

M183 Applikationssicherheit Implementieren

Tutorial zur Übung Single Sign On

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

Contents

Idee	3
Setup Google Account.....	3
Setup ASP.NET MVC Applikation	3
Herangehensweise Login-Prozedere	3
Setup Auth Server / Google API Account.....	4
Setup Client / Button	10
Get Profile Infos (Client-Side).....	15
Authenticate with Backend.....	16
Sign Out Button.....	21

Idee

In dieser Übung soll der Single Sign On dienst von Google in eine .NET MVC Applikation integriert werden.

Setup Google Account

1. Setup Auth Server (Google API Account Infos) for Client / Button Integration (Google-Account wird benötigt). <https://developers.google.com/identity/sign-in/web/devconsole-project>

Setup ASP.NET MVC Applikation

1. Erstellen eines neuen ASP.NET Projektes

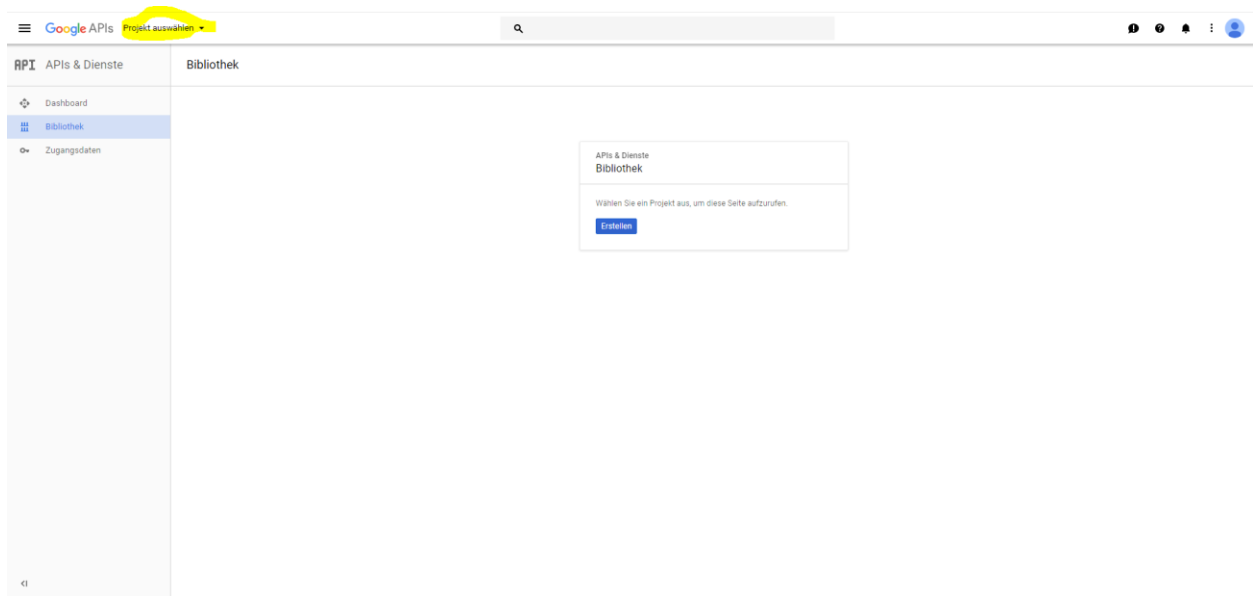
Herangehensweise Login-Prozedere

1. Setup Client / Button. <https://developers.google.com/identity/sign-in/web/sign-in>
2. Get Profile Infos (Client Side): <https://developers.google.com/identity/sign-in/web/people>
3. Backend-Side: Authenticate with Backend Server (in order to get a session):
<https://developers.google.com/identity/sign-in/web/backend-auth>
4. Sing-Out Button anzeigen

