

# M183 Applikationssicherheit Implementieren

Tutorial zur Übung 2-Factor-Authentication with OTP Token  
(SMS & Email)

Version 1	26.10.2017	Jürg Nietlispach
-----------	------------	------------------

## Contents

Idee .....	3
Setup SMS Gateway .....	3
Setup eMail Gateway .....	3
Setup ASP.NET MVC Applikation .....	3
Herangehensweise Login-Prozedere .....	3
Setup SMS Gateway .....	4
Setup Email Gateway .....	6
Setup eines neuen ASP.NET MVC Projekt.....	9
Login-Prozedere .....	11

## Idee

Two Factor Authentication kommt in verschiedenen Varianten vor. In der vorliegenden Übung soll ein Login-Prozedere (ASP.NET-MVC-Applikation) durch die Abfrage eines One Time Passworts (OTP) erweitert werden. Das OTP wird per SMS bzw. per Email versendet. Für beide Fälle sollen unterschiedliche Token-Charakteristiken verwendet werden.

## Setup SMS Gateway

1. Erstellen eines Accounts bei einem SMS-Gateways (z.B. Nexmo)
2. Demo-SMS an eigene Nummer senden (via CURL an das API)

## Setup eMail Gateway

1. Account bei Mailgun (mailgun.com) erstellen und aktivieren
2. Demo-Email an eigene Emailadresse senden (via CURL an das API)
- 3.

## Setup ASP.NET MVC Applikation

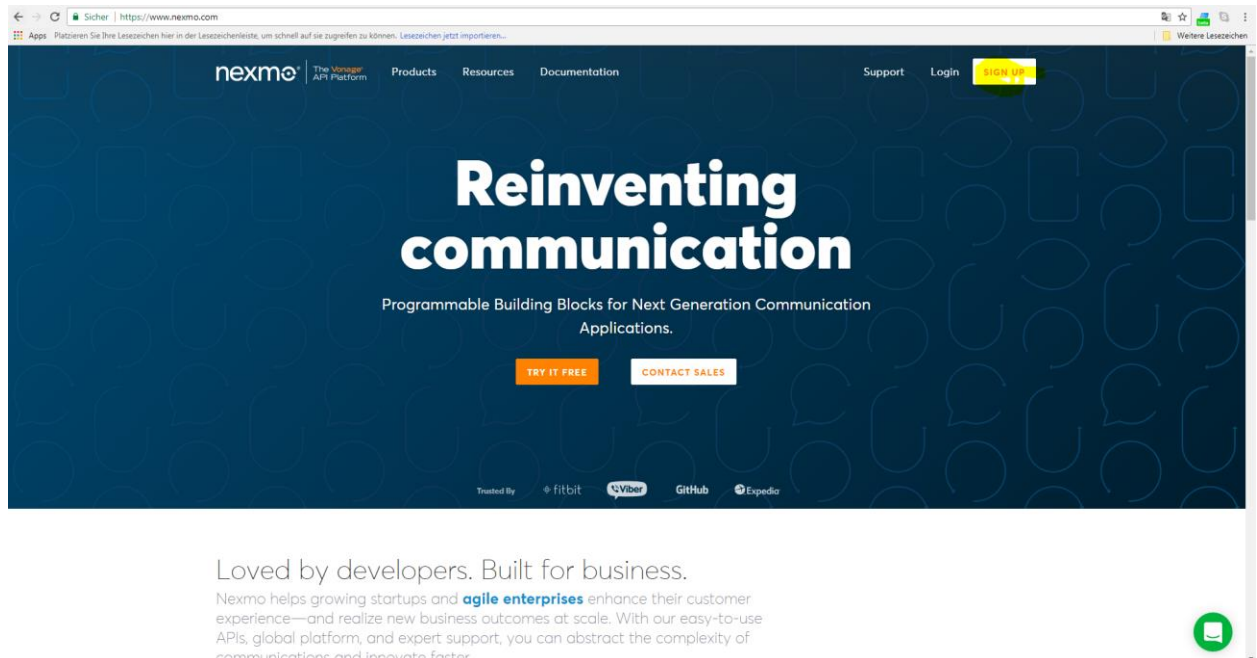
1. Erstellen eines neuen ASP.NET Projektes

## Herangehensweise Login-Prozedere

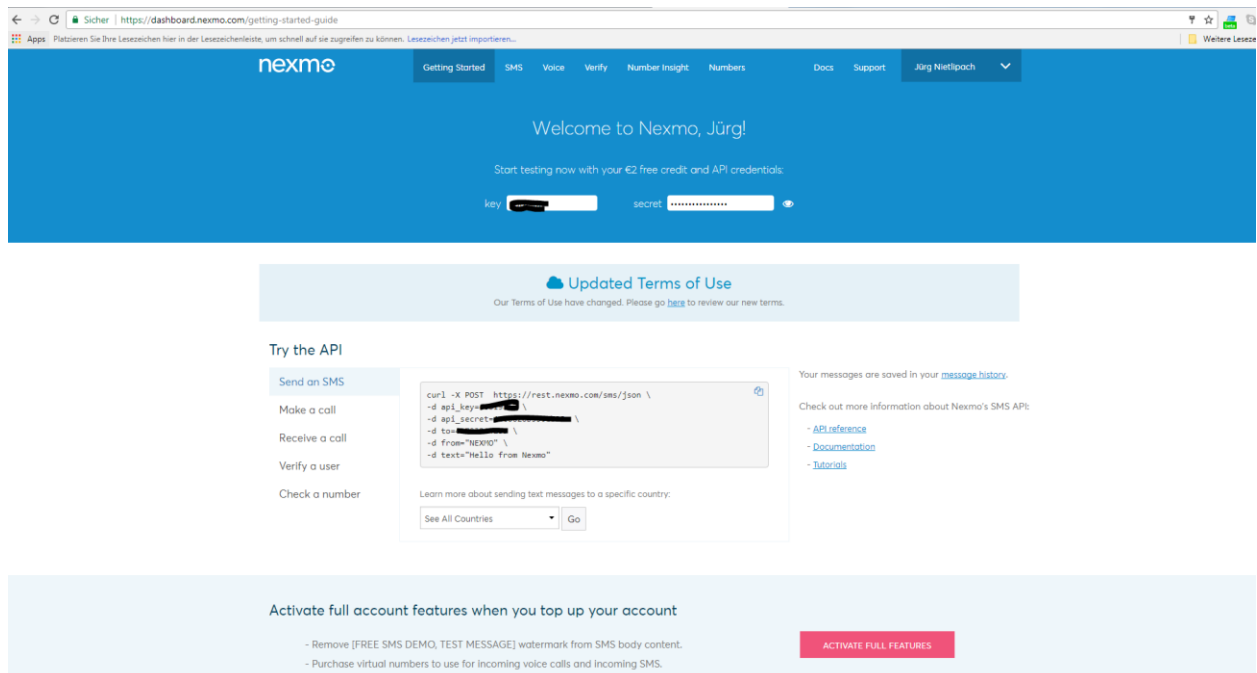
1. Loginmaske mit Benutzernamen und Passwort erstellen
2. Business-Logik erstellen, welches einfach Benutzernamen und Passwort prüft
3. Sind die Eingaben korrekt soll Business-Logik auch das Generieren und versenden des OTPs ausführen.
4. Wurde ein Token erfolgreich versendet, soll ein drittes Feld für das OTP angezeigt werden
5. Bei Eingabe des Tokens soll die Business-Logik der Applikation das Token prüfen und eine entsprechende Antwort senden.

