LOGGING-NLOG TUTORİAL

Loglama nedir?

İlgili sistemde meydana gelen olayların kayıt altına alındığı dosyalama işlemidir. Direkt olarak başarılı ve başarısız sonuçlar tutulduğu gibi, işlemin neden başarısız olduğu bilgiside verilir. Bu bilgiler yazılım sektöründe önemli rol oynamaktadır. Her sistem için kayıtlar ve neden o kayıtların tutulduğuna dair bilgiler bir problemin çözümü için değerli bilgilerdir.

Loglama, son kullanıcılar, sistem yöneticileri, destek mühendisleri ve yazılım geliştirme ekipleri tarafından analize uygun raporlar üreterek yazılım hizmetini ve bakımını kolaylaştırır.

Loglama işlemleri için kullanabileceğimiz platformlar vardır. Bunlardan biri Nlog platformudur.

1. Kaydediciler mesajları yayınlar.
2. Hedefler mesajları kaydeder veya raporlar.
3. Log dosyaları raporları içerir.

NLOG

Nlog, .NET standartı dahil olmak üzere çeşitli .NET platformları için esnek ve ücretsiz bir loglama platformudur. Nlog, mesajları bir veya daha fazla hedefe kaydetmek için harika bir API sağlar. Nlog, aynı anda birkaç farklı alana (veritabanı,dosya,konsol) Log Kaydı tutmayı ve bu verileri yönetmenizi kolaylaştıran bir kütüphanedir.

Log Level

Tüm log mesajlarının “Level” adı verilen bir önem derecesi vardır.

Aşağıda belirtilmiştir:

-Fatal

-Error

-Warn

-Info

-Debug

-Trace

Log mesajlarını tam anlamıyla anlamak için her mesaja doğru “Level”’i koymak çok önemlidir. Log “Level”’leri, log filtreleme gibi Nlog’daki diğer özelliklerin yanı sıra her mesajın önem derecesini göstermek için kullanılır.

Fatal: Uygulamanın sonlandırılmasına neden olabilecek çok ciddi hata olaylarını temsil eder.

Error: Normal program yürütülmesini önleyecek, ancak yine de uygulamanın çalışmaya devam etmesine izin verebilecek, oldukça önemli hata olaylarını temsil eder.

Warn: Potansiyel sorunları gösteren, son kullanıcıları veya sistem yöneticilerini ilgilendiren potansiyel olarak zararlı durumları temsil eder.

Info: Son kullanıcılar ve sistem yöneticileri için anlamlı olabilecek ve uygulamanın ilerleyişini vurgulayabilecek bilgi mesajlarını içerir.

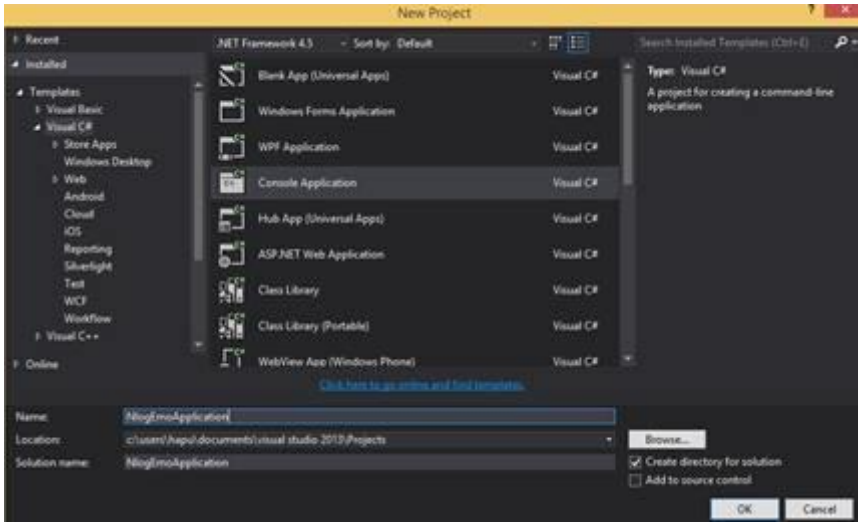
Debug: Uygulama geliştiricileri tarafından kullanılan nispeten ayrıntılı izleme denilebilir.

