

# Workshop

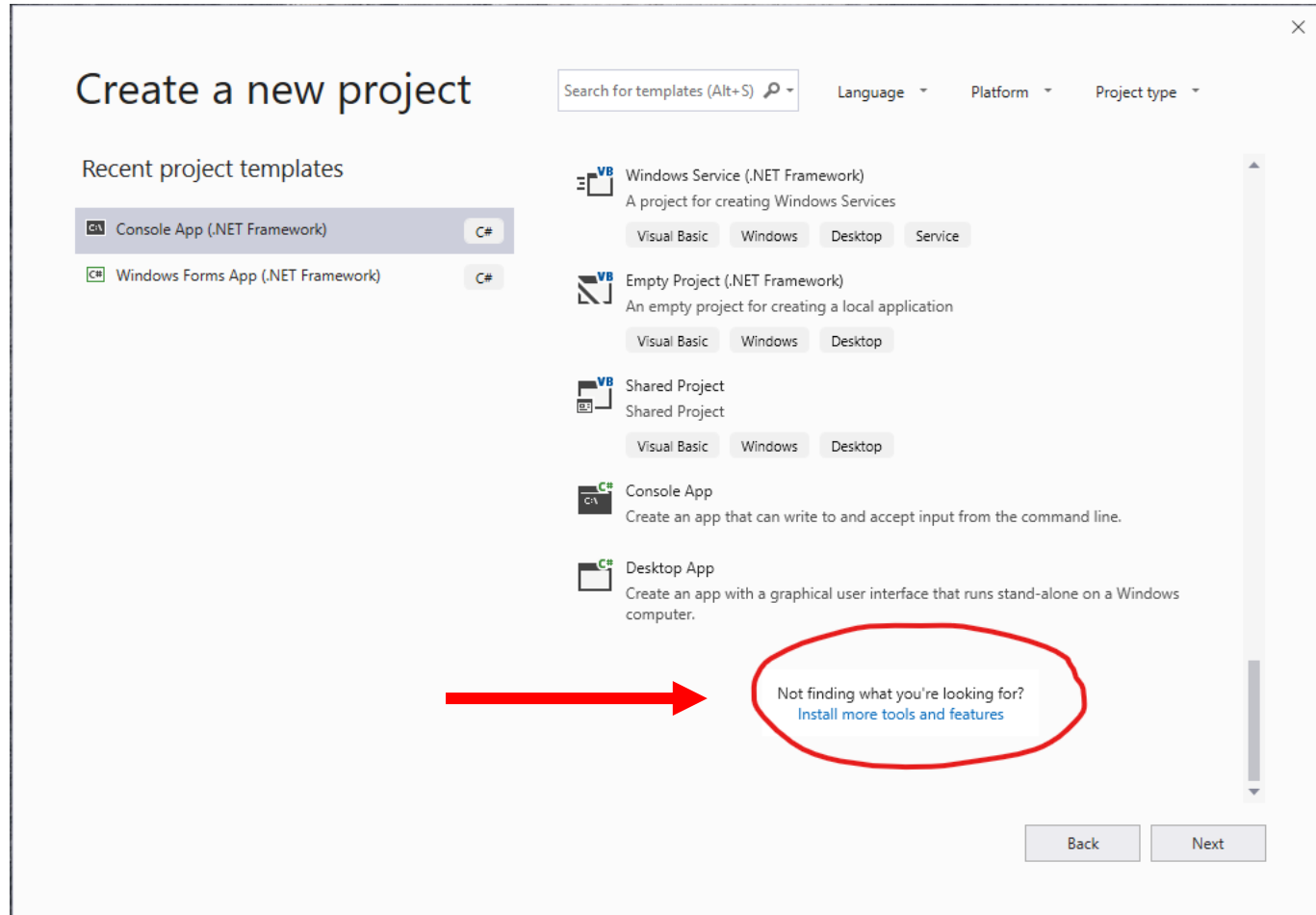
*“Maak je eigen Web API in C#”*

Joris van der Straten

Gebaseerd op <https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>

# Benodigdheden

Visual Studio 2017 / 2019 template



Modifying — Visual Studio Community 2019 — 16.2.5

Workloads Individual components Language packs Installation locations

Web & Cloud (4)

