

ASP.NET MVC Labo Week 4

Doelstelling:

- Herhalings oefening modelbinding & presentation model
- ASP.NET Web API

Oefening 1: Herhalings oefening

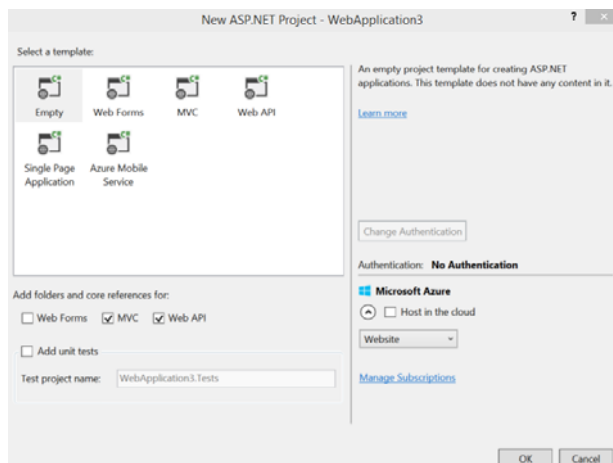
Deze oefening heeft als doel de basisconcepten van de eerste weken volledig te herhalen. Voor deze oefening is het de bedoeling dat u GEEN gebruik maakt van de voor gegenereerde templates zodat u de werking van modelbinding & HTML helpers goed onder de knie krijgt.

Opdracht

We schrijven een zoekertjes applicatie met volgende schermen:

Voor u start

Maak een nieuwe Empty ASP.NET MVC project aan, maar vink zowel MVC als Web API. Na het aanmaken van het project moet u de file Data.cs toevoegen aan de het ASP.NET project. U krijgt van mij ook alle models die u nodig heeft voor dit project. Voeg deze ook toe.



Overzicht zoekertjes

Voeg een controller “ZoekertjesController” toe aan het project. Deze controller zal om te beginnen één ActionMethod “Index” bevatten. Deze methode zal een overzicht van alle zoekertjes terugkeren. Dit is het scherm dat zichtbaar moet zijn als we de applicatie starten. Pas dit aan in RouteConfig.cs.

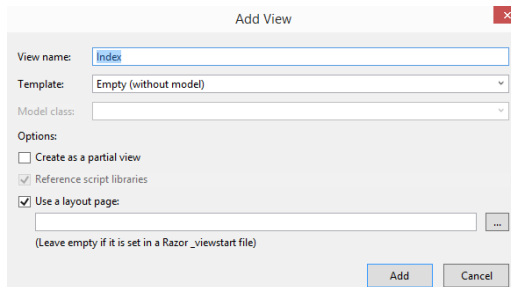
De methode Index in de Zoekertjes controller zal alle zoekertjes ophalen in Data.cs en deze doorsturen naar de view.

```

0 references
public ActionResult Index()
{
    List<Zoekertje> alleZoekertjes = new List<Zoekertje>();
    alleZoekertjes = Data.GetZoekertjes();
    return View(alleZoekertjes);
}

```

Voeg nu een “Empty” view toe (we schrijven vandaag alles zelf)



Open de view “Index.cshtml” en maak deze volledig leeg. Nu kunnen we starten met het schrijven van code in de view. Het eerste wat we moeten doen is opgeven wat voor soort data we wensen weer te geven in deze view. We doen dit door bovenaan het type van data te declareren. Opgepast: de naam van de namespace (Zoekertjes.WebApp.Models) kan bij u anders zijn !!!

```
@model IEnumerable<Zoekertjes.WebApp.Models.Zoekertje>
```

Indien u niet constant de volledige naam (namespace + model) wenst op te geven dan kan u de namespace toevoegen aan de web.config zodat de viewengine deze automatisch weet. Dit kan u doen door de web.config in de Views folder te openen en de namespace toe te voegen (zie onderstaande screenshot). Daarna herstart u best Visual Studio.

```

<system.web.webPages.razor>
  <host factoryType="System.Web.Mvc.MvcWebRazorHost" />
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="System.Web.Mvc" />
      <add namespace="System.Web.Mvc.Ajax" />
      <add namespace="System.Web.Mvc.Html" />
      <add namespace="System.Web.Routing" />
      <add namespace="Zoekertjes.WebApp" />
      <add namespace="Zoekertjes.WebApp.Models" />
    </namespaces>
  </pages>
</system.web.webPages.razor>