Trace: Uygulamanın dabranışlarıyla ilgili tüm ayrıntıları yakalar. Çoğunlukla tanı amaçlıdır ve Debug log düzeyünden daha ayrıntılı ve daha hassastır. Bu log düzeyi, uygulamanızda ne olduğunu veya kullanılan üçüncü taraf kitaplıklarında neler olduğunu görmeniz gereken durumlarda kullanılır.

Daha iyi pekiştirebilmek için bir Konsol uygulamasında görelim:

Adım 1

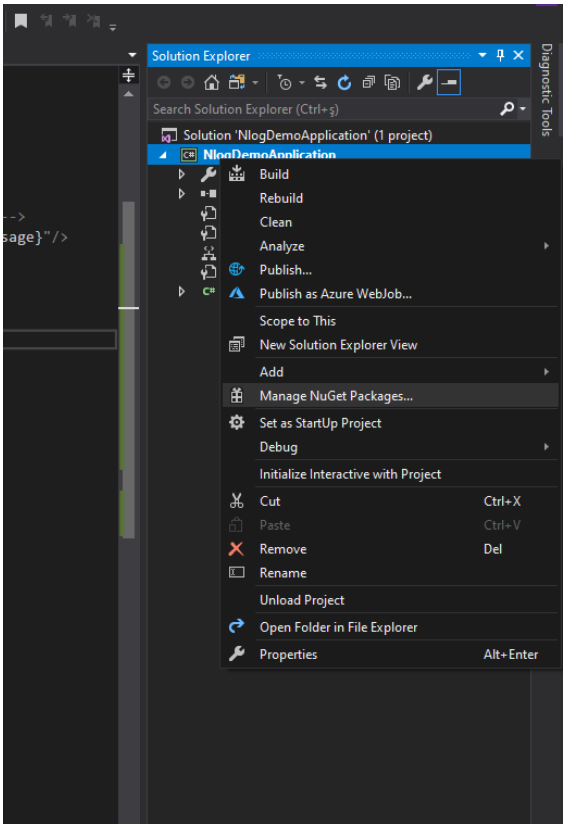
Bir C# konsol uygulaması oluşturalım ve buna NlogDemoApplication diyelim.