- ASP.NET and web development**  
Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.
- Python development**  
Editing, debugging, interactive development and source control for Python.
- Azure development**  
Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including...
- Node.js development**  
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Windows (3)

- .NET desktop development**  
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.
- Desktop development with C++**  
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.
- Universal Windows Platform development**  
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Location  
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

Total space required 2.99 GB

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

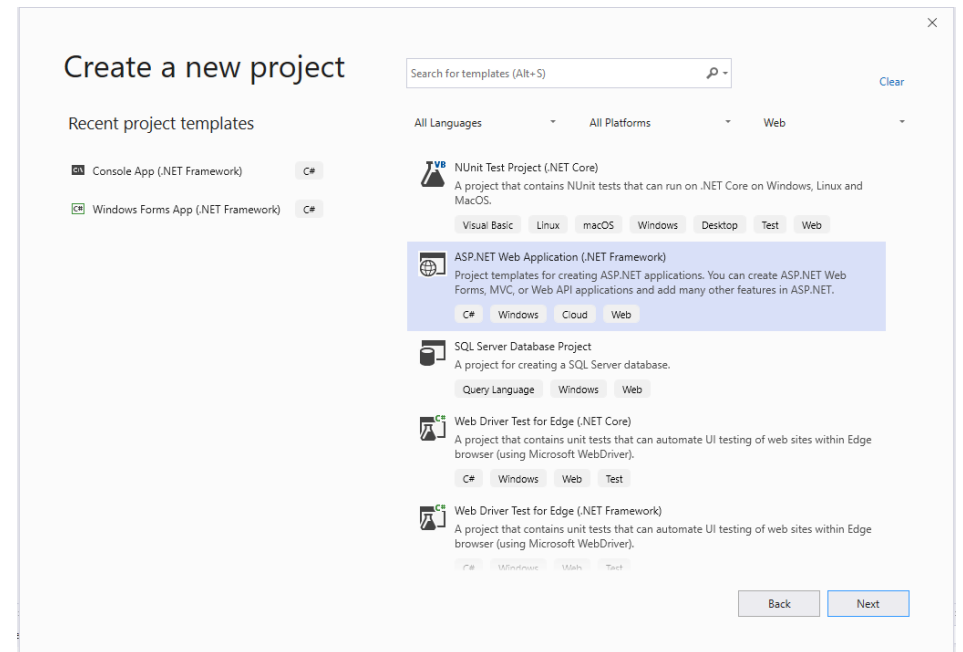
Install while downloading

**Installation details**

- > visual studio core editor
- > .NET desktop development \*
- ▼ **ASP.NET and web development \***
  - Included
    - ✓ .NET Core 2.1 development tools
    - ✓ .NET Framework 4.7.2 development tools
    - ✓ ASP.NET and web development tools
    - ✓ IntelliCode
  - Optional
    - ✓ .NET Framework 4 – 4.6 development tools
    - ✓ Cloud tools for web development
    - ✓ .NET profiling tools
    - ✓ Entity Framework 6 tools
    - ✓ Advanced ASP.NET features
    - ✓ Developer Analytics tools
    - ✓ Web Deploy
    - ✓ Live Share
    - Windows Communication Foundation
    - .NET Core 2.2 development tools
    - .NET Framework 4.6.1 development tools

# Let's get started!

Maak een *ASP.NET Web Application* project aan



## Create a new ASP.NET Web Application



### Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.



### Web Forms

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.



### MVC

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.



### Web API

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.