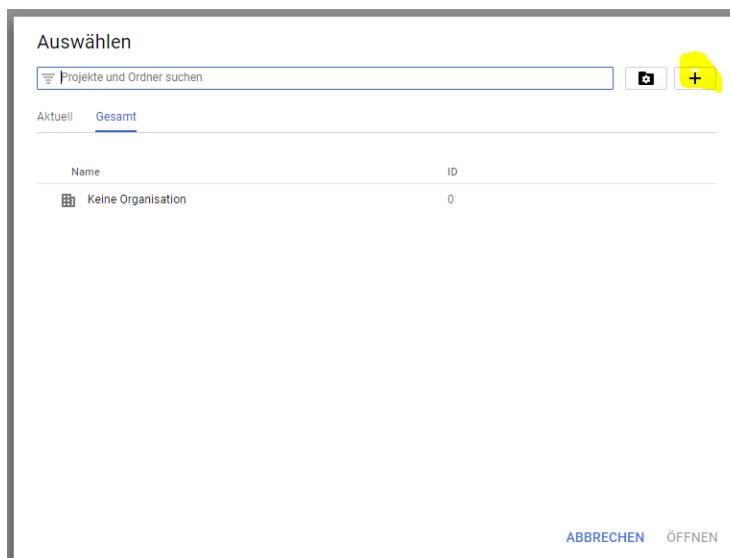
Setup Auth Server / Google API Account

<https://console.developers.google.com/projectselector/apis/library>


1. Neues API Projekt eröffnen




2. Plus klicken




3. Projektnamen z.B. „M183 SSO 1“ angeben




Neues Projekt

 Ihr Kontingent umfasst noch 12 Projekte. [Weitere Informationen.](#)

Projektname 

M183

Die Projekt-ID lautet: m183-181120  [Bearbeiten](#)

Ich möchte Updates zu Funktionsankündigungen, Anwendungsvorschlägen, Feedbackumfragen und besonderen Angeboten erhalten.

☒ Ja ☐ Nein

Ich stimme zu, dass ich die [Dienste und zugehörigen APIs](#) nur unter der Voraussetzung nutzen darf, dass ich die geltenden [Nutzungsbedingungen](#) einhalte.

☒ Ja ☐ Nein

Erstellen

Abbrechen

4. OAuth-Details angeben

Zugangsdaten

Anmeldedaten **OAuth-Zustimmungsbildschirm** Domainbestätigung

E-Mail-Adresse ?

juerg.nietlispach@gmail.com

Produktname, den Nutzer sehen ?

M183

Homepage-URL (optional)

https:// oder http://

Produktlogo-URL (optional) ?

http://www.example.com/logo.png



So sehen Endnutzer Ihr Logo.
Max. Größe: 120 x 120 px

URL der Datenschutzerklärung

Optional, bis Sie Ihre App bereitstellen

https:// oder http://

URL der Nutzungsbedingungen (optional)

https:// oder http://

Speichern

Abbrechen



Der Zustimmungsbildschirm wird Nutzern gezeigt, wenn Sie mit Ihrer Client-ID Zugriff auf deren private Daten anfordern. Er erscheint für alle Anwendungen, die in diesem Projekt registriert sind.

Sie müssen eine E-Mail-Adresse und einen Produktnamen bereitstellen, damit OAuth funktioniert.

5. Anmeldedaten für OAuth erstellen

APIs

Anmeldedaten

Für den Zugriff auf APIs sind Anmeldedaten erforderlich. Aktivieren Sie die APIs, die Sie verwenden möchten, und erstellen Sie dann die erforderlichen Anmeldedaten. Je nach API sind ein API-Schlüssel, ein Dienstkonto oder eine OAuth 2.0-Client-ID erforderlich. Weitere Informationen finden Sie in der API-Dokumentation.

Anmeldedaten erstellen ▾

API-Schlüssel

Identifiziert Ihr Projekt durch einen einfachen API-Schlüssel, um Kontingent und Zugriff zu prüfen

OAuth-Client-ID

Fordert die Zustimmung des Nutzers an, damit Ihre App auf die Daten des Nutzers zugreifen kann

Dienstkontoschlüssel

Aktiviert mithilfe von Robot-Konten Server-zu-Server-Authentifizierung auf App-Ebene

Auswahlhilfe

Beantworten Sie einige Fragen, um herauszufinden, welche Form der Anmeldung am besten geeignet ist.