Adım 2

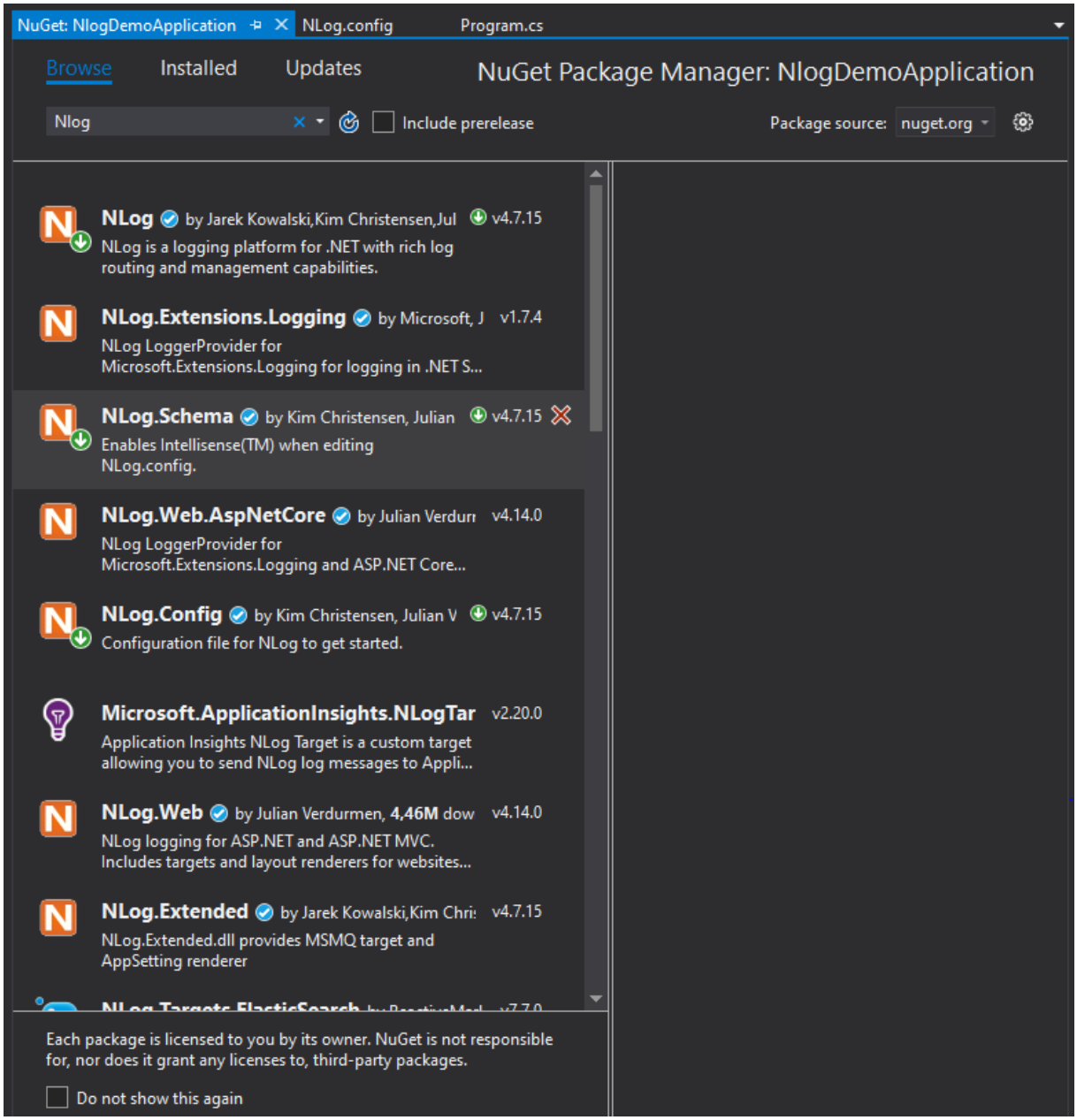
Uygulamaya Nlog paketini kuralım. Bunun için NuGet Paket Yöneticisi'ni kullanabiliriz.

NlogDemoApplication yazan kısma sağ tıklayıp Manage NuGet Packages 'e tıklamalıyız.



Adım 3

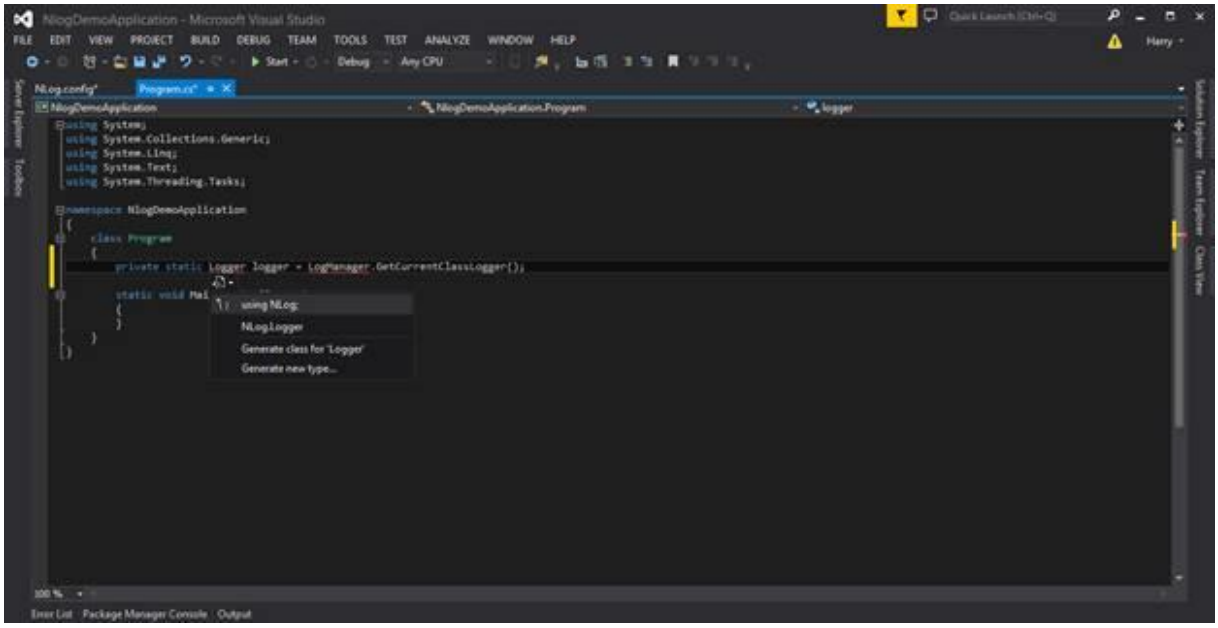
Gelen ekranda Browse kısmına Nlog yazdığımız zaman aşağıda görüldüğü gibi Nlog ve Nlog.Config dosyalarını kurmamız gerekmekte. Bu dosyaları kurduktan sonra kodlama kısmına geçebiliriz.



