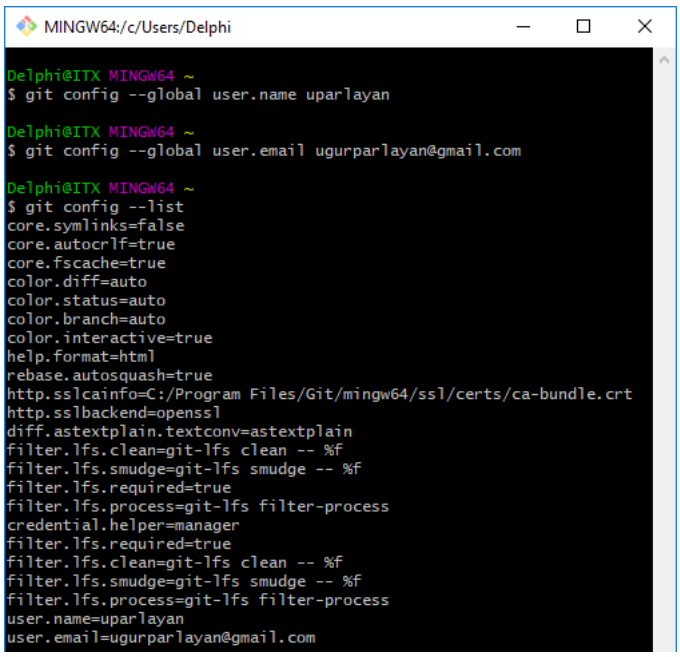


Kaldığımız yerden devam edecek olursak, GIT Uygulaması kurulduğunda masaüstüne “Git Bash” adlı bir simge ekleyecektir. Bu simgeyi açarak Bash konsolunu çalıştırıyoruz. Bu noktada sırasıyla aşağıdaki iki satır komutu çalıştırıyoruz; (çift tırnaklar hariç)

- git config --global user.name “GİTTEKİKULLANICIADINIZ”
- git config --global user.email “GİTTEKİ@EPOSTA.ADRESİNİZ”

Ardından ayarlarımızın doğru bir şekilde kaydedilip kaydedilmediğini teyit etmek, doğrulamak için aşağıdaki komutu çalıştırıp devamındaki listeyi inceliyoruz. Yukarıda eklediğimiz iki satır parametre, söz konusu listenin muhtemelen en alt iki satırında yer alacaktır.

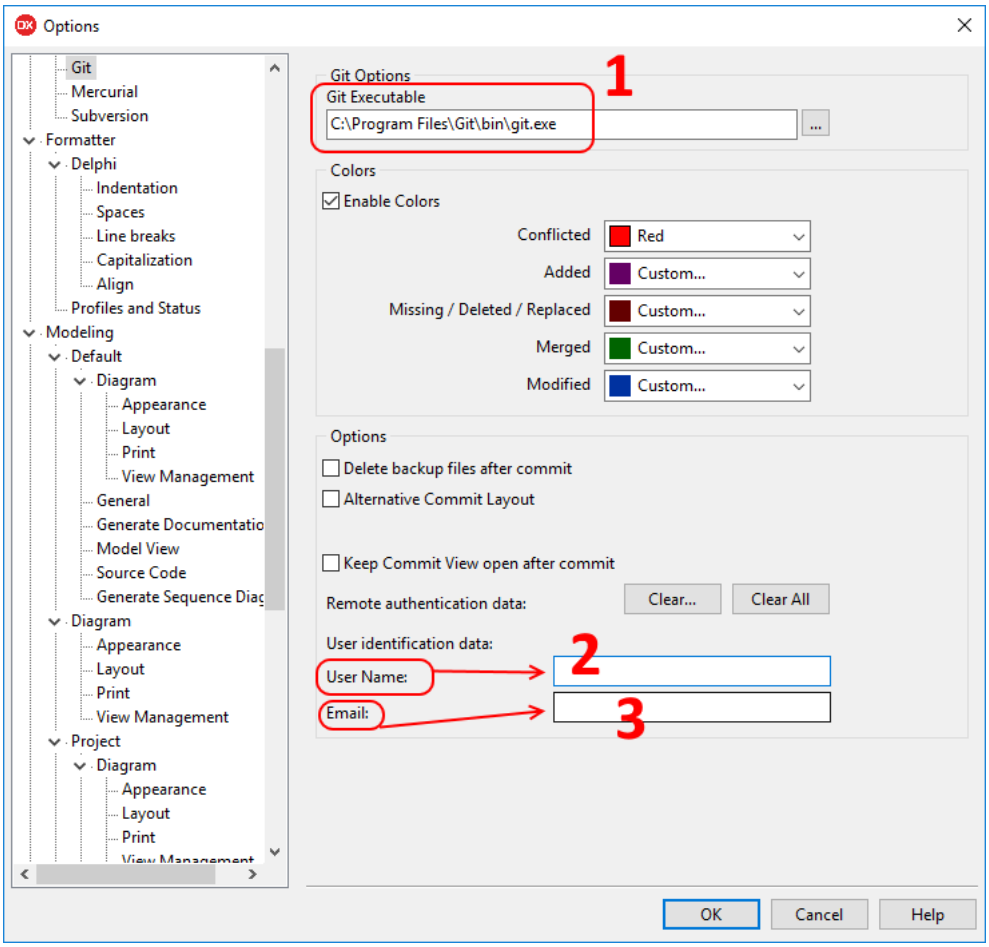
- git config --list



Yukarıdaki ekran görüntüsünde de görüleceği üzere listenin en altında yer alan iki adet parametreye dikkat edelim. Doğru işlenmemişse git config parametrelerini düzgün şekilde yeniden çalıştıralım ve yeniden kontrol edelim. Yukarıdaki görüntüde benim kullanıcı adım ve benim eposta adresim örnek olarak verilmiştir.