### Single Page Application

A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

### Authentication

No Authentication

[Change](#)

### Add folders & core references

- Web Forms
- MVC
- Web API

### Advanced

- Configure for HTTPS
- Docker support  
(Requires [Docker Desktop](#))
- Also create a project for unit tests

MyWebApi.Tests

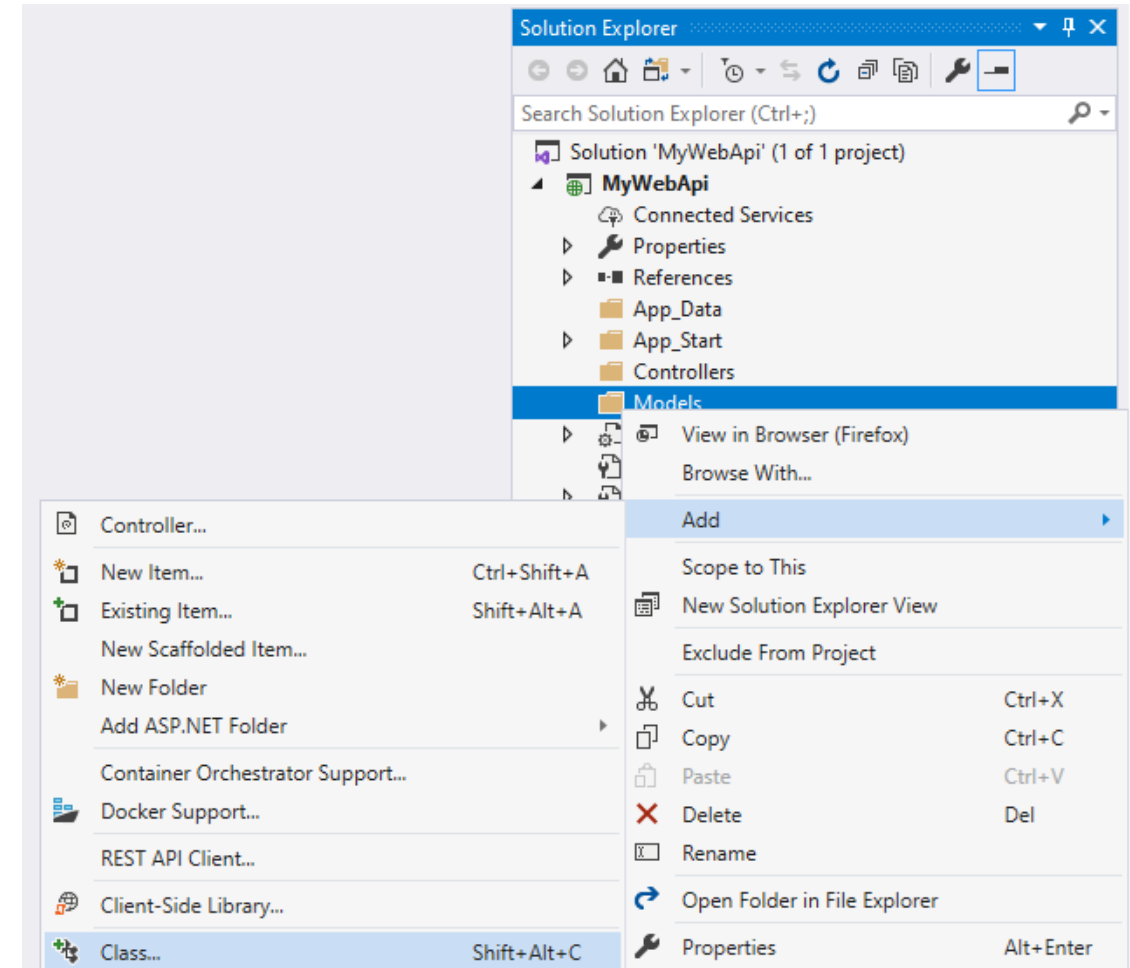
Back

Create

# Een *Model* toevoegen

Een *model* is een object dat de gegevens van je applicatie beschrijft.