Adım 4

Program.cs dosyasını açalım ve sınıf başında Nlog kütüphanesinden yararlanarak özel bir static Logger nesnesi oluşturalım. Logger örnekleri oluşturmak için LogManager'i kullanmamız gerekiyor. Ad alanı da eklemeyi unutmayalım.

```
private static Logger logger = LogManager.GetCurrentClassLogger();
```



Adım 5

Main fonksiyonuna aşağıda verilen kodu yazalım. Bu yöntemde, statik bir hata mesajını log'a kaydetmek için sadece Nlog hata fonksiyonunu çağırırız.

```
1. static void Main(string[] args)
2.             {
3.                     logger.Error("This is an error message");
4.                     Console.Read();
5.             }
```

Adım 6

Şimdi, Nlog.Config' i açalım ve aşağıdaki verilen kodu yazalım.

```
1. <targets>
2.     <target name="console" xsi:type="Console" layout="${longdat
   e}|${message}"/>
3. </targets>
4.
5. <rules>
6.     <logger name="*" minlevel="Error" writeTo="console" />
7. </rules>
```

Targets etiketinin içine, anahtarları hedef olarak ekliyoruz; yani sonuçları nereye yazdıracağımızı ve yazdıracağımız log içeriğini belirliyoruz. Mesela yukarıdaki örnekte “layout” parametresinde belirttiğimiz gibi log mesajımızda tarih,saat ve belirlediğimiz mesaj gösterilecektir.

Rules etiketlerinin içine ise hangi level’de bir mesaj basacağımızı ve nereye basacağımızı tanımlarız.

Not

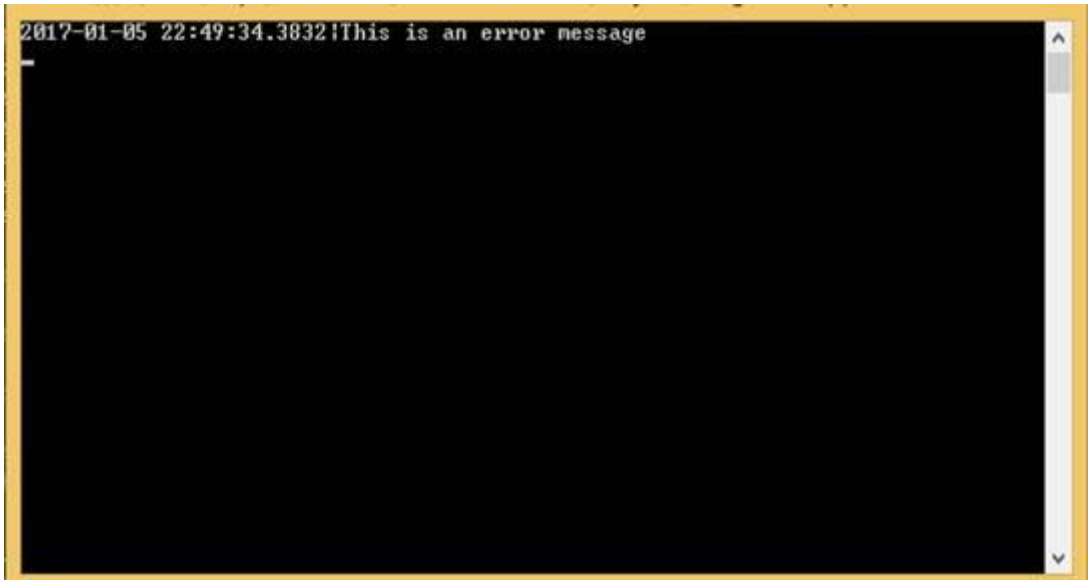
Xsi:type : Log için target tipi

Layout : Çıktı (Output)