```

Na het herstarten kan u de code in de view als volgt aanpassen:

```
@model IEnumerable<Zoekertje>
```

Nu weet de viewengine van ASP.NET MVC het soort data waar we mee wensen te werken binnen die view. Wij moeten een lijst van data weergeven die er als volgt kan uitzien:

Application name

- iPad gezocht
- Oude wijn

© 2014 - My ASP.NET Application

We maken gebruik van een `` element met daarin `` elementen. Deze `` elementen bevatten links naar de detail pagina van het zoekertje. Deze html code genereren we door via `@foreach` de elementen in ons "Model" (van het type `IEnumerable<Zoekertje>`) te overlopen. Voor iedere element genereren we dan de juiste `` tag. We kunnen dit doen op twee manieren: via `@Html.ActionLink` of door zelf een `` te plaatsen. Onderstaande code maakt gebruik van de `@Html.ActionLink`. De eerste parameter is de tekst die moet verschijnen in de link. De tweede parameter is de actie op de server die we wensen aan te roepen. Als laatste parameter geven we het id van het zoekertje mee waarvoor we de details wensen op te vragen.

```
@foreach (Zoekertje zoekertje in Model)
{
    <li>@Html.ActionLink(zoekertje.Titel, "Details", new { Id = zoekertje.Id })</li>
}
```

Details Zoekertje

Het tweede scherm dat we wensen uit te werken is het detail scherm. Dit scherm zal alle info over ons zoekertje bevatten. Voeg een action method Details toe met als nullable parameter `int? id` en dit in de controller `ZoekertjesController`. Daarna kunnen we het juiste zoekertje ophalen.

We zitten echter met een probleem. Het zoekertje object bevat heel wat informatie maar nog niet alles. We hebben bv. Het categoried maar niet de omschrijving. Zelfde scenario voor locatie. We hebben het locatied in ons zoekertje, maar niet de omschrijving. Toch wensen we deze ook weer te geven. Dit kunnen we oplossen op twee manieren, via de `ViewBag` en via een `PresentationModel`. In dit voorbeeld maken we eens gebruik van de `ViewBag`. We halen beide objecten op en slaan deze op in de `ViewBag`. Deze kunnen we dan uitlezen in de view.

```
0 references
public ActionResult Details(int? id)
{
    if (!id.HasValue)
        return RedirectToAction("Index");

    Zoekertje zoekertje = Data.GetZoekertje(id.Value);

    Categorie categorie = Data.GetCategorie(zoekertje.CategorieId);
    ViewBag.Categorie = categorie.Naam;

    Locatie locatie = Data.GetLocatie(zoekertje.LocatieId);
    ViewBag.Locatie = locatie.Naam;

    return View(zoekertje);
}
```

Voeg nu een “Empty” view “Detail” toe zonder model en maak deze volledig leeg. Wat moet u nu bovenaan plaatsen in de view?

@model Zoekertje

Probeer nu zelf onderstaande layout te benaderen (moet zeker niet zelfde zijn, maar wel ALLE gegevens weergeven). Maak gebruik van volgende @Html helpers=

- @Html.DisplayFor(x => x.PropertyName)
 - o Print de waarde van de property af
- @Html.LabelFor(x => x.PropertyName)
 - o Zal waarde DisplayName tussen <label/> plaatsen
- @Html.Label(string tekst)
 - o Deze helper zal gewoon de tekst tussen <label/> tags plaatsen

Voor telefoon en email adres moet u zelf de teksten “Contacteren via telefoon....” en “Contacteren via e-mail.....” toevoegen in de view. Deze mogen echter ENKEL verschijnen als ContacteerViaTelefoon of ContacteerViaEMail true zijn. Voorzie onderaan ook een link om terug naar het overzicht te gaan.

Oude wijn

Wat zoek ik iemand nog een oude fles wijn liggen uit 1988

Max. prijs die ik wens te betalen € 10

Waar kan ik het zoekertje afhalen Brugge

Wat is de categorie van het zoekertje Eten & Drinken

Contacteren via telefoon mogelijk op het nr:0555454545

Contacteren via e-mail mogelijk op het adres:wijnrinker@wijn.be

[<< terug](#)

Toevoegen zoekertje

In dit scherm is het de bedoeling dat we nieuwe zoekertjes kunnen toevoegen. Voeg op het startscherm een link toe “Nieuw zoekertje”. Deze link zal verwijzen naar een actionmethode “New” in de controller ZoekertjesController.