RAD Studio'daki Git Ayarları

Eğer yazdığımız parametreler söz konusu listede düzgün bir şekilde kaydedilmişse RAD Studio'yu çalıştırıyoruz. Açıldığında Tools \ Options menüsünden veya IDE Insight arama çubuğuna "Git page" yazarak "Options" başlıklı ekranı açıyoruz;

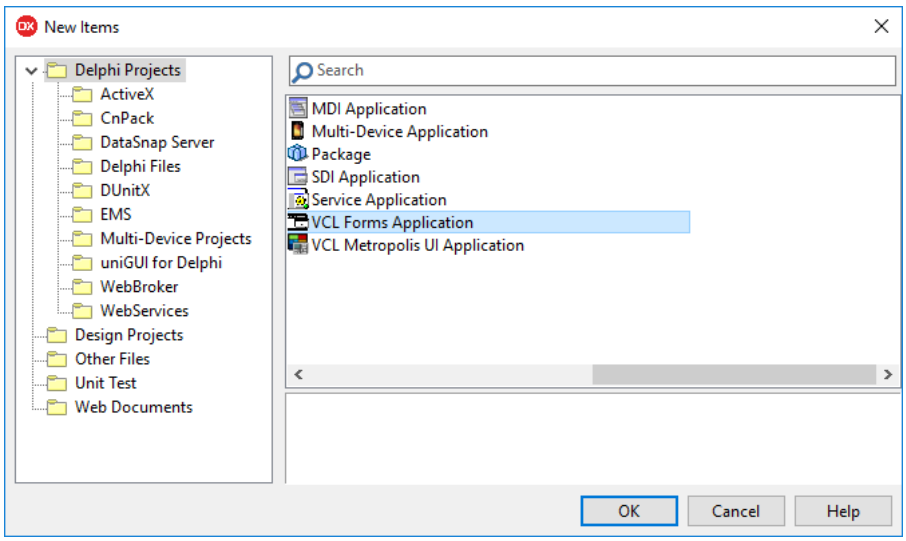


Bu ekran görüntüsünde "1" ile vurgulanan kısma "git.exe"nin tam yolunu yazıyoruz. Ekranın en altında yer alan "User Identification Data" kısmına da Bash'ta da yazdığımız gibi github kullanıcı adını ve githup'da kullandığımız eposta adresini yazıyoruz.

RAD Studio'ya Git entegrasyonunu bu şekilde yapmış olduk. Bu noktadan sonrasında ise bu senkronizasyon, eşleştirme, kıyaslama, yayınlama ve yayından çekme gibi konular yazının devamında anlatılmaktadır. Buraya kadar anlatılan konuları sadece bir kere yapmak yeterlidir. Bu noktadan sonra anlatılacak olan ayarlar her git projesi için ayrı ayrı yapılmalıdır.

Basit Bir Proje

Örnek olması açısından ve tüm adımları tamamen gösterebilmek adına RAD Studio'da boş bir proje oluşturmakla işe başlayalım. Hepimizin bildiği gibi standart bir VCL Application oluşturalım; (Bu noktada projenin türünün ve olduğunun hiç bir önemi yok)



Ardından boş unitimizi bu haliyle hemen kaydedelim;

Projemizi de peşinden kaydedelim;

Proje İçin Yerel bir Git Deposu Oluşturulması

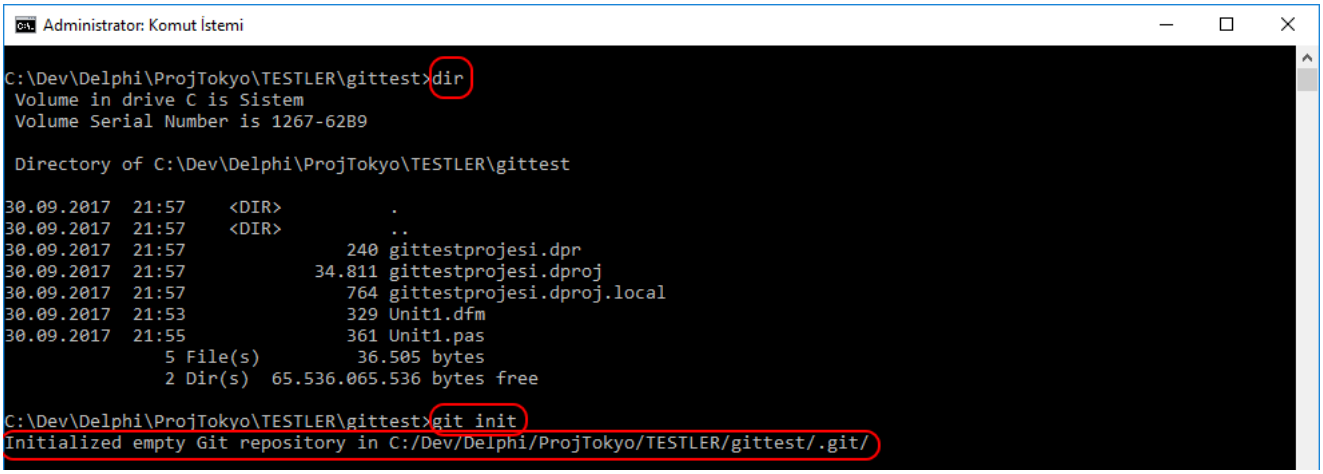
Projemizi kaydettikten sonra DOS komut satırını yönetici kipinde çalıştırıyoruz ve ardından proje dosyalarımızın olduğu klasöre giriyoruz. Projemizin klasöründeyken aşağıdaki komutu çalıştırıyoruz;

- git init

Bu komut, projemizin olduğu klasöre ".git" adında gizli bir klasör oluşturacak ve bunu bir repository'e (depoya) dönüştürecektir. Bu klasörü komut satırında DIR çektiğinizde göremeyebilirsiniz fakat komut başarılı bir şekilde çalışmışsa konsola aşağıdaki gibi bir mesaj yansıyacaktır.

- "Initialized empty Git repository in C:/Dev/Delphi/ProjTokyo/TESTLER/gittest/.git/"

Tıpkı aşağıdaki ekran görüntüsünde de gösterildiği gibi...



Bu noktaya kadar başarılı bir şekilde ulaştıysak projemiz için yerel olarak bir Git havuzu oluşturabildik demektir. Yani projemizin olduğu klasörü incelersek ".git" adlı "gizli" bir klasör oluşturulduğunu göreceksinizdir.

Artık elimizde yerel bir havuz olduğuna göre bunu RAD Studio üzerinden nasıl yönetebileceğimizin yollarını kullanabilir duruma geldik sayılır.