Name : Target ismi

Adım 7

Konsol uygulamasını çalıştıralım. Aşağıda verilen çıktıyı göreceğiz. Konsol çıktı penceresine bir log mesajını bu şekilde bastırıyoruz.



Adım 8

Şimdi, bir hata mesajı kaydetmek için bir dosyayı hedefleyeceğiz. Yine Nlog.config dosyasını açalım ve aşağıda verilen kod satırını target etiketi altına ekleyelim.

```
<target
name= "file"  xsitype= "File"  fileName= "D\logs\NLog.log"  layout= "${long
date}|${message}" />
```

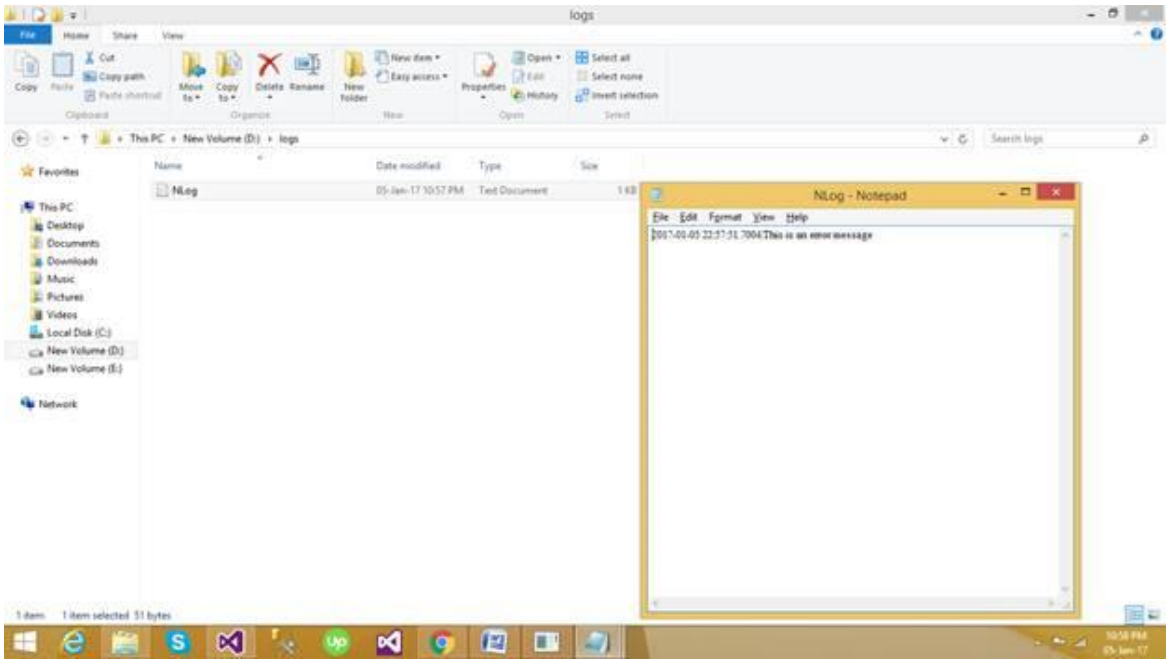
Mevcut kuralda(rules etiketi içerisine), dosyayı eklemeliyiz. Şimdi, kodumuz aşağıda gösterildiği gibi görünecektir.

```
1. <targets>
2.     <target name="console" xsitype="Console" layout="${longdate
   }|${message}"/>
3.      <target name="file" xsitype="File" fileName="D\logs\NLog.l
   og" layout="${longdate}|${message}"/>
4. </targets>
5.
6.   <rules>
7.     <logger name="*" minlevel="Error" writeTo="console,file" />

8.   </rules>
```

Adım 9

Kodu çalıştıralım. Şimdi çıktımızı, biri konsol penceresinde ve ikincisi yazdığımız hedef dosyada olmak üzere iki yerde göreceğiz. Dosya hedefinde belirtilen yolu kontrol edelim. Referans için aşağıdaki resime bakabilirsiniz.