## Setup SMS Gateway

1. Account erstellen bei nexmo.com



2. Nach dem Login sind auf dem Dashboard alle nötigen Informationen ersichtlich, welche es für das Versenden einer SMS Nachricht benötigt:



3. Es soll nun ein Test-SMS ans eigene Gerät gesendet werden. Hierzu kopiert man den CURL-Befehl (mit den eigenen Zugangsdaten).

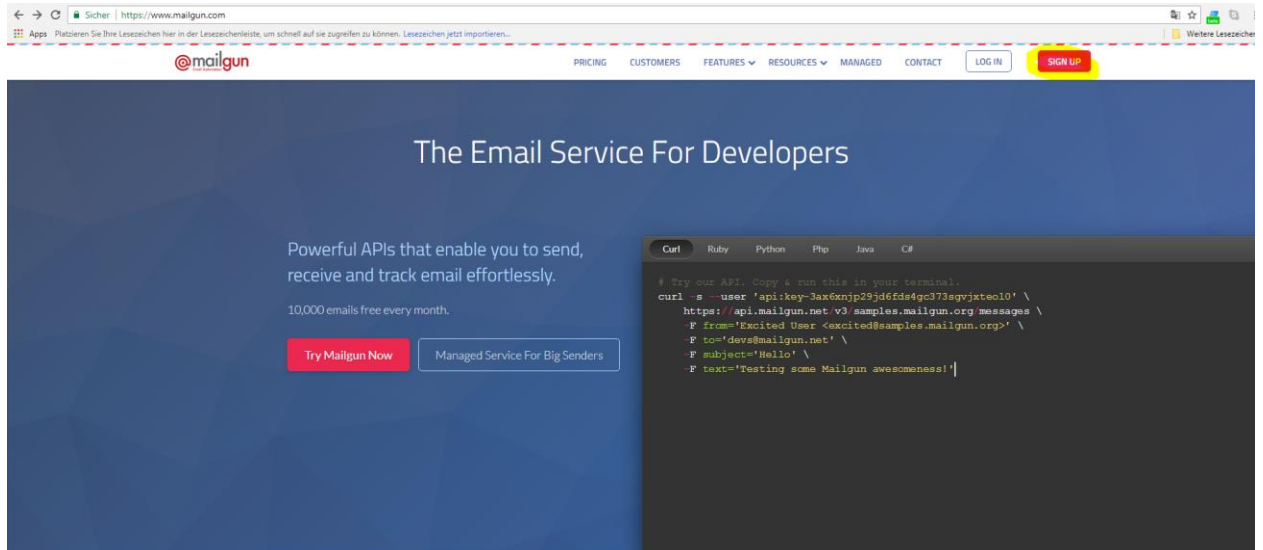
```
curl -X POST https://rest.nexmo.com/sms/json \
-d api_key=[REDACTED] \
-d api_secret=[REDACTED] \
-d to=[REDACTED] \
-d from="NEXMO" \
-d text="Hello from Nexmo"
```

Möglicher CURL-Online-Dienst: [onlinecurl.com](http://onlinecurl.com)

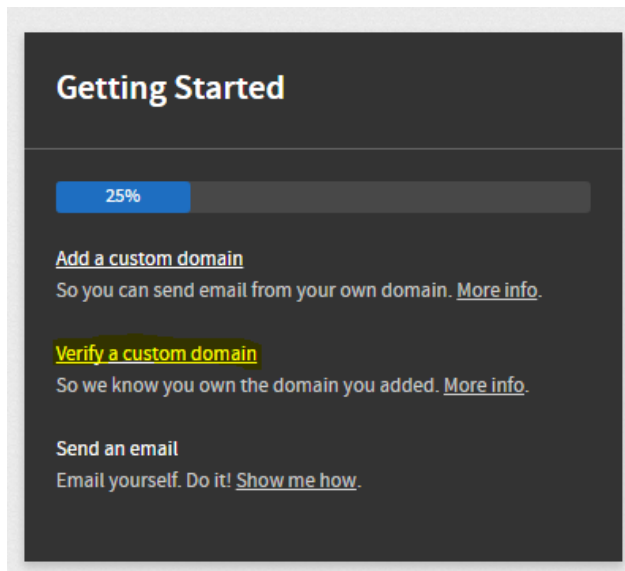
4. Auf dem Telefon sollte nun eine Nachricht erscheinen

## Setup Email Gateway

1. Bei Mailgun.com ein Benutzerkonto eröffnen und einloggen



2. Custom Domain angeben und verifizieren



uns sich selber als Authorized Recipient hinzufügen:

<div> Add New Domain Authorized Recipients </div>			
	State	Domain Name	Outgoing (30d)
	Active	sandbox72a01b2843be4149ae092942e3bd0452.mailgun.org	0

Und Emailadresse angeben.