Git Menüsünün Detayları

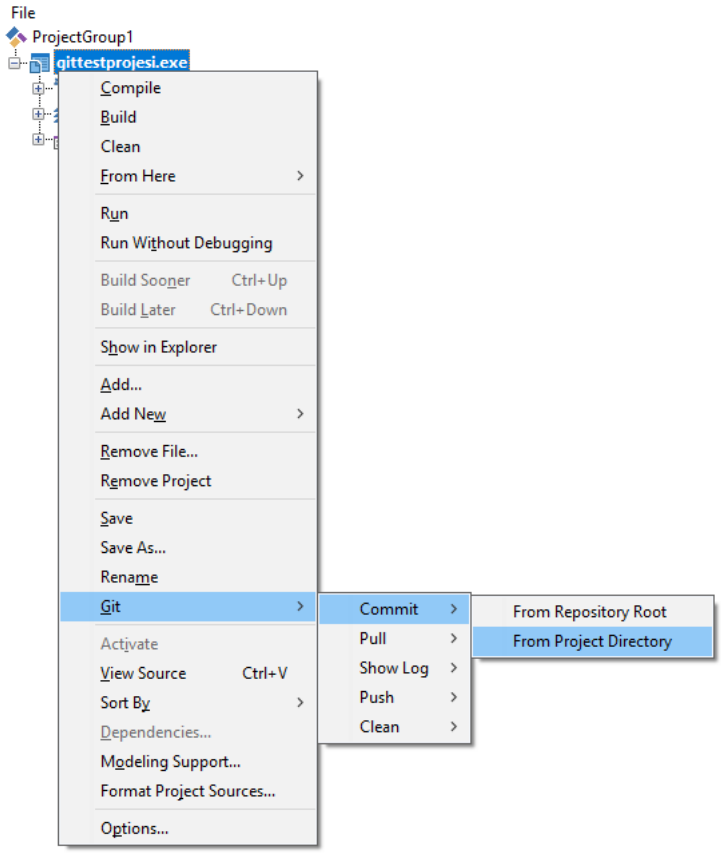
Yeri gelmişken Git menüsündeki öğelerin ne işe yaradığından bu noktada kısaca bahsetmek yerinde olacaktır.

- **Commit** : Değişikliklerin yerel depoya veya github'daki depoya işlenmesini sağlar.
- **Push** : Proje dosyalarının github deposuna aktarılmasını veya github deposundan yerel depoya aktarılmasını sağlar
- **Show Log** : Yaptığınız değişiklikler ile ilgili log bilgilerini yerel veya github deposu bazında incelemenizi sağlar.
- **Clean** : Yereldeki veya github'daki deponuzu "boşaltmanızı" sağlar. Bu seçeneği kullanırken dikkat etmenizi tavsiye ederiz.
- **Pull** : Push'un tersini yapar, yani güncelleme işlemini yerel depo kaynağından veya github depo kaynağından güncellemenizi sağlar. Bu noktada Pull yerine Push menüsünün kullanılması kafa karışıklığını önleme adına yararlı olabilir.

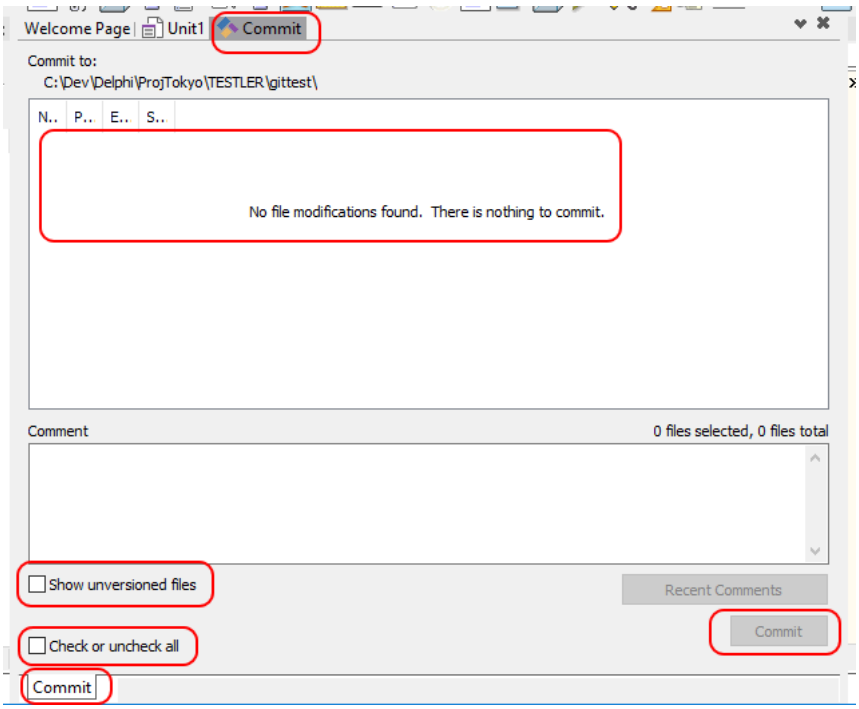
Proje Senkronizasyonu

Tekrar RAD Studio'ya dönüp projemize sağ tıklayalım, tıpkı yandaki gibi bir seçenekler manzumesiyle karşılaşacağız. Sırasıyla örneklemeler üzerinden adım adım konuyu inceleyelim.

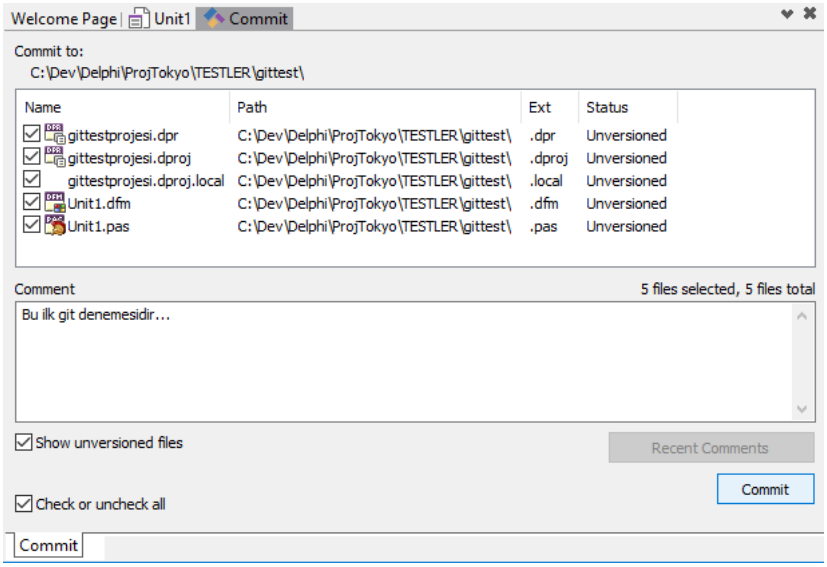
Project Manager bölmesinde projemize sağ tıkladığımızda açılan popup menünün ortalarından sonrasındaki bir bölümde "Git" menüsü belirir. Bu menünün (RAD Sürümüne bağlı olarak) birkaç alt menüsü daha vardır. Biz örneğimizde Tokyo versiyonu üzerinden devam edeceğimiz için aşağıdaki gibi bir menü ile karşılaşacağız;



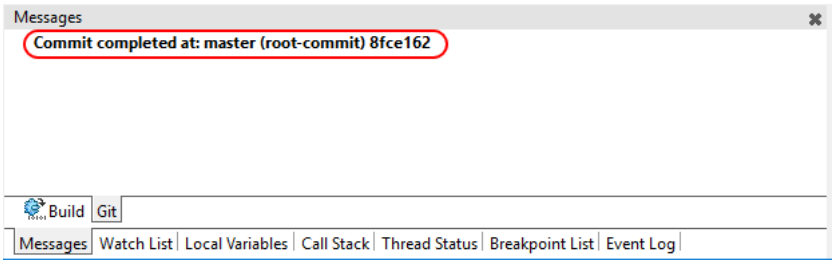
Projemizde henüz bir sürüm takibi yapmadığımız için ilk önce nelerin sürümünü takip edeceğimizi de belirlememiz gerekir. Bunun için yandaki ekran görüntüsünde de olduğu gibi popup menüden Git \ Commit \ "From Project Directory" öğesine basıyoruz. Karşımıza içi boş bir dosya listesi çıkacaktır. Tıpkı aşağıdaki ekran görüntüsünde olduğu gibi.