6. Daten angeben

← Client-ID erstellen

Anwendungstyp

- ☒ Webanwendung
- ☐ Android [Mehr erfahren](#)
- ☐ Chrome-App [Mehr erfahren](#)
- ☐ iOS [Mehr erfahren](#)
- ☐ PlayStation 4
- ☐ Sonstige

Name

M183

Einschränkungen

Geben Sie JavaScript-Quellen oder Weiterleitungs-URLs oder beides ein.

Autorisierte JavaScript-Quellen

Zur Verwendung bei Anfragen über einen Browser. Dies ist die Ursprungs-URL der Clientanwendung. Sie darf weder einen Platzhalter (`http://*.ihrebeispielurl.de`) noch einen Pfad (`http://ihrebeispielurl.de/subdir`) enthalten. Wenn Sie einen nichtstandardmäßigen Port verwenden, müssen Sie ihn in der Ursprungs-URL angeben.

`http://localhost:49670`

Autorisierte Weiterleitungs-URLs

Für die Verwendung mit Anfragen über einen Webserver. Dies ist der Pfad in Ihrer Anwendung, zu dem Nutzer nach der Authentifizierung mit Google weitergeleitet werden. An den Pfad wird der Autorisierungscode für den Zugriff angehängt. Muss ein Protokoll aufweisen. Darf keine URL-Fragmente oder relativen Pfade enthalten. Öffentliche IP-Adressen sind nicht zulässig.

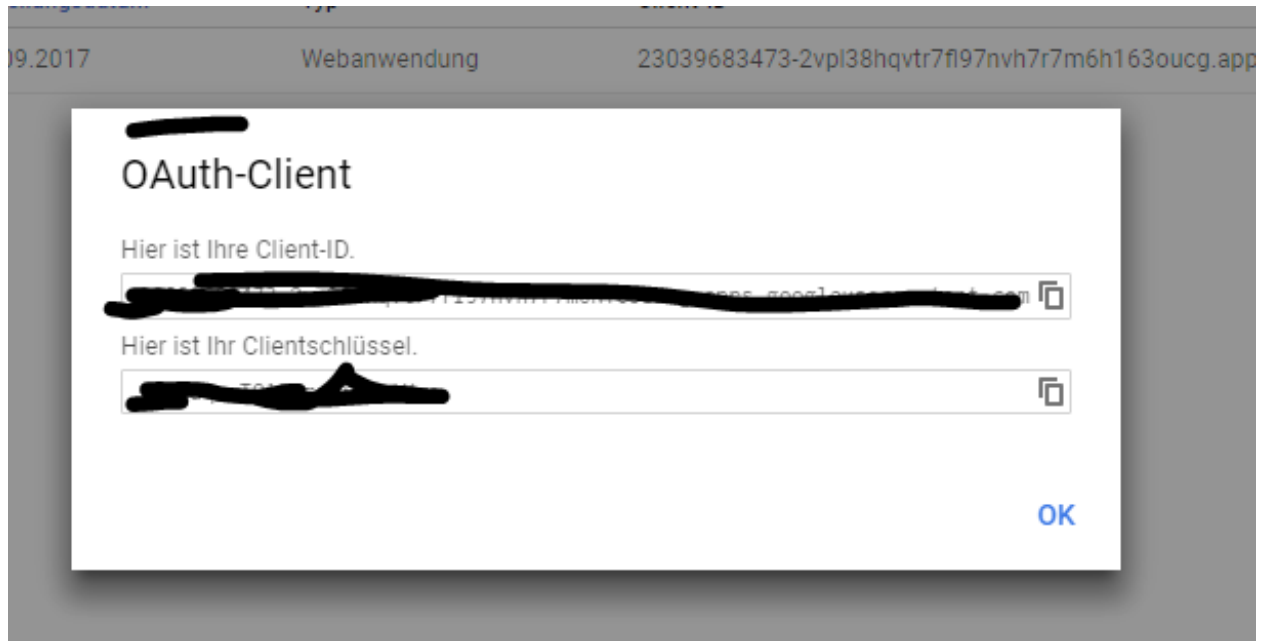
`http://www.example.com/oauth2callback`

Erstellen

Abbrechen

ACHTUNG: Autorisierte JavascriptQuellen. Hier muss die Portnummer der lokal ausgeführten C# Applikation hinterlegt werden!