Application name

- iPad gezocht
- Oude wijn

[Nieuwe zoekertje](#)

Maak nu zelfstandig het scherm waar je een zoekertje kan ingeven. Zorg ook dat validatie meldingen worden weergegeven. Dit scherm ziet er ongeveer als volgt uit:

Wat zoek ik

The Wat zoek ik field is required.

Korte omschrijving (max 100 kar.)

The Korte omschrijving (max 100 kar.) field is required.

Naam

The Naam field is required.

Telefoon

The Telefoon field is required.

E-Mail

The E-Mail field is required.

Max. prijs die ik wens te betalen

The Max. prijs die ik wens te betalen field is required.

Wat is de categorie van het zoekertje

Waar kan ik het zoekertje afhalen

Contacteer me via telefoon
☐



Contacteer me via e-mail
☐

Enkele tips:

- U maakt best gebruik van presentationmodels
- @Html.ValidationMessageFor(m => m.propertyname) zal voor foutmelding zorgen indien deze moet worden weergegeven.

Verwijderen van een zoekertje

Het laatste onderdeel is het verwijderen van een zoekertje. Wijzig het startscherm zodat dit er als volgt uit ziet:

-  iPad gezocht
-  Oude wijn

Nieuwe zoekertje

We voegen een icoon toe waarop we kunnen klikken. Daarna zal de vraag komen of we het product wensen te verwijderen.

Wenst u dit zoekertje te verwijderen?

Oude wijn

Iemand nog een oude fles wijn liggen uit 1988

Reden van verwijderen

- ☐ Reeds gevonden
- ☐ Geen interesse meer
- ☐ Aanbod is te duur

Delete [Terug](#)

Wanneer we iets willen verwijderen moeten we een reden opgeven. Zorg ervoor dat de gebruiker één reden kan aanvinken. De mogelijkheden zijn afkomstig uit Data.cs.

Extra:

- Wijzig de code in Details method en Details view zodat je gebruik maakt van "PresentationModel" en GEEN ViewBag