Account			
Settings Security Authorized Recipients			
Authorized Recipients			
<div> Invite New Recipient Delete Selected </div>			
<input type="checkbox"/>	State	Email	
<input type="checkbox"/>	Verified		

- Nun soll via CURL eine Email-Nachricht versendet werden (send me an email)

## Getting Started

25%

[Add a custom domain](#)  
 So you can send email from your own domain. [More info.](#)

[Verify a custom domain](#)  
 So we know you own the domain you added. [More info.](#)

[Send an email](#)  
 Email yourself. Do it! [Show me how.](#)

Ist verlinkt auf die Dokumentation des APIs und den CURL-Befehl für einen Request:

The screenshot shows the Mailgun documentation page for sending via API. The left sidebar contains a navigation menu with links like 'Quickstart Guide', 'Sending Email', 'Verify Your Domain', 'Common DNS Providers', 'Receiving Email', 'Tracking Events', 'User Manual', 'Libraries', 'API Reference', 'FAQ', and 'Email Best Practices'. The main content area is titled 'Send via API' and includes a 'Run this:' section with a code block containing a curl command. Below the code block, it explains what's happening: Mailgun assembles a MIME message, adds log entries to the full text search index, and delivers the email. It also mentions finding the API key in the dashboard.

umentation.mailgun.com/en/latest/quickstart-sending.html#send-via-api  
ier in der Lesezeichenleiste, um schnell auf sie zugreifen zu können. [Lesezeichen jetzt importieren...](#)

Code sample preference: **curl** Ruby Python PHP Java C# Go Node.js [Edit](#)

**Quickstart Guide**  
[Sending Email](#)  
**Send with SMTP or API**  
[Verify Your Domain](#)  
[Common DNS Providers](#)  
[Receiving Email](#)  
[Tracking Events](#)  
**User Manual**  
**Libraries**  
**API Reference**  
**FAQ**  
**Email Best Practices**  
[You can find your API Key in your Control Panel](#)

## Send via API

Run this:

```
curl -s --user 'api:YOUR_API_KEY' \
  https://api.mailgun.net/v3/YOUR_DOMAIN_NAME/messages \
  -F from:'Excited User <mailgun@YOUR_DOMAIN_NAME>' \
  -F to:YOU@YOUR_DOMAIN_NAME \
  -F to:bar@example.com \
  -F subject:'Hello' \
  -F text:'Testing some Mailgun awesomness!'
```

What's actually happening:

- Mailgun assembles a MIME message.
- Added the log entries to our full text search index.
- Delivered the email.

You can find your secret API key on your [dashboard](#).

4. Mit onlinecurl.com lässt sich dieser Befehl absetzen, was dann ein Emailversand an die im Befehl angegebene Email-Adresse auslöst:

The screenshot shows the onlinecurl.com website. It features a header with the 'RIGOR' logo and a 'TRY RIGOR' button. The main content area is titled 'ONLINE CURL' and includes a description: 'CURL FROM THE CLOUD! PING YOUR SERVERS AND WEBPAGES FROM ANYWHERE AND RECEIVE A NEATLY FORMATTED RESPONSE. ADD IN DIFFERENT OPTIONS TO CUSTOMIZE YOUR CURL REQUEST.' Below this is a large input field labeled 'Enter curl command or URL'. To the right of this field is a 'CUSTOMIZE' section with an 'ADD OPTION' button. Below the input field is a section labeled 'ENTER YOUR EMAIL ADDRESS TO RECEIVE THE FREE REPORT.' with an input field containing 'email@example.com' and a 'START YOUR CURL' button.

← → ↻ 📄 [onlinecurl.com](#) Apps Platzieren Sie Ihre Lesezeichen hier in der Lesezeichenleiste, um schnell auf sie zugreifen zu können. [Lesezeichen jetzt importieren...](#) [Weitere Lesezeichen](#)

**RIGOR** [TRY RIGOR](#) Check out our other free apps →

## ONLINE CURL

CURL FROM THE CLOUD! PING YOUR SERVERS AND WEBPAGES FROM ANYWHERE AND RECEIVE A NEATLY FORMATTED RESPONSE. ADD IN DIFFERENT OPTIONS TO CUSTOMIZE YOUR CURL REQUEST.

↓

➤ Enter curl command or URL

CUSTOMIZE

➕ ADD OPTION

↓

ENTER YOUR EMAIL ADDRESS TO RECEIVE THE FREE REPORT.

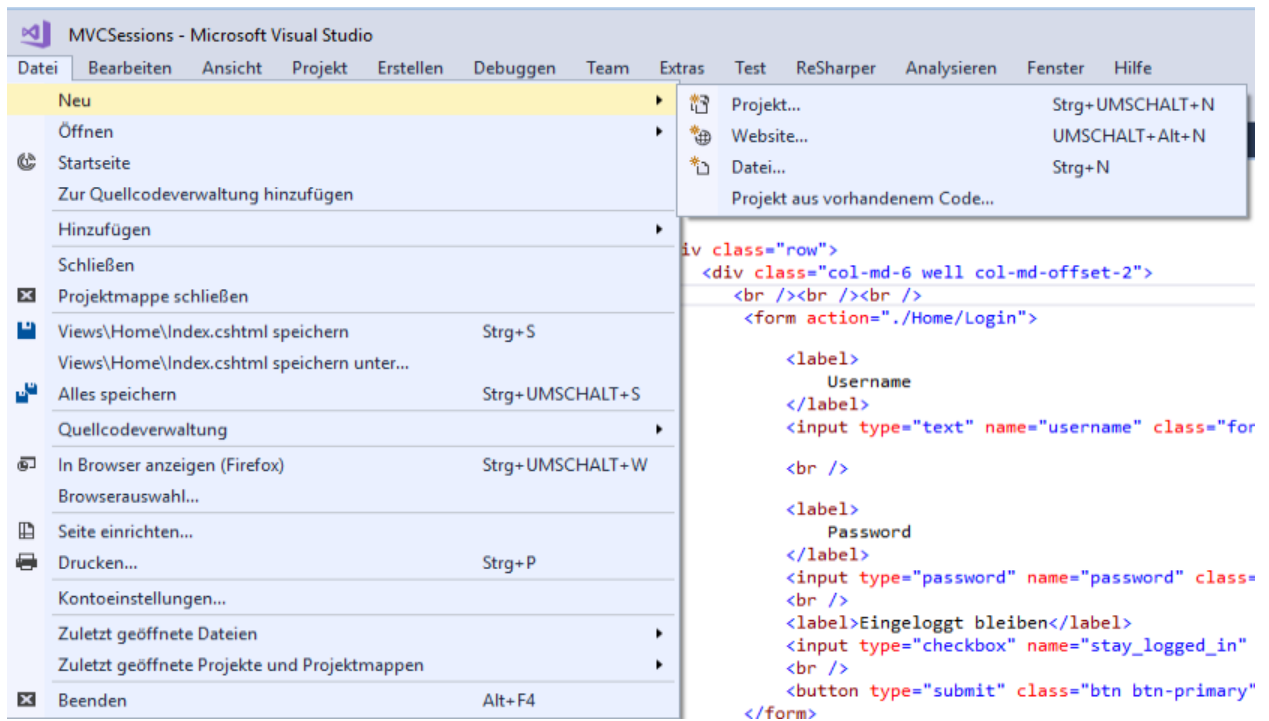
email@example.com

[START YOUR CURL](#)