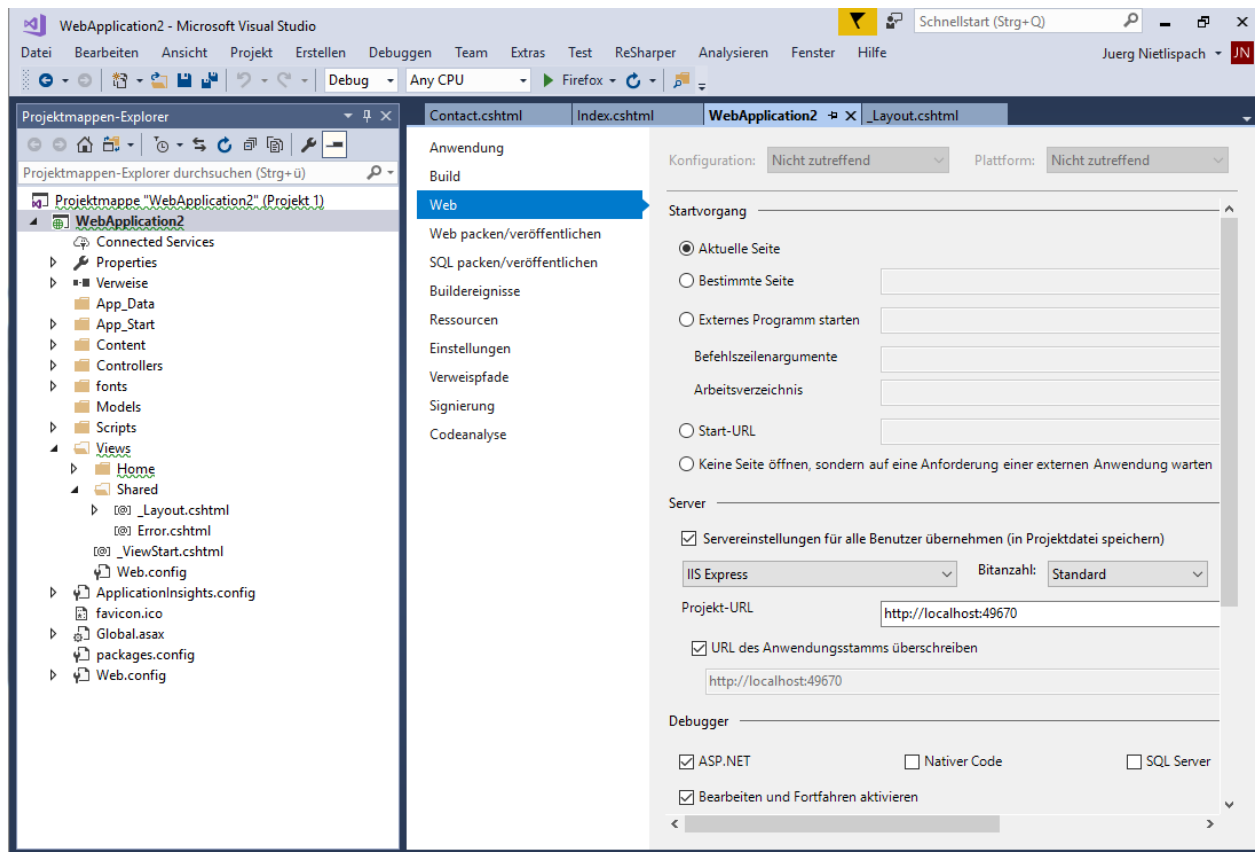
7. Zugangsdaten in Visual Studio zwischenspeichern



Setup Client / Button

<https://developers.google.com/identity/sign-in/web/sign-in>

1. Neues oder bisheriges .NET MVC – Projekt erstellen



2. Im _Layout.cshtml-File das folgende META-Tag einbinden im <head>



ACHTUNG: content="" -> mit den Zugangsdaten aus der vorigen Aufgabe Schritt 7 anreichern.