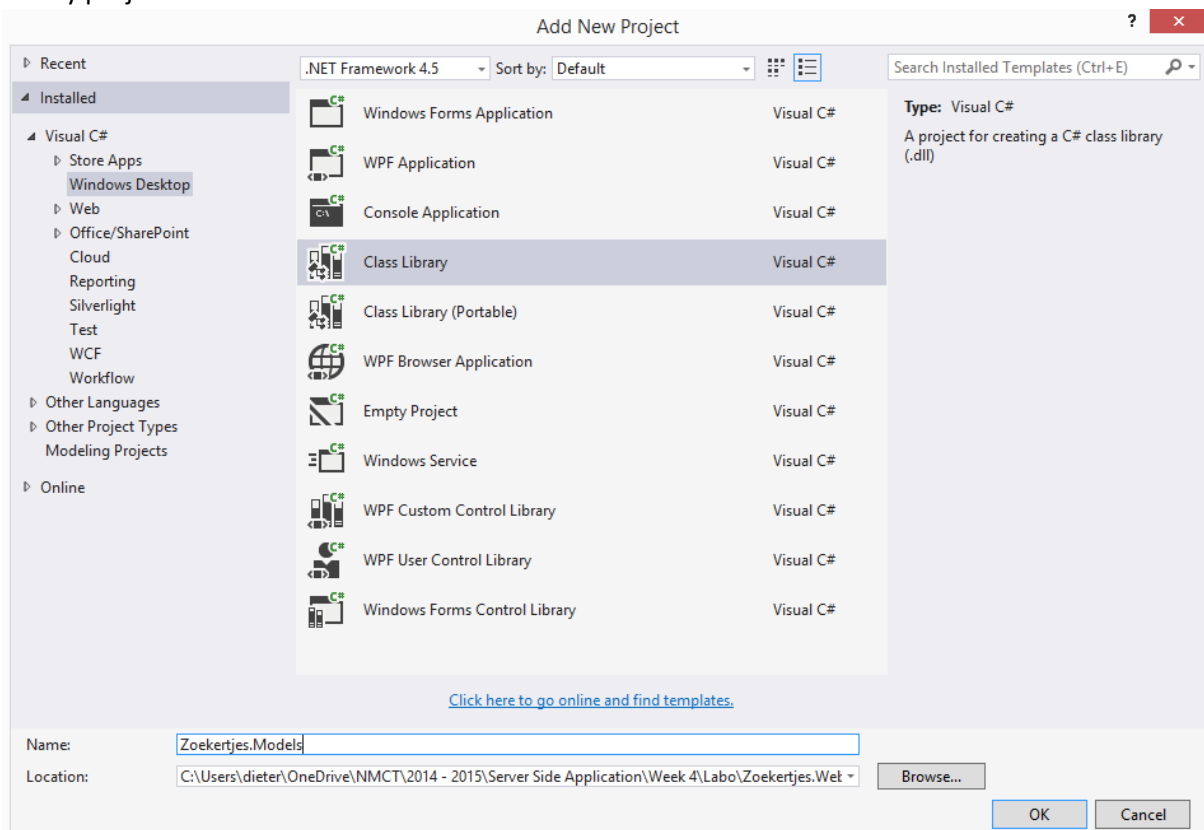
Oefening 2: Web API

We bouwen verder op de vorige oefening en gaan een eenvoudige API bouwen zodat mobile applicaties of desktop applicaties ook kunnen zoeken in de zoekertjes. Voeg nu zelf een WebAPI Controller toe waarmee je alle zoekertjes kan ophalen. Deze controller zal één ActionMethod Get() bevatten die de lijst zal terugkeren (wat is terugkeer type?). Test de API in de browser. In chrome zal je volgende terugkrijgen:

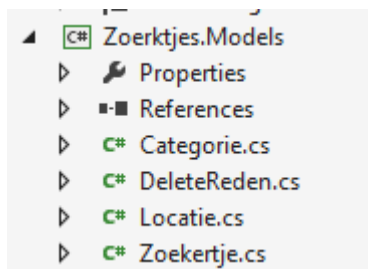
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<ArrayOfZoekertje xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/Zoekertjes.WebApp.Models">
  <Zoekertje>
    <CategoryId>1</CategoryId>
    <ContacteerViaEmail>true</ContacteerViaEmail>
    <ContacteerViaTelefoon>false</ContacteerViaTelefoon>
    <Email>dieter@howest.</Email>
    <Id>1</Id>
    <LocatieId>3</LocatieId>
    <Naam>Dieter De Preester</Naam>
    <Omschrijving>Ik ben opzoek naar een oude ipad</Omschrijving>
    <Prijs>120</Prijs>
    <Telefoon/>
    <Titel>iPad gezocht</Titel>
  </Zoekertje>
  <Zoekertje>
    <CategoryId>3</CategoryId>
    <ContacteerViaEmail>true</ContacteerViaEmail>
    <ContacteerViaTelefoon>true</ContacteerViaTelefoon>
    <Email>wijndrinker@wijn.be.</Email>
    <Id>2</Id>
    <LocatieId>4</LocatieId>
    <Naam>Steven Martnes</Naam>
    <Omschrijving>Iemand nog een oude fles wijn liggen uit 1988</Omschrijving>
    <Prijs>10</Prijs>
    <Telefoon>0555454545</Telefoon>
    <Titel>Oude wijn</Titel>
  </Zoekertje>
</ArrayOfZoekertje>
```

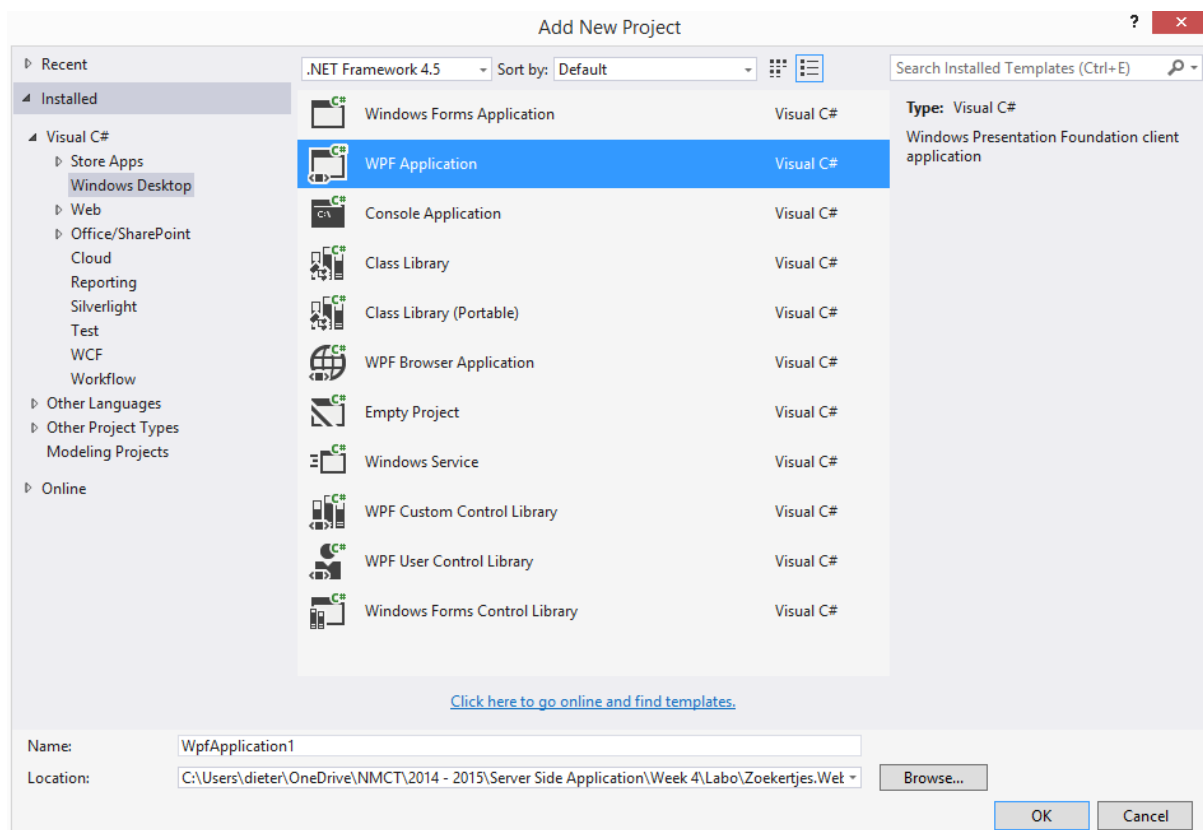
Indien de service goed werkt kunnen we starten met het client project. Eerst gaan we een nieuw project toevoegen waarin we onze models plaatsen. Dit is nodig aangezien we de models zowel wensten te gebruiken in onze desktop applicatie als ook in de webservice. Voeg een nieuw class library project toe aan de solution.