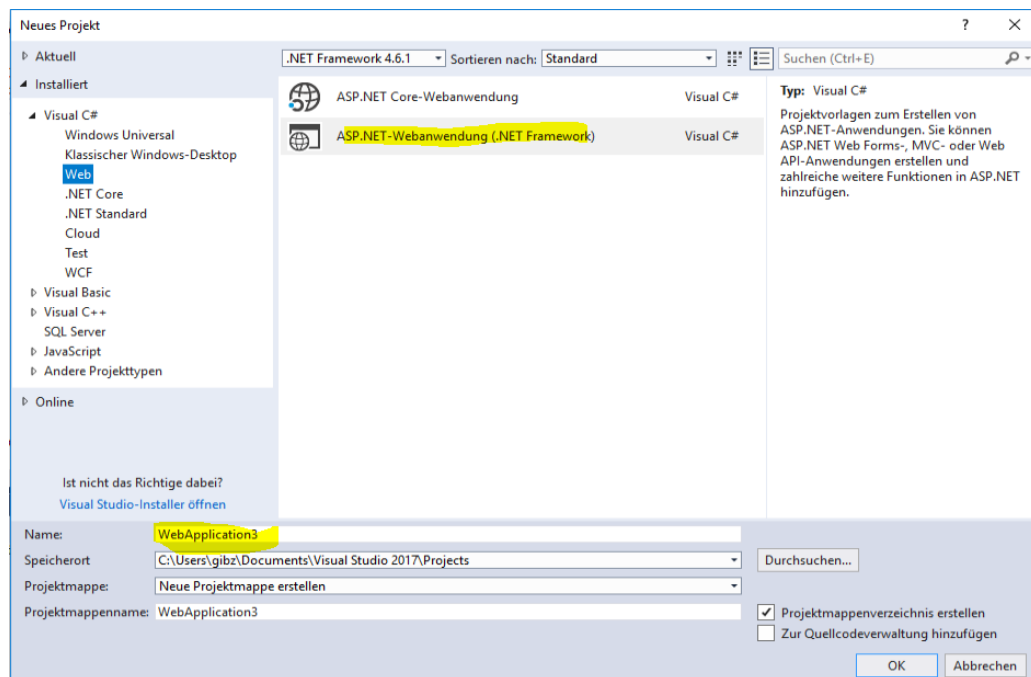


## Setup eines neuen ASP.NET MVC Projekt

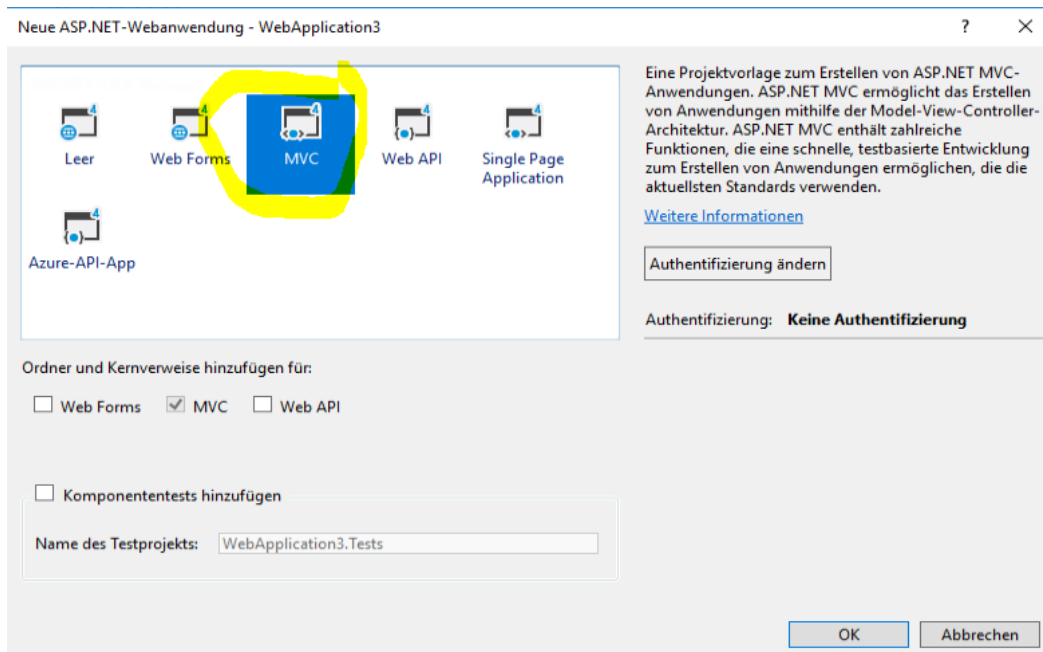
1. In Visual Studio ein neues Projekt erstellen:



2. Entsprechendes Projekt auswählen: C# Web-Projekt, ASP.NET Projekt. Namen für Projekt angeben:

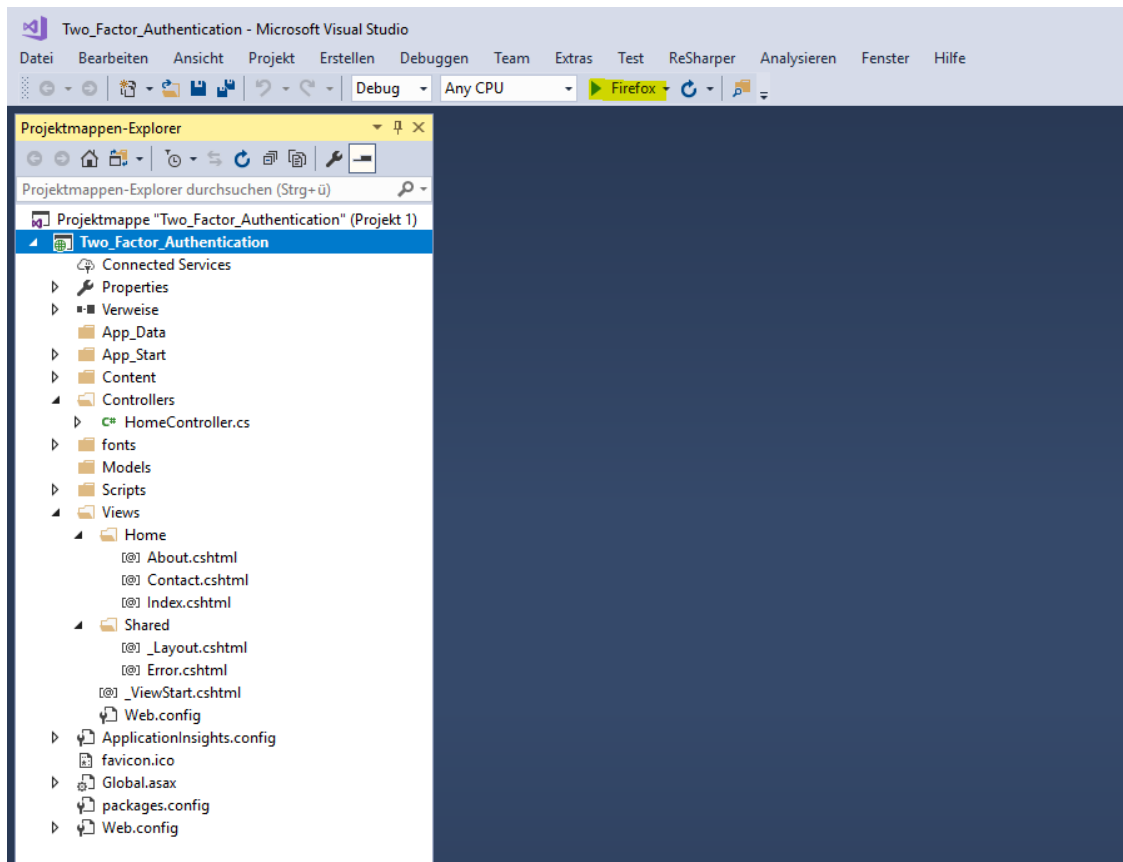


3. MVC-Projekt auswählen (erstellt ein MVC-Boilerplate-Projekt inkl. Bootstrap Layout Engine)

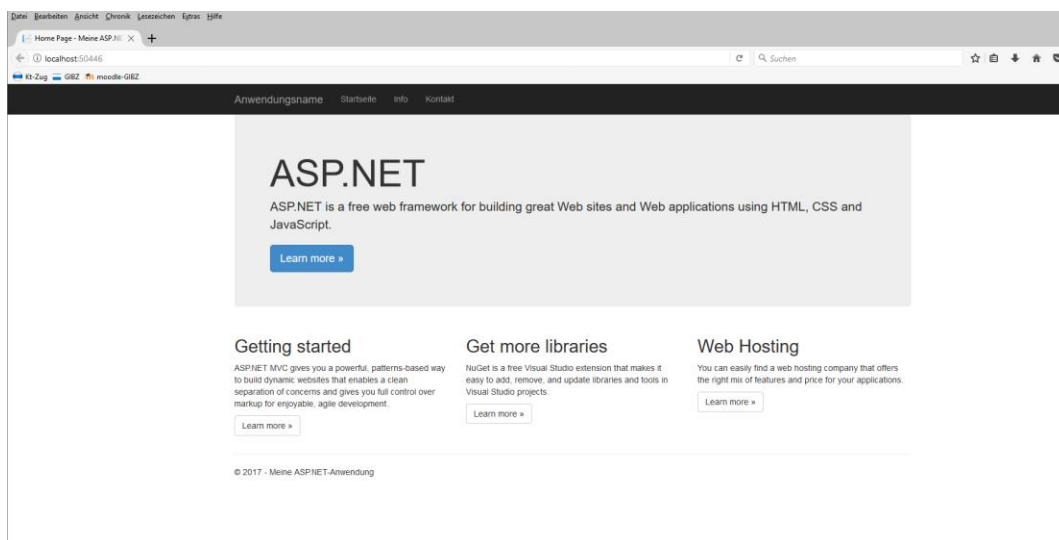


## Login-Prozedere

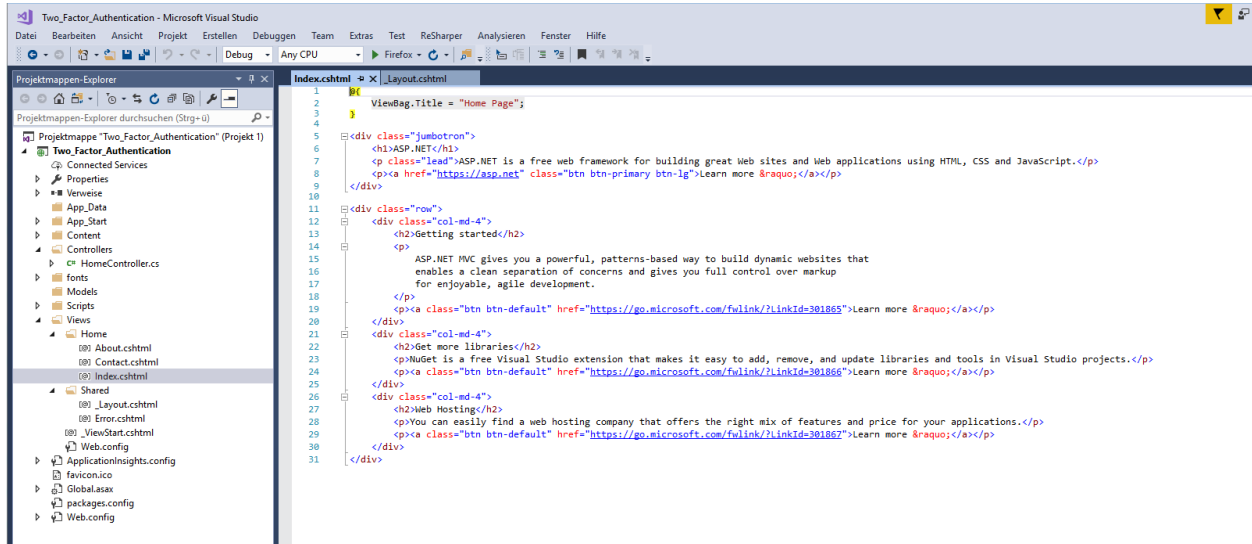
Visual Studio generiert beim Erstellen einer MVC Applikation folgende Verzeichnisstruktur. Die rudimentäre Applikation kann mit dem Browser gestartet werden:



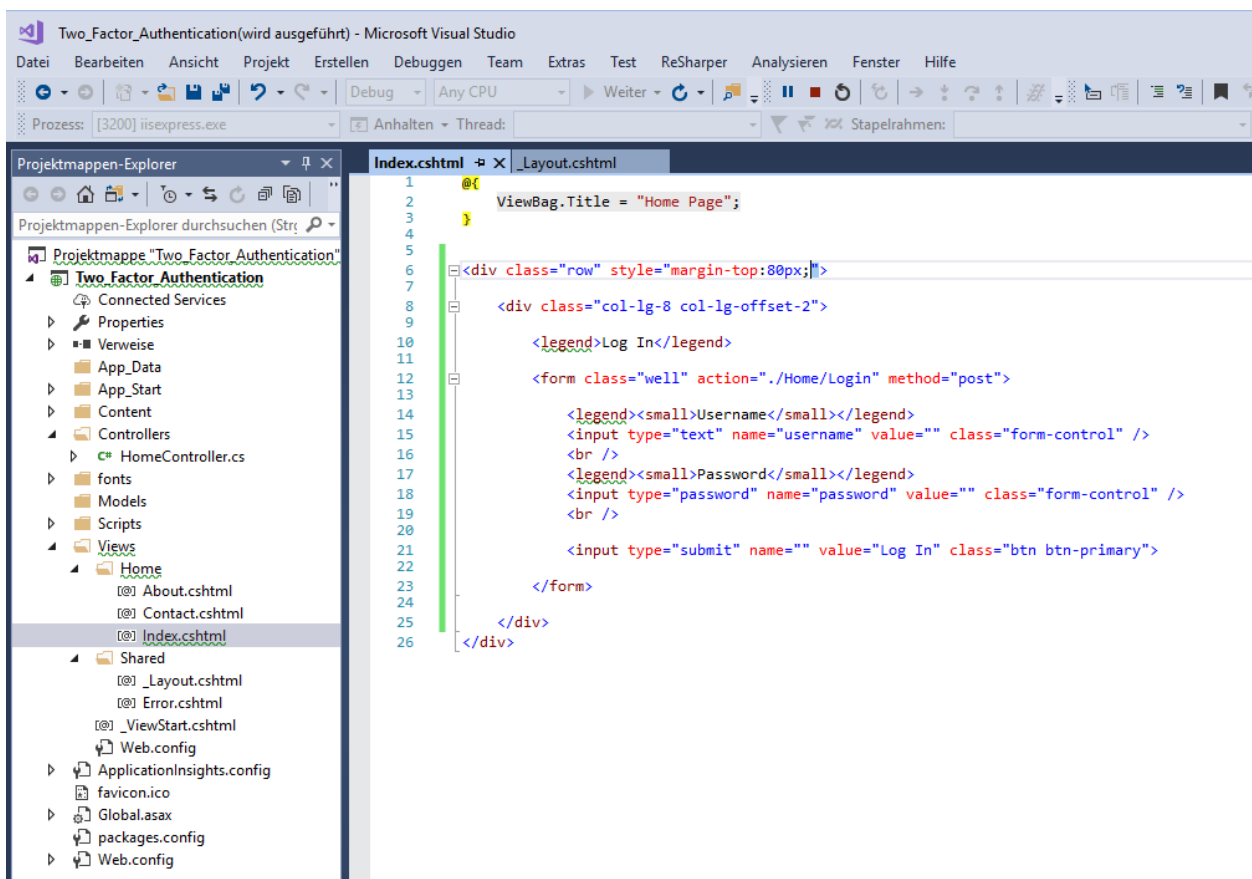
Resultat im Browser:



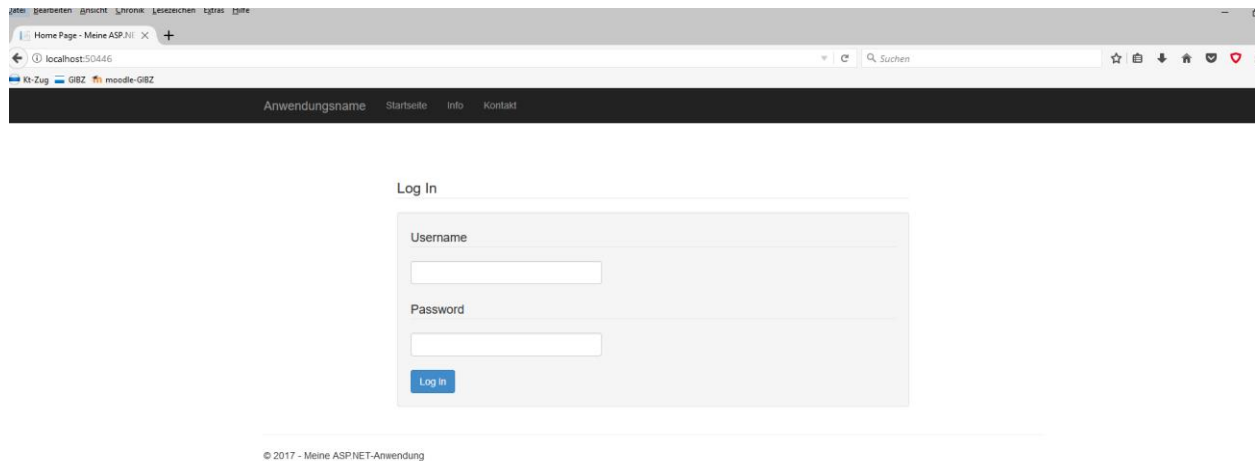
Im folgenden File kann nun das Login-Formular (Anstelle der Info-Boxen) eingebunden werden:



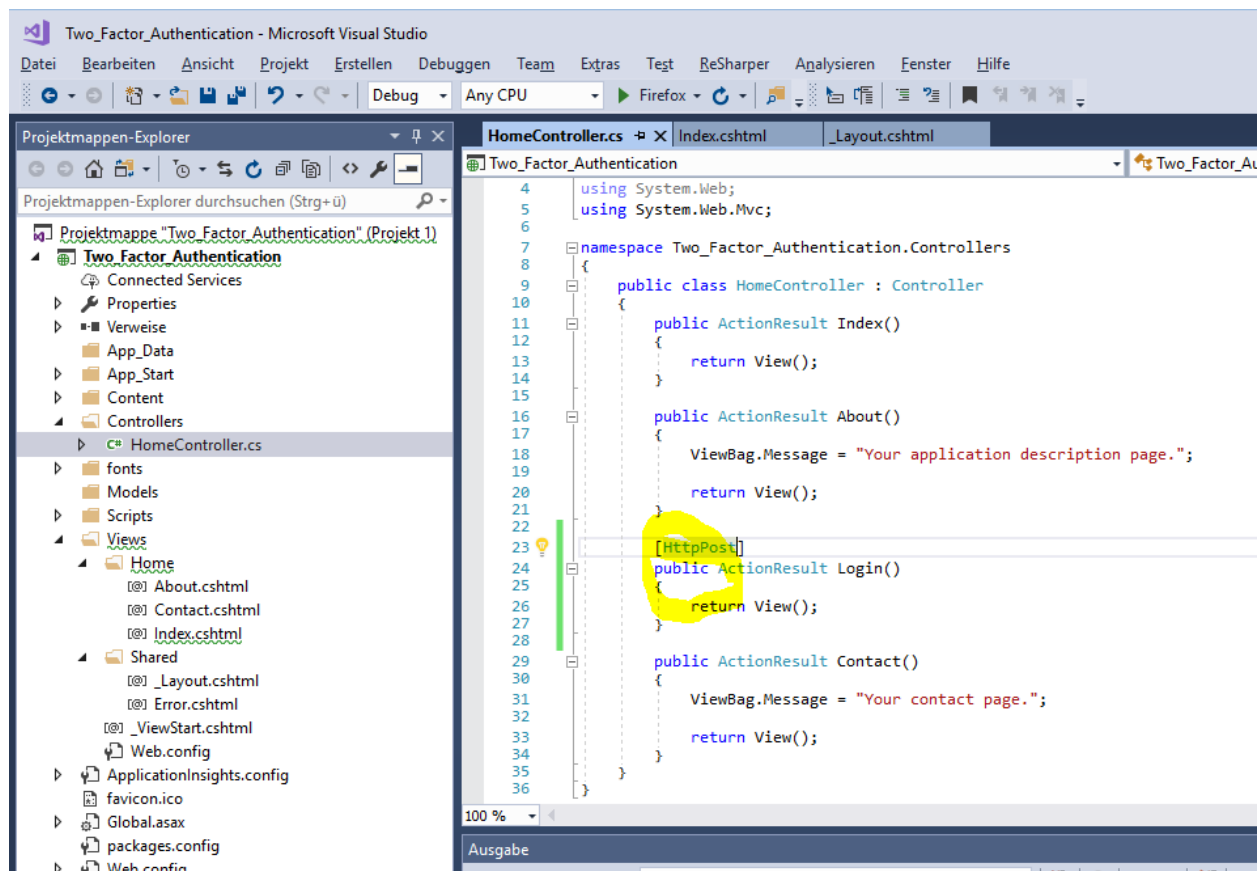
Dies sieht im Quellcode dann folgendermassen aus:



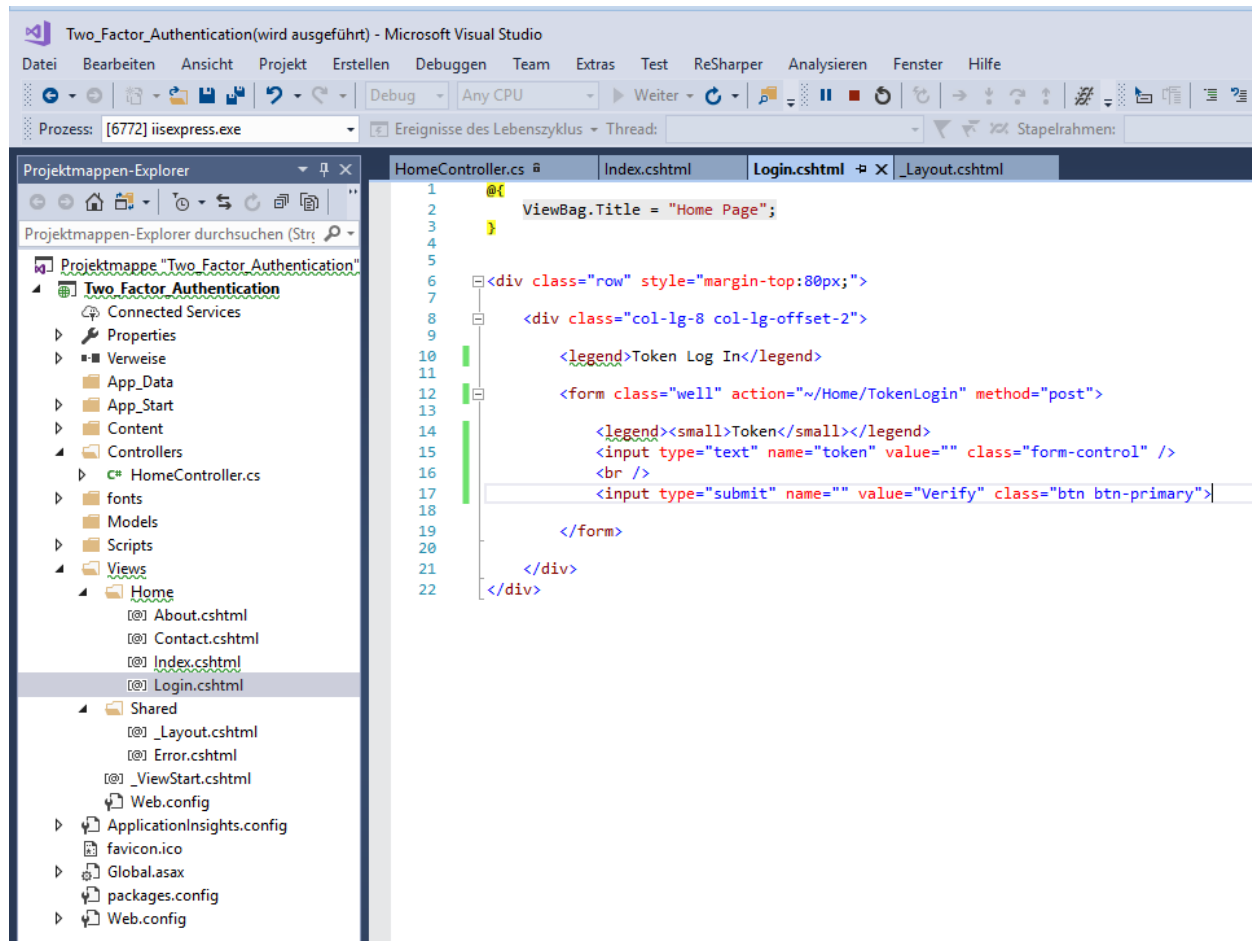
Und im Browser dann so:



Nun muss das Abschicken des Formulars von einem Controller gehandelt werden. Hierzu erweitert man den HomeController um die folgende Funktion:



Da diese Funktion eine View() retourniert (ein Login-Formular für die Eingabe des Tokens), muss hier noch ein neues Template Login.cshtml mit folgendem Inhalt erstellt werden:



Nun müssen Benutzernamen und Passwort überprüft werden. Der Einfachheit halber sind diese im Source-Code fix hinterlegt (diese sind normalerweise gehasht in einer Datenbank gespeichert).

```
[HttpPost]
public ActionResult Login()
{
    var username = Request["username"];
    var password = Request["password"];

    if (username == "test" && password == "test")
    {
```