3. Google-Platform.js Javascript Library einbinden im _Layout.cshtml vor dem </body>-Tag

```
<script src="https://apis.google.com/js/platform.js" async defer></script>
</body>
```

4. Login-Formular mit Google-Button im index.cshtml einbinden



```
1  @{
2      ViewBag.Title = "Home Page";
3  }
4
5
6  <div class="row">
7      <div class="col-md-8 col-md-offset-2">
8          <br /><br />
9          <legend>Log In</legend>
10         <form>
11
12             <label>Username</label>
13             <input type="text" class="form-control" />
14
15             <br />
16             <label>Password</label>
17             <input type="password" class="form-control" />
18
19             <br />
20             <input type="submit" class="btn btn-primary" />
21
22             <div class="g-signin2 pull-right" data-onsuccess="onSignIn"></div>
23
24         </form>
25
26     </div>
27 </div>
28 </div>
```

5. Webapplikation starten. Resultat:

Home Page - Meine ASP.NET X +

localhost:49670/Home/Index

Kt-Zug GIBZ moodle-GIBZ


Anwendungsname Startseite Info Kontakt

Log In

Username

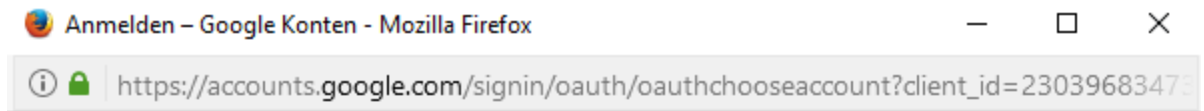
Password

Daten absenden

 Anmelden

© 2017 - Meine ASP.NET-Anwendung

6. Beim Klick auf den Google-Button erscheint nun ein Pop-Up mit der Anmeldemaske von den Google Accounts:



Konto auswählen

weiter zu [M183](#)



Jürg Nietlispach



[Redacted text]



Anderes Konto verwenden

Deutsch ▼

Hilfe

Datenschutz

Nutzungsbedingungen

7. Nach der Anmeldung und dem Redirect, ändert der Button seinen Status auf „Signed In“

The screenshot shows a web browser window with the address bar displaying 'localhost:49670/Home/Index'. The browser's address bar includes a search icon and the text 'Suchen'. Below the address bar, there are three tabs: 'Kt-Zug', 'GIBZ', and 'moodle-GIBZ'. The main content area of the browser shows a dark navigation bar with the text 'Anwendungsname' and three links: 'Startseite', 'Info', and 'Kontakt'. Below the navigation bar, there is a 'Log In' section. This section contains two input fields: 'Username' and 'Password'. Below the 'Password' field is a blue button labeled 'Daten absenden'. To the right of the 'Daten absenden' button is a button labeled 'Signed in' with a Google logo icon. At the bottom of the page, there is a footer that reads '© 2017 - Meine ASP.NET-Anwendung'.

localhost:49670/Home/Index

Kt-Zug GIBZ moodle-GIBZ

Anwendungsname Startseite Info Kontakt

Log In

Username

Password

Daten absenden

Signed in

© 2017 - Meine ASP.NET-Anwendung

Get Profile Infos (Client-Side)

<https://developers.google.com/identity/sign-in/web/people>

1. Folgendes Javascript Snipped im _Layout.cshtml einfügen, um id_token und Profile-Infos zu erhalten. Skript NACH der platform.js Library einfügen.

```
<script src="https://apis.google.com/js/platform.js" async defer></script>
<script type="text/javascript">
    function onSignIn(googleUser) {
        var id_token = googleUser.getAuthResponse().id_token;

        var profile = googleUser.getBasicProfile();
        console.log('ID: ' + profile.getId()); // Do not send to your backend! Use an ID token instead.
        console.log('Name: ' + profile.getName());
        console.log('Image URL: ' + profile.getImageUrl());
        console.log('Email: ' + profile.getEmail()); // This is null if the 'email' scope is not present.
        console.log('ID-TOKEN: ' + id_token);
    }
</script>
```