Bu noktada Commit ekranının sol alt köşesinde iki adet işaret kutucuğu yer alacaktır. Her ikisini de işaretlediğimizde yukarıdaki listede projemize dahil olan dosyaların listesi yer alır. Ayrıca sağ alt köşedeki "Commit" butonu da (aslında bir comment (açıklama) yazdığınızda) etkinleşecektir. Son durum şunun gibi bir şey olacaktır;



Görüldüğü üzere "Commit" butonuna basıldığında işlemin başarıyla gerçekleştirildiğine dair bir mesaj alacaksınız. Bu mesaj aşağıdaki ekran görüntüsünde de gösterildiği üzere Mesaj penceresinde gözükecektir.



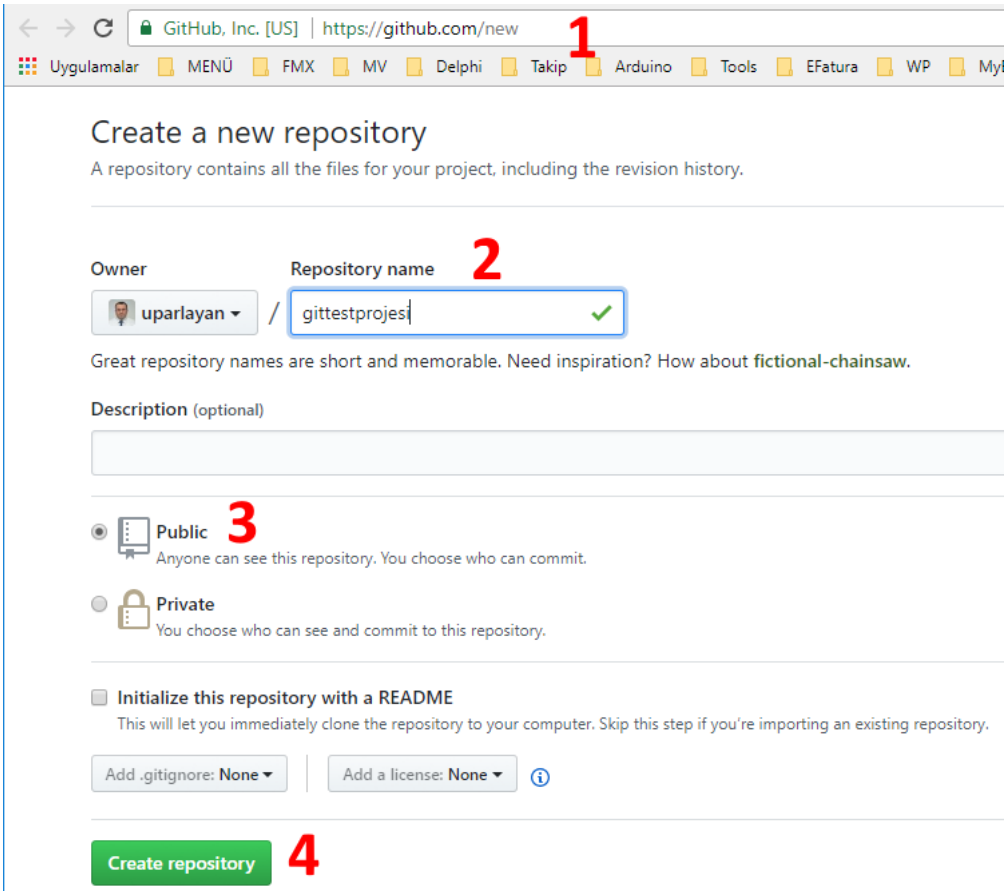
Değişiklikleri yerel depomuza işlediğimize göre artık bu depoyu github üzerinden de erişilebilir ve takip edilebilir bir hale dönüştürmenin zamanı geldi.