Adım 10

Şimdi hedefimiz log mesajımızı e-mail yoluyla göndermek olacak. Yine target etiketinin içine aşağıda gösterildiği gibi bir hedef daha ekleyelim.

```
1. <target name="sendMail" xsitype="Mail"
2.           subject="Application Error Log"
3.           to="XXXXXXXXXXXXXXXXXXXXXXXXXX"
4.           from="XXXXXXXXXXXXXXXXXXXXXXXX"
5.           body="${longdate}|${message}"
6.           enableSsl="true"
7.           smtpAuthentication="Basic"
8.           smtpServer="smtp.gmail.com"
9.           smtpUserName="XXXXXXXXXXXXXXXXXX"
10.               smtpPassword="XXXXXXXXXXXXXXXXXXX"
11.               smtpPort="587"/>
12.       </targets>
```

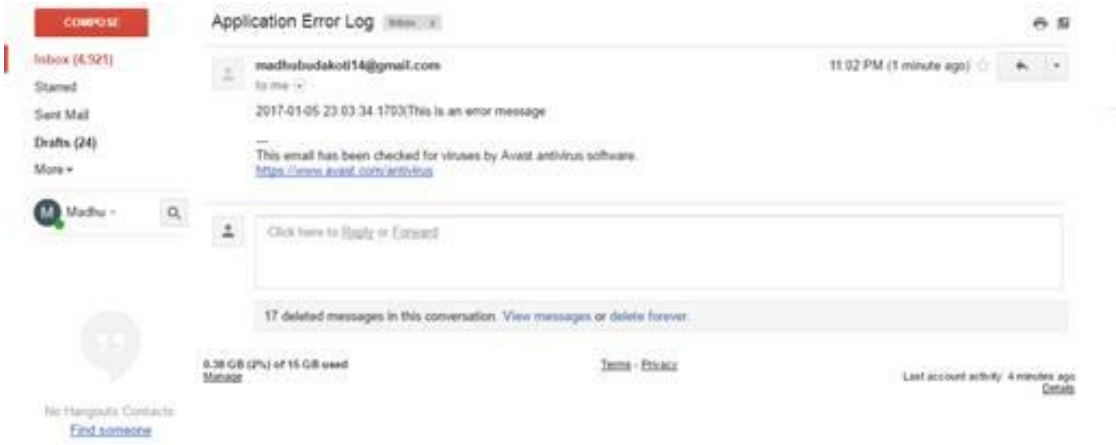
SendEmail olan hedefin adı ve mevcut kural aşağıda gösterilmiştir.

```
1. <logger
   name= "*"  minlevel= "Hata"  writeTo= "console,file,sendMail"
   />
```

Tip	Posta
subject	İstediğimiz ismi verebiliriz. Mailimizin konusu olacak.
To	E-mail kime gönderilecek?
From	E-mail nereden gönderilecek?
Body	E-mail'de görüntülenecek bilgiler
enableSSl	Güvenli e-posta için
SmtpAuthentication	True
SmtpUserName	E-mail gönderilecek olan mailin kullanıcı adı
SmtpPassword	E-mail gönderilecek olan mailin şifresi
SmtpServer	E-maili göndermek istediğiniz sunucu.
Smptport	Sunucu port numarası

Adım 11

Uygulamayı çalıştıralım. Bu durumda, çıktı aşağıda gösterildiği gibi olacaktır. Bu kodu denediğimizde, belirttiğimiz e-posta adresinden sonucu kontrol edebiliriz. E-posta adresimizin güvenlik ayarlarını düzenlemeyi unutmayalım.



Bunlara benzer şekilde, Info, Fatal, Debug vb. gibi loglara kaydedebiliriz.