2. In der Firefox Konsole (-> F12 drücken) werden die Logs ausgegeben (console.log)

Authenticate with Backend

<https://developers.google.com/identity/sign-in/web/backend-auth>

1. Erweitern des Javascript Snippets aus dem File _Layout.cshtml um einen POST request an das "eigene" Backend, sobald ein Ticket verfügbar ist:

```
<script type="text/javascript">

    function onSignIn(googleUser) {

        var id_token = googleUser.getAuthResponse().id_token;

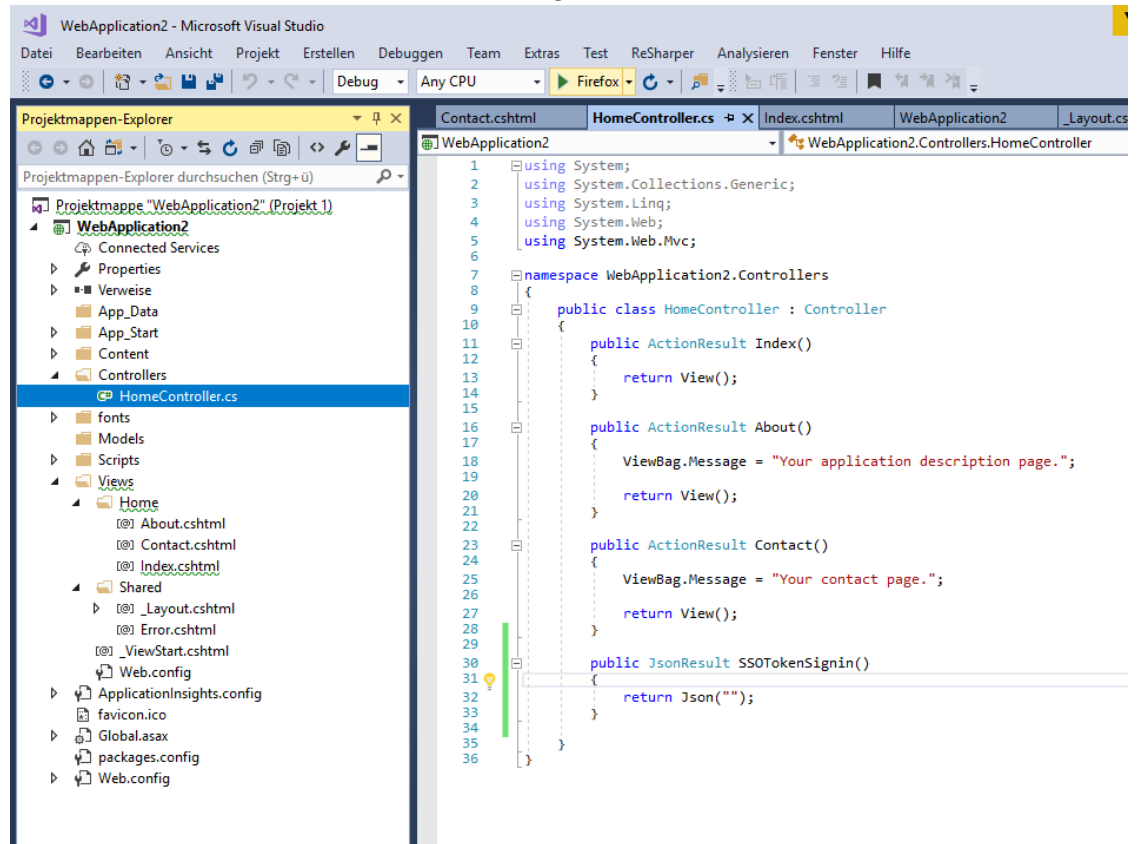
        var profile = googleUser.getBasicProfile();
        console.log('ID: ' + profile.getId()); // Do not send to your backend! Use an ID token instead.
        console.log('Name: ' + profile.getName());
        console.log('Image URL: ' + profile.getImageUrl());
        console.log('Email: ' + profile.getEmail()); // This is null if the 'email' scope is not present.
        console.log('ID-TOKEN: ' + id_token);

        var xhr = new XMLHttpRequest();
        xhr.open('POST', 'http://localhost:49670/Home/SSOTokenSignIn');
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        xhr.onload = function () {
            console.log('Signed in as: ' + xhr.responseText);
        };
        xhr.send('idtoken=' + id_token);
    }

</script>
```