Projenin Github'a Tanıtılması

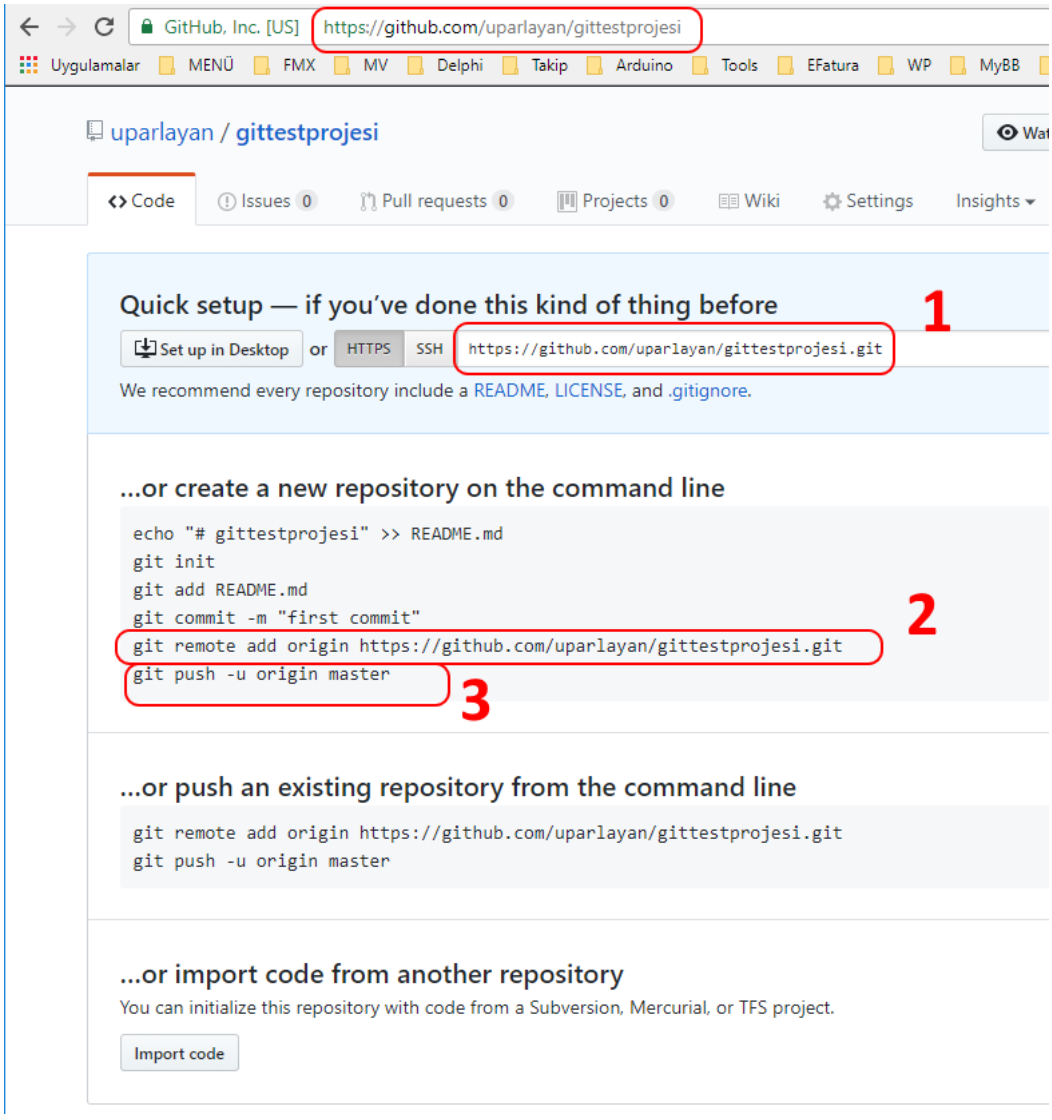
Commit işlemini gerçekleştirdiğimizde versiyon kontrolünü yerel düzeyde ve basit anlamda takip eder hale gelmiştik. Artık sıra bu projenin diğer takım arkadaşlarımızla paylaşımını ve dağıtımını yapmaya geldi. Bu noktada proje popup menüsünde "Git\Push\From Project Directory" menüsünü kullandığımızda URL'nin bulunamadığı ile ilgili bir hata mesajıyla karşılaşacağızdır. Bu hata mesajıyla karşılaşmamak veya bu sorunu aşmak için ise önce aşağıdaki bağlantıyı açıp yeni bir depo tanımı yapmamız gerekir;

- https://github.com/new

Linke gittiğimizde aşağıdaki gibi bir sayfa açılacaktır. Bu sayfada (ben daha önceden login olduğum için) owner kısmında sizin kullanıcı adınızın olduğu bir form ekranı gelecektir. Repository Name kısmına projenize verdiğiniz ismin aynısını yazın. Eğer kamuya açık, herkesin kaynak kodlara erişebileceği bir depo olmasını istiyorsanız "public" tersini istiyorsanız da "Private" seçeneğini işaretleyip yeşil renkli "Create Repository" butonuna basıyoruz. (Private seçeneğinin ücrete tabi olduğunu unutmayın. Biz public seçeneği üzerinden devam edeceğimiz için bu kısımları es geçiyoruz.)



"Create Repository" butonuna bastıktan sonra github, projemiz ile ilgili depoyu kendi sisteminde oluşturmasının ardından, aşağıdaki ekran görüntüsünde de görüldüğü üzere web depomuzu nasıl yöneteceğimize dair bazı ek bilgiler ve ipuçları verecektir.

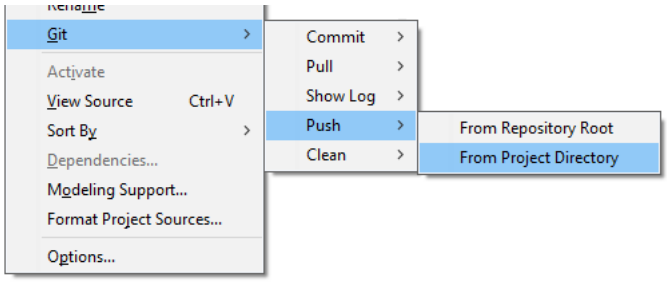


Yukarıdaki resimde "1" ile vurgulanan adres, github havuzumuzdaki projenin git adresini içermektedir. Bu adresi kullanarak github ve RAD Studio arasında projemizin havuzlarını birbirine bağlayabiliriz. "2" ve "3" ile vurgulanan kısımlar ise DOS komut satırını kullanarak demin bahsettiğimiz bağlantıyı kurabilmemizi sağlar.

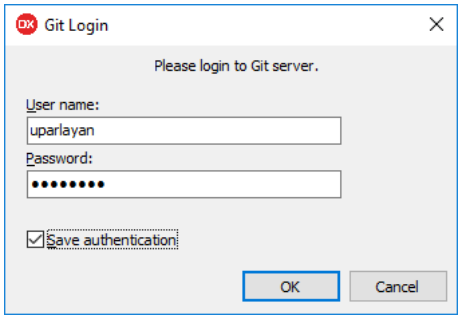
Bu noktada projemiz ile github deposu arasında bağlantı kurmak için iki seçeneğimiz olduğunu görmüş oluyoruz. Sırayla bu yöntemlerin nasıl uygulanacağını inceleyelim.

1. Yöntem / RAD Studio Üzerinden

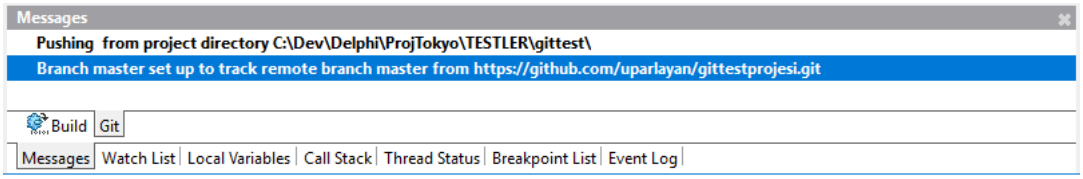
Project Management bölümündeki projemize sağ tıkladığımızda açılan aşağıdaki menüden, yani "Git\Push\From Project Directory" butonuna basıyoruz. Bunu yaptığımızda yereldeki depomuzu github'daki depoya aktarmış olacağız.



Bu butona bastığımızda RAD Studio bize Github'daki kullanıcı adı ve şifremizi soran bir ekran çıkaracaktır.