Sleep nu de models uit de webapplicatie in het nieuwe project. Daarna mag je deze verwijderen uit de webapplicatie. Probeer dit project te compileren. U zal normaal gezien foutmeldingen krijgen, enkele namespaces zullen niet gevonden worden. Probeer dit zelf op te lossen (u zou dit nu toch reeds moeten kunnen). Als het project compileert leg je een referentie naar dit project vanuit het webservice project zodat dit project ook terug compileert.

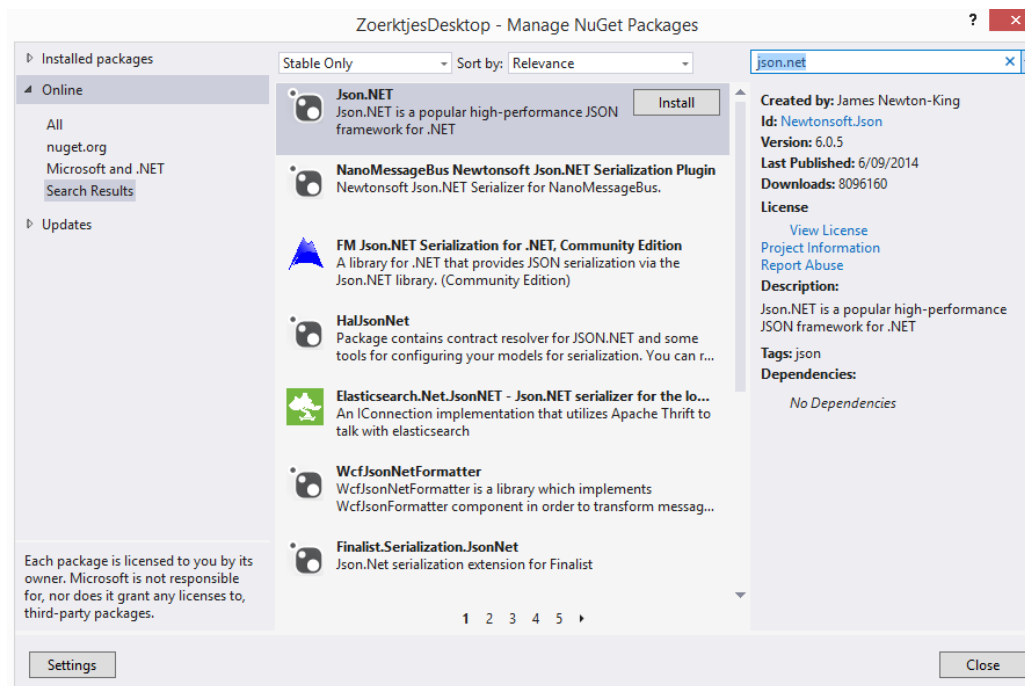
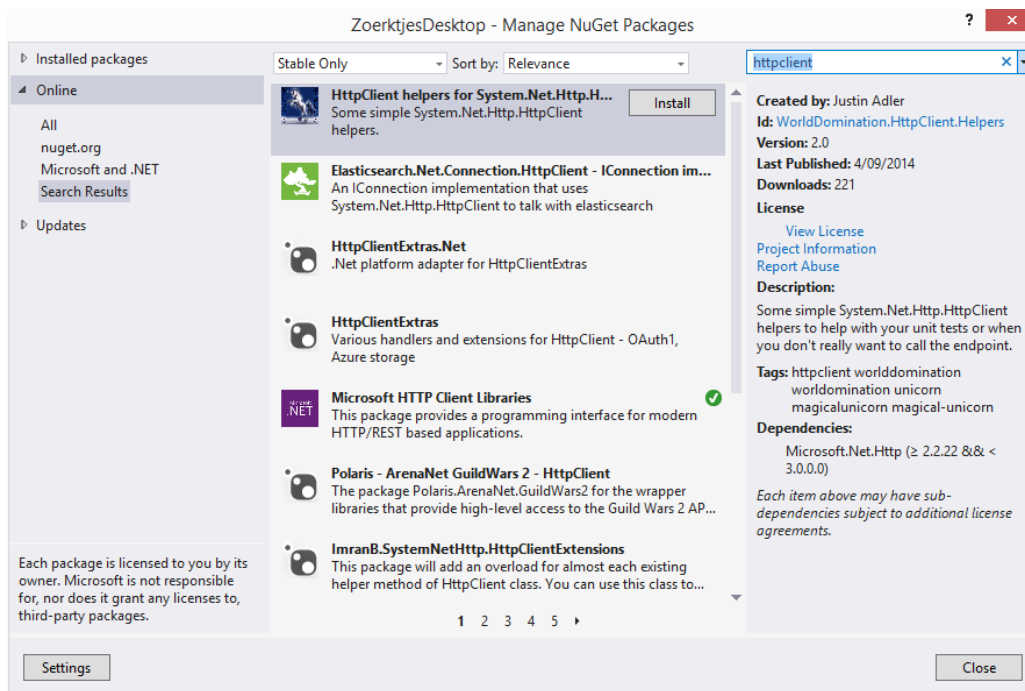


Maak daarna een WPF applicatie (indien u een onweerstaanbare drang zou voelen om een Windows Phone of Windows Store applicatie te maken dan mag dit ook) die deze webservice zal aanspreken en de lijst van zoekertjes zal weergeven in een listbox. Maak goed gebruik van de demo applicatie van de theorie les.



Leg eerst en vooral een referentie naar het models project zodat de models aanspreekbaar zijn in de WPF applicatie.

Om webservices te kunnen aanspreken moeten we gebruik maken van de HttpClient library. Voeg deze toe via Nuget. Daarnaast moeten we ook de JSON.Net library toevoegen.



Bekijk nu de code uit de demo applicatie en zoek zelf verder hoe je de juiste data kan ophalen.