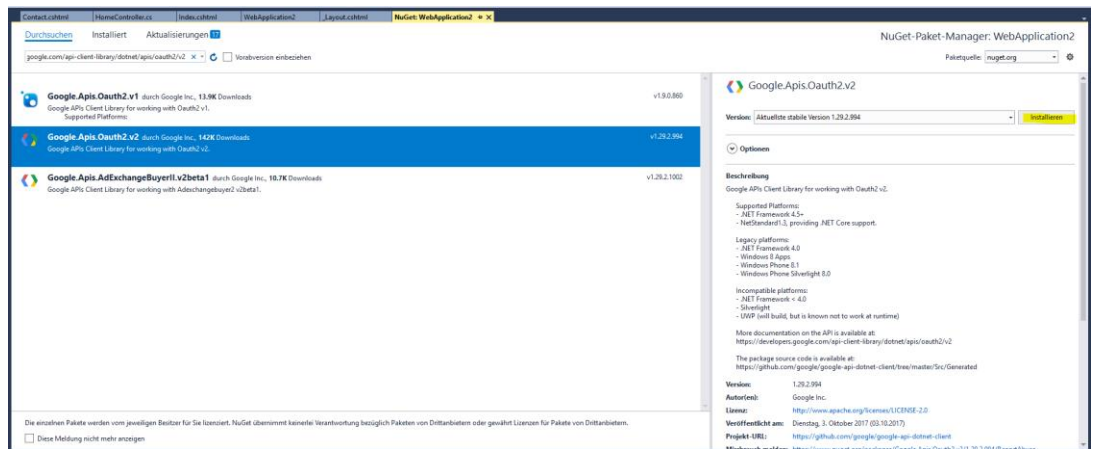
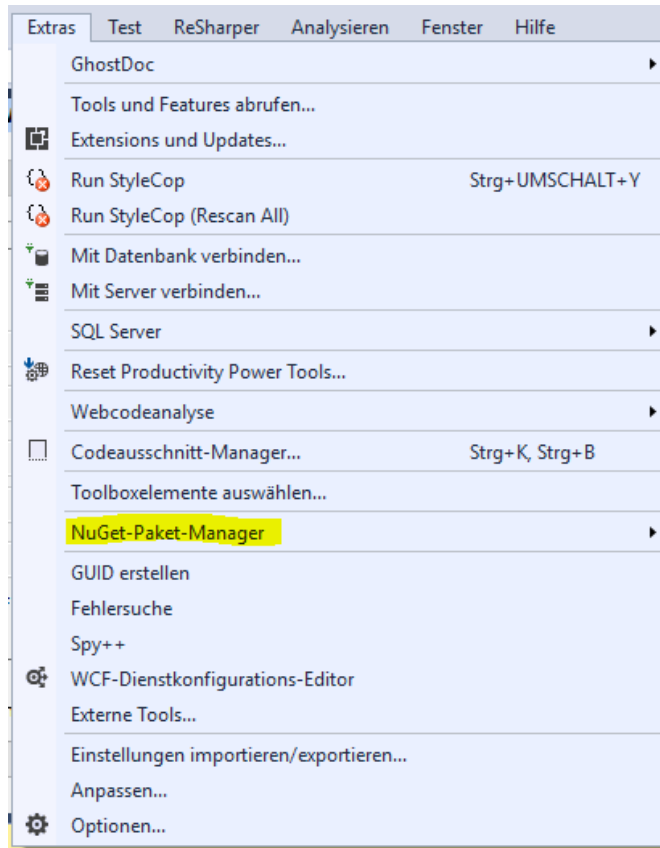
2. Erweitern des „eigenen Backends“, für die Überprüfung des Tokens, des Login-Status des Users bei den Google Accounts (Auth Server) und Generierung/Retournierung einer Session:

a. Neue HomeController – Funktion SSOTokenSignin



b. Erweitern der SSOTokenSignin-Funktion. Entweder via API Client Library **ODER** via Google-API Endpunkt (siehe weiter unten)