Bu ekrana github kullanıcı adı ve şifremizi yazdığımızda RAD Studio gereken kopyalama ve eşitleme işlemini gerçekleştirecek ve aşağıdaki bilgilendirme mesajını bize verecektir.



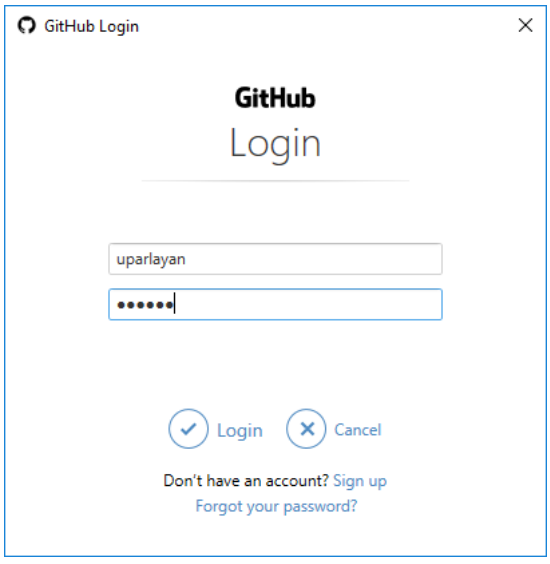
Olay bitti, işte bu kadar! Geçelim diğer yönteme;

2. Yöntem / DOS Komut Satırı Üzerinden

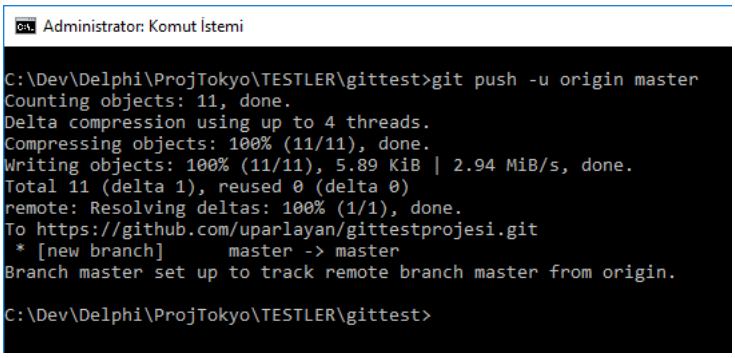
1. Yöntemin ilk başta işe yaramaması durumunda şimdi anlatacağım yöntemi gerçekleştirerek yerel deponuzu github deposuna bağlayabilirsiniz. Bunun için DOS komut satırını yönetici kipiyle açalım. Ardından sırayla şu komutları yazalım; (mavi renkli olan kısım sizin kullanıcı adınızı, kırmızı renkte olan kısım ise projenizin adını ifade etmektedir.)

- git remote add origin https://github.com/**uparlayan**/**gittestprojesi**.git
- git push -u origin master

Bu noktada aşağıdaki gibi bir login ekranı karşınıza gelecektir.



Kullanıcı adı ve şifrenizi yazdıktan sonra DOS komut satırı işlemi gerçekleştirdiğine dair detaylı bir çıktı ile sonucu bize bildirmiş olacaktır.

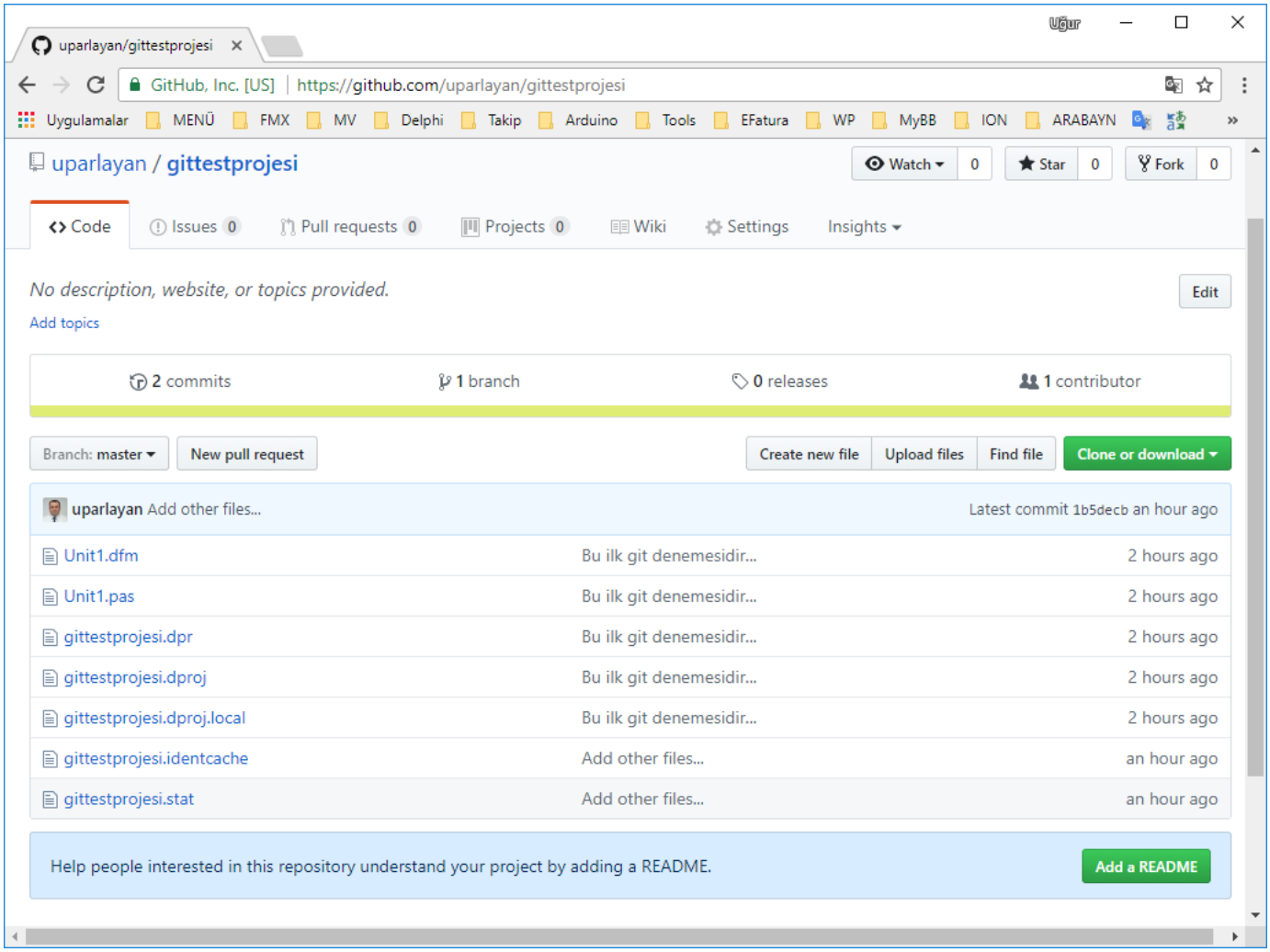


Push yönteminin kurulumu aslında bu kadar.