Stimmen Benutzernamen und Passwort überein, soll nun das Token über einen zweiten Kommunikationskanal (SMS / EMAIL) versendet werden. Die Routine für den SMS – Versand via nexmo sieht dann folgendermassen aus:



```

if (username == "test" && password == "test")
{
    var request = (HttpWebRequest)WebRequest.Create("https://api.mailgun.net/v3/[REDACTED]@mailgun.org/messages");

    // Add Basic Auth
    String encoded = System.Convert.ToBase64String(System.Text.Encoding.GetEncoding("ISO-8859-1").GetBytes("api:[REDACTED]:[REDACTED]"));
    request.Headers.Add("Authorization", "Basic " + encoded);

    var secret = "TEST_SECRET";

    //var postData = "[REDACTED]";
    var postData = "from=Test User <mailgun@[REDACTED]@mailgun.org>";
    postData += "&to=[REDACTED]";
    postData += "&subject=Secret-Token";
    postData += "&text=\"\" + secret + \"\"";
    var data = Encoding.ASCII.GetBytes(postData);

    request.Method = "POST";
    request.ContentType = "application/x-www-form-urlencoded";
    request.ContentLength = data.Length;

    using (var stream = request.GetRequestStream())
    {
        stream.Write(data, 0, data.Length);
    }

    var response = (HttpWebResponse)request.GetResponse();
    var responseString = new StreamReader(response.GetResponseStream()).ReadToEnd();

    ViewBag.Message = responseString;
}

```

Nachdem das Token versendet wurde (SMS oder Email), kann dieses in der Loginform für das Token eingegeben werden:

localhost:50446/Home/Login

Kt-Zug GIBZ moodle-GIBZ

Anwendungsname Startseite Info Kontakt

### Token Log In

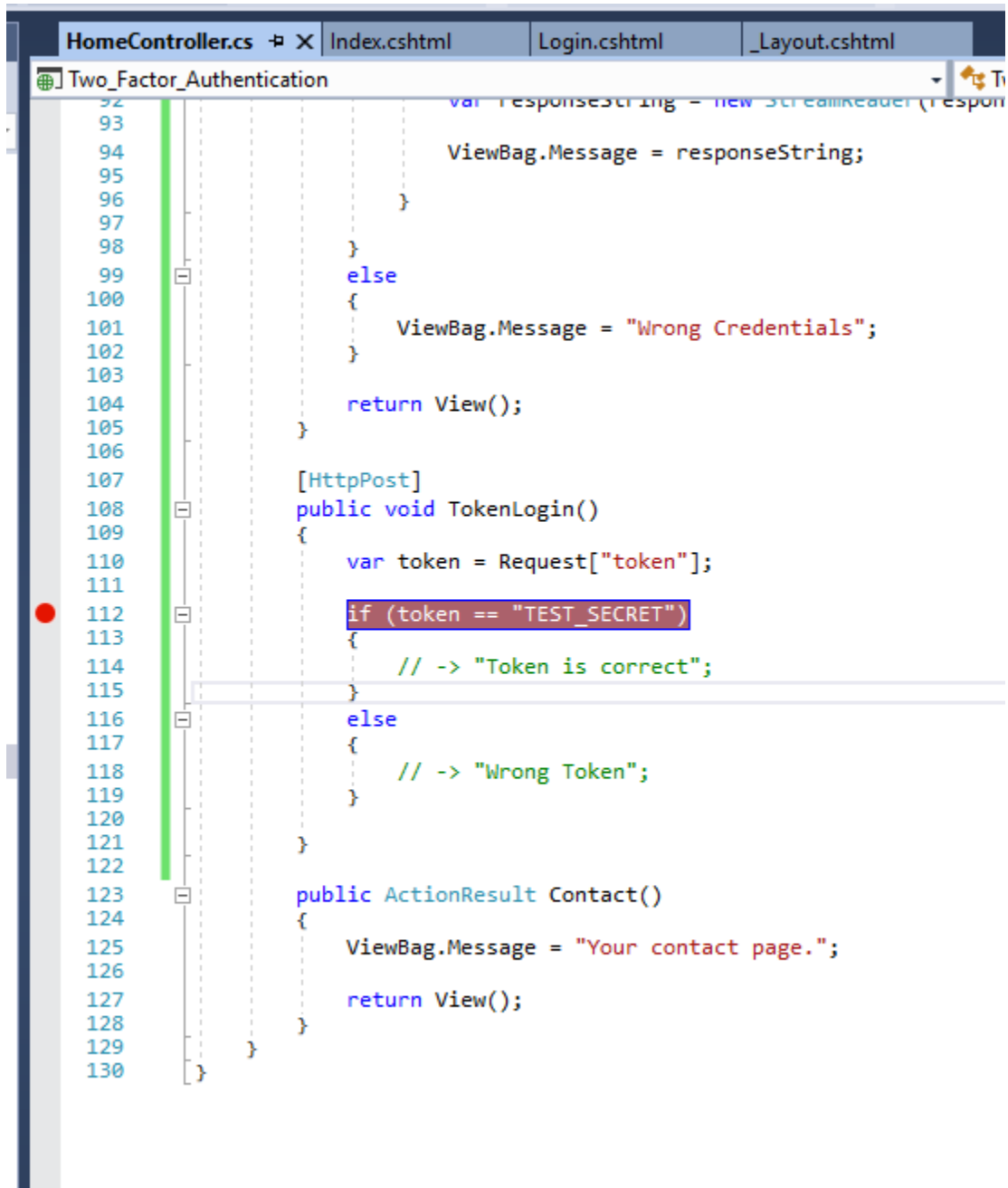
Token

Verify

© 2017 - Meine ASP.NET-Anwendung

Dieses Token-Login wird an den HomeController geschickt, wo wir noch eine Route ergänzen müssen:





```
HomeController.cs X Index.cshtml Login.cshtml _Layout.cshtml
Two_Factor_Authentication
92 var responseString = new StreamReader(respon
93
94 ViewBag.Message = responseString;
95
96 }
97
98 }
99 else
100 {
101     ViewBag.Message = "Wrong Credentials";
102 }
103
104 return View();
105 }
106
107 [HttpPost]
108 public void TokenLogin()
109 {
110     var token = Request["token"];
111
112     if (token == "TEST_SECRET")
113     {
114         // -> "Token is correct";
115     }
116     else
117     {
118         // -> "Wrong Token";
119     }
120 }
121
122
123 public ActionResult Contact()
124 {
125     ViewBag.Message = "Your contact page.";
126
127     return View();
128 }
129
130 }
```

Hier wird keine View retourniert – im Moment können die Resultate einfach mit dem Debugger nachvollzogen werden.. Idealerweise findet hier das Session-Init-Handling und den Redirect ins Backendsystem statt.