1. Verify id_token mittels API Client:
<https://developers.google.com/api-client-library/dotnet/apis/oauth2/v2>
2. NuGet Package installieren:



1. **Oder** verify id_token via Google-API-Endpunkt (ID_TOKEN wurde per AJAX gesendet)
https://www.googleapis.com/oauth2/v3/tokeninfo?id_token={ID_TOKEN}

```
WebApplication2 | HomeController.cs | index.cshtml | _Layout.cshtml
WebApplication2.Controllers.HomeController

31 return view();
32 }
33
34 public JsonResult SSOTokenSignin()
35 {
36     // Get ID_Token and Verify it Using a Library https://developers.google.com/api-client-library/dotnet/apis/oauth2/v2
37
38     // OR check Google Endpoint: https://www.googleapis.com/oauth2/v3/tokeninfo?id_token=XY7123
39     var id_token = Request["idtoken"];
40     var request = (HttpRequest)WebRequest.Create("https://www.googleapis.com/oauth2/v3/tokeninfo?id_token="+id_token);
41
42     var postData = "id_token=" + id_token;
43     var data = Encoding.ASCII.GetBytes(postData);
44
45     request.Method = "POST";
46     request.ContentType = "application/x-www-form-urlencoded";
47     request.ContentLength = data.Length;
48
49     using (var stream = request.GetRequestStream())
50     {
51         stream.Write(data, 0, data.Length);
52     }
53
54     var response = (HttpWebResponse)request.GetResponse();
55
56     var responseString = new StreamReader(response.GetResponseStream()).ReadToEnd();
57
58     // depending on the response: create session or not
59
60     // return the response of googleapis.com
61     return Json(responseString);
62 }
63
64 }
65 }
```

2. Response des Endpunktes prüfen und abhängig davon eine SESSION für den User geben. Mögliche Response des googleapis.com:

```
3. {
4.   // These six fields are included in all Google ID
   Tokens.
5.   "iss": "https://accounts.google.com",
6.   "sub": "110169484474386276334",
7.   "azp": "1008719970978-
   hb24n2dstb40o45d4feuo2ukqmcc6381.apps.googleusercontent.c
   om",
```

```
"aud": "1008719970978-
hb24n2dstb40o45d4feuo2ukqmcc6381.apps.googleusercontent.com",
```

```
8.   "iat": "1433978353",
9.   "exp": "1433981953",
10.
11.   // These seven fields are only included when the
   user has granted the "profile" and
12.   // "email" OAuth scopes to the application.
13.   "email": "testuser@gmail.com",
14.   "email_verified": "true",
```

```
15.      "name" : "Test User",
16.      "picture": "https://lh4.googleusercontent.com/-
      kYgzyAWpZzJ/ABCDEFghi/AAAJKLMNOP/tIXL9Ir44LE/s99-
      c/photo.jpg",
17.      "given_name": "Test",
18.      "family_name": "User",
19.      "locale": "en"
20.  }
```

Sign Out Button

<https://developers.google.com/identity/sign-in/web/sign-in>

1. Platzieren des Buttons im Template (Navigation-Bar)

```
<div class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      @Html.ActionLink("Anwendungsname", "Index", "Home", new { area = "" }, new { @c
    </div>
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Startseite", "Index", "Home")</li>
        <li>@Html.ActionLink("Info", "About", "Home")</li>
        <li>@Html.ActionLink("Kontakt", "Contact", "Home")</li>
      </ul>

      <ul class="nav navbar-nav pull-right">
        <li><a href="#" onclick="signOut();">Sign out</a></li>
      </ul>
    </div>
  </div>
</div>
```

2. Einfügen des Snippets im _Layout.cshtml

```
<script type="text/javascript">
  function signOut() {
    var auth2 = gapi.auth2.getAuthInstance();
    auth2.signOut().then(function () {
      console.log('User signed out.');
```