Projenizin yeni güncellemelerini ise bu noktadan sonra RAD Studio üzerinden (yani 1. Yöntem üzerinden) yapmaya devam edebilirsiniz. Bunun otomatik olmaması size aslında bir hareket alanı da bırakmış oluyor, yani bu noktada siz neyi ne kadar paylaşacağınızı kendiniz belirlemiş oluyorsunuz...

Her şey tamamlandığında projeniz github sitesinde aşağıdakine benzer şekilde gözükecektir; Sizin de fark ettiğiniz gibi, sayfada ilk commit işleminde işaretlediğiniz dosyaların ve sonrasında 2. Commit işleminde eklediğiniz dosyaların tamamı yer almaktadır.

Fakat github sisteminin tek faydası bu değil, bu sayfada kaynak kodlar dışında sorunları takip edebileceğiniz, projenize bağlanma isteğinde bulunanları görebileceğiniz ve Wiki tarzı dokümantasyon yapabileceğiniz alt sayfalar da bulunmaktadır.

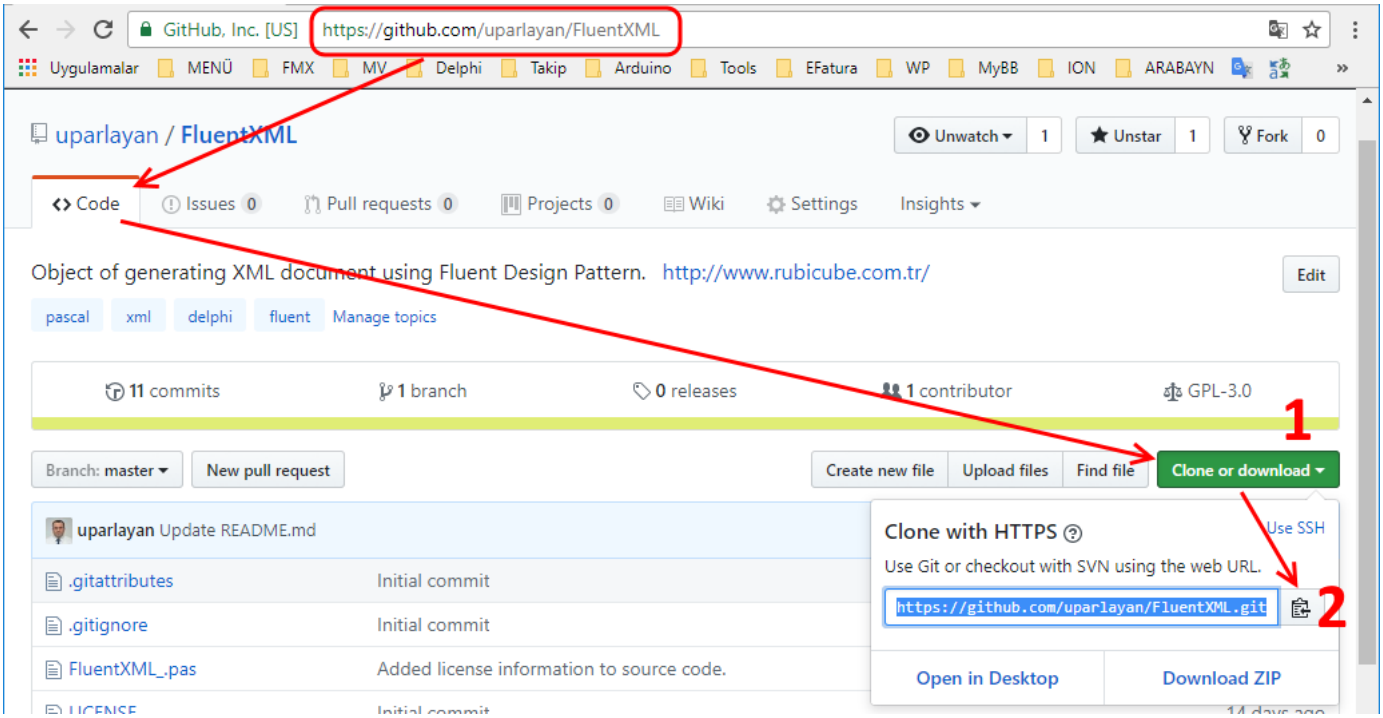


Bir Github projesine RAD Studio üzerinden Bağlanmak

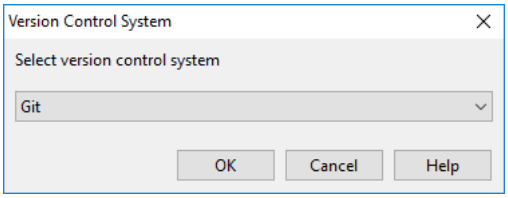
Buraya kadar olan konular bizim kendi projemizi github üzerinde nasıl yayınlarız, yerel ve github deposunu nasıl eşleştiririz, değişiklikleri nasıl yönetiriz gibi konuları içeriyordu. Şimdi ise projemize 2. Bir kişinin katıldığında bu kişinin ne yapması gerektiğine dair ufak bir iki bilgi vereceğiz. Burada anlatılacak olan yöntem, makinanıza format attığınızda projenizi github'daki depodan yerel deponuza alırken de kullanabileceğiniz teknikleri içermektedir. Başlayalım. Öncelikle RAD Studio'daki mevcut projemizi kaydedip kapattıktan sonra github sitesine girelim. Burada Klonunu üretmek istediğimiz projenin sayfasını açalım ve CODE sayfasında listenin sağ üst köşesinde yer alan "Clone or Download" isimli yeşil butona basalım;

Örnek olması açısından FluentXML projesinin yerel bir kopyasını github üzerinden oluşturmak üzere kullanacağım.

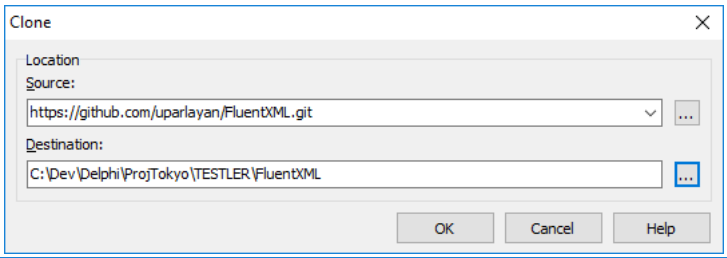
Aşağıdaki resimde "2" ile numaralanmış butona bastığımızda ilgili projenin github adresini elde etmiş oluyoruz.

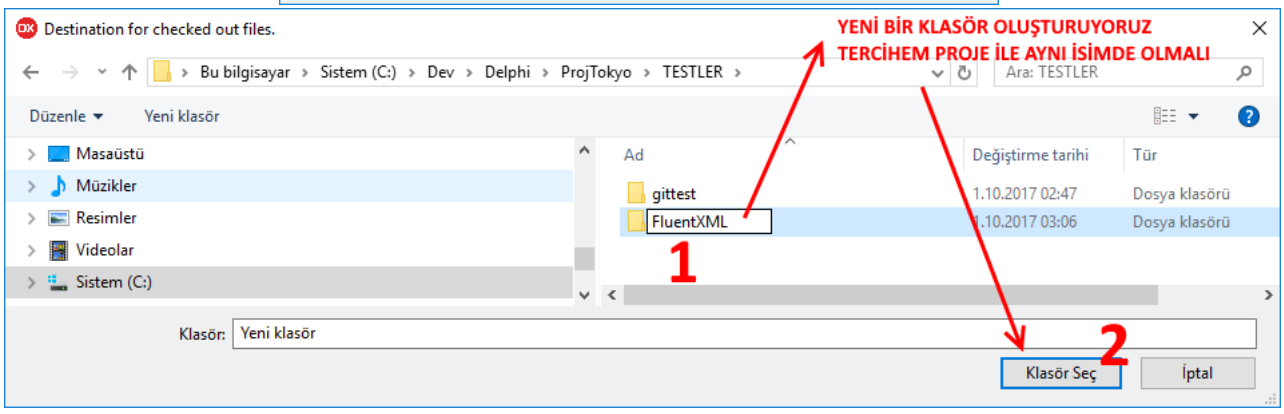


Ardından RAD Studio'da "**File\Open From Version Control...**" butonuna basıyoruz. Karşımıza hangi sürüm kontrol sistemini kullanacağımıza dair bir pencerecik çıkıyor.

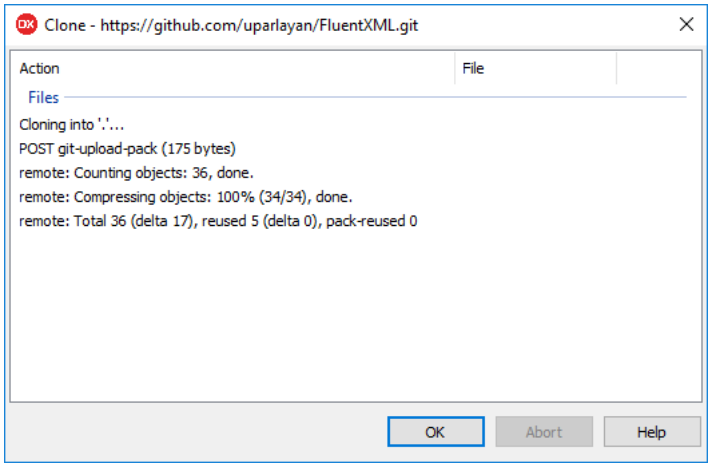


"git"i seçip tamama basıyoruz. Devamında git ile ilgili başka bir pencere daha geliyor. "Clone" başlıklı bu pencerenin SOURCE satırına yukarıdan kopyaladığımız linki yapıştırıyoruz. DESTINATION kısmına ise yerel repository'imizi nereye oluşturmak istiyorsak orayı seçiyoruz.



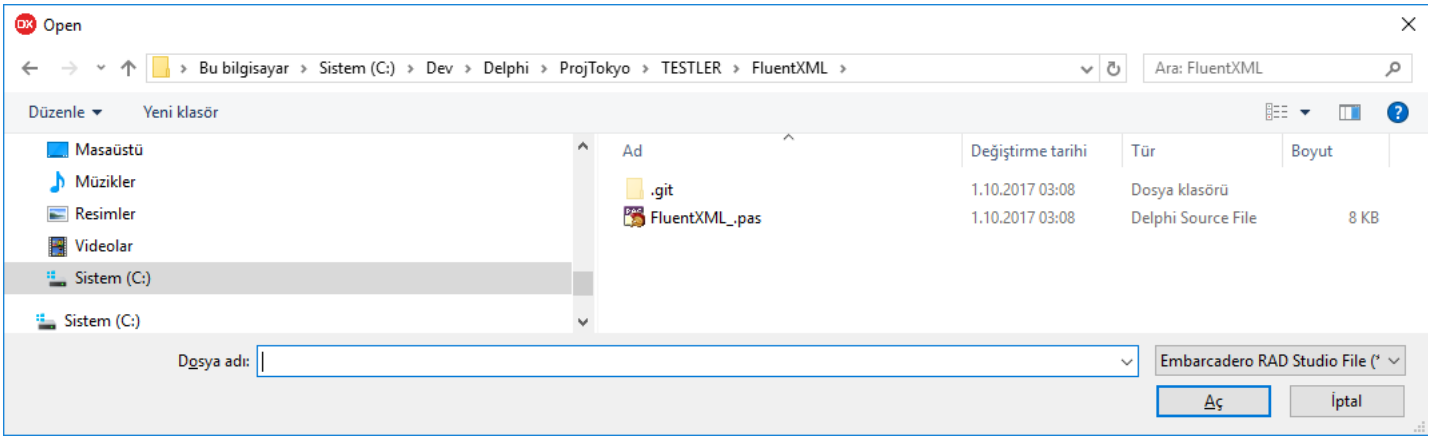


OK butonuna bastığımızda ise aşağıdaki gibi süreci takip edebileceğimiz bir ekran daha çıkıyor. Bu ekranda sihirbazın tam olarak ne yaptığı gösteriliyor.



En sonunda OK butonuna bastığımızda süreç tamamlanmış oluyor.

Bu noktada kontrol amaçlı olarak bir önceki pencerede DESTINATION olarak verdiğiniz klasöre gidip, dosyalarımızın kopyalanıp kopyalanmadığı teyit edebiliriz.



Bir proje ile tek başına bir pas dosyasının github üzerinden takip edilmesinde RAD Studio'nun dosyayı açma anlamında farklı bir davranışı vardır. Klonunu çekeceğimiz şey bir "proje" ise RAD Studio bunu doğrudan açacaktır, aksi durumda, yani sadece birkaç unitten oluşan bir depoda RAD Studio herhangi bir proje yükleme işlemi gerçekleştirmez.

Umarım Camiamıza faydalı olur.

Uğur PARLAYAN