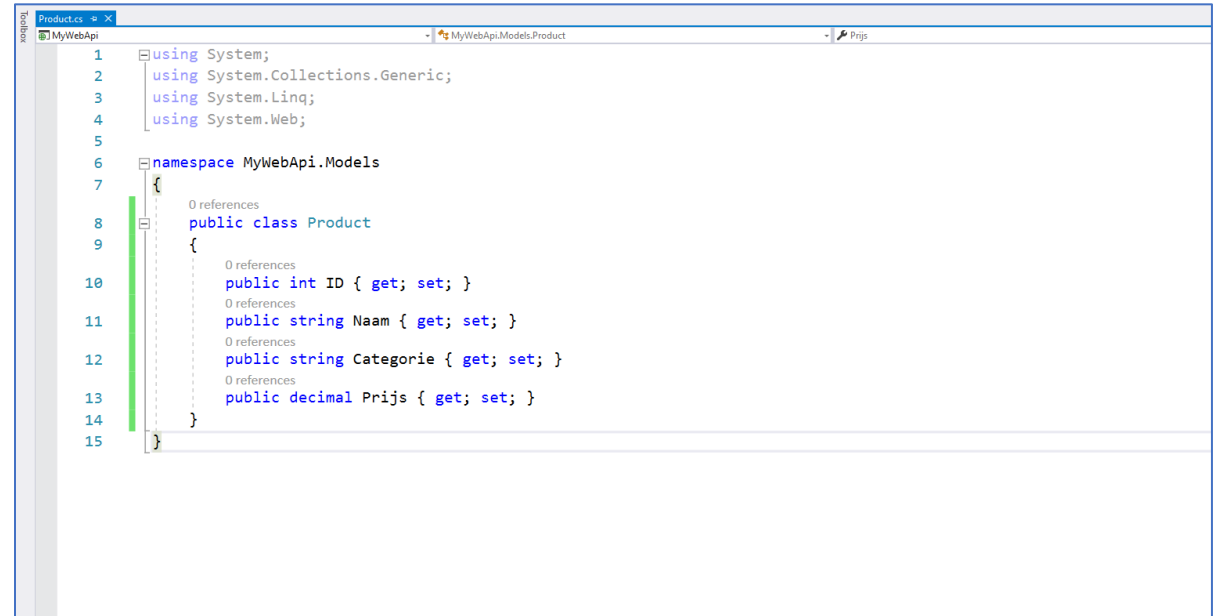
- Klik met *rechtermuisknop* op de folder *Models* en kies achtereenvolgens op *Add* -> *Class*
- Noem het bestand *Product.cs*



# Een *Model* toevoegen

Voeg de volgende *properties* (eigenschappen) toe aan je *Product* klasse:

```
public class Product
{
    public int ID { get; set; }
    public string Naam { get; set; }
    public string Categorie { get; set; }
    public decimal Prijs { get; set; }
}
```



The screenshot shows a Visual Studio code editor window with the following C# code:

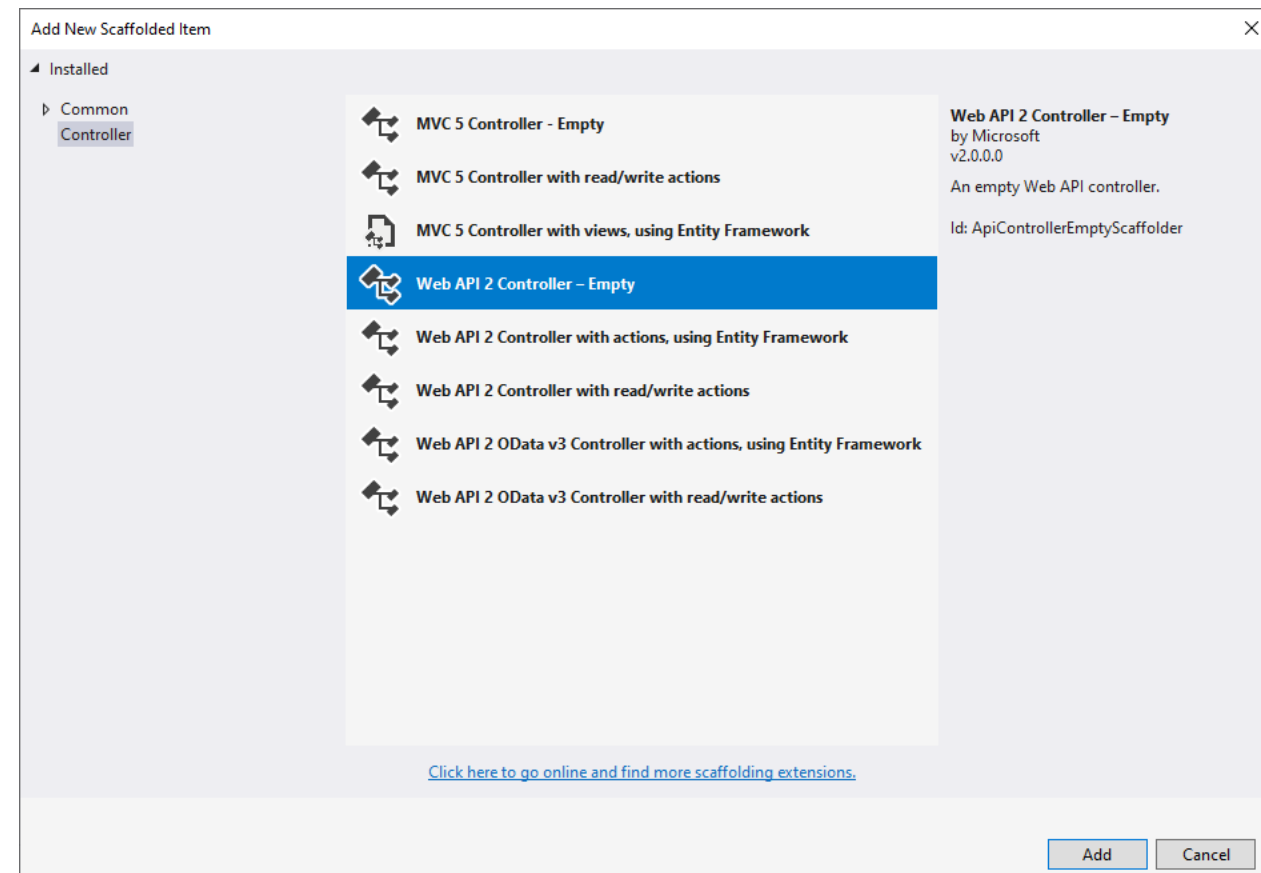
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5
6 namespace MyWebApi.Models
7 {
8     public class Product
9     {
10         public int ID { get; set; }
11         public string Naam { get; set; }
12         public string Categorie { get; set; }
13         public decimal Prijs { get; set; }
14     }
15 }
```



# Een *Controller* toevoegen

Binnen ASP.net Web API is een *controller* een object dat HTTP aanvragen verwerkt.

- Klik met *rechtermuisknop* op de folder *Controller* en kies achtereenvolgens op *Add* -> *Controller...*
- Kies voor *Web API 2 Controller – Empty*
- Pas de naam aan van *DefaultController* naar *ProductsController*



# Een *Controller* toevoegen

Verander de code naar het volgende:

```
public class ProductsController : ApiController
{
    private List<Product> products = new List<Product>()
    {
        new Product { ID = 1, Naam = "Samsung Galaxy A50", Merk = "Samsung", Prijs = 261 },
        new Product { ID = 2, Naam = "Apple iPhone 11 64GB", Merk = "Apple", Prijs = 794.99M }
    };

    public IEnumerable<Product> GetAllProducts() { return products; }

    public IHttpActionResult GetProduct(int id)
    {
        var product = products.FirstOrDefault((p) => p.ID == id);
        return product == null ? NotFound() : (IHttpActionResult)Ok(product);
    }
}
```

```
0 references
public class ProductsController : ApiController
{
    private List<Product> products = new List<Product>()
    {
        new Product { ID = 1, Naam = "Samsung Galaxy A50", Merk = "Samsung", Prijs = 261 },
        new Product { ID = 2, Naam = "Apple iPhone 11 64GB", Merk = "Apple", Prijs = 794.99M }
    };

    0 references
    public IEnumerable<Product> GetAllProducts() { return products; }

    0 references
    public IHttpActionResult GetProduct(int id)
    {
        var product = products.FirstOrDefault((p) => p.ID == id);
        return product == null ? NotFound() : (IHttpActionResult)Ok(product);
    }
}
```

# Gebruik van de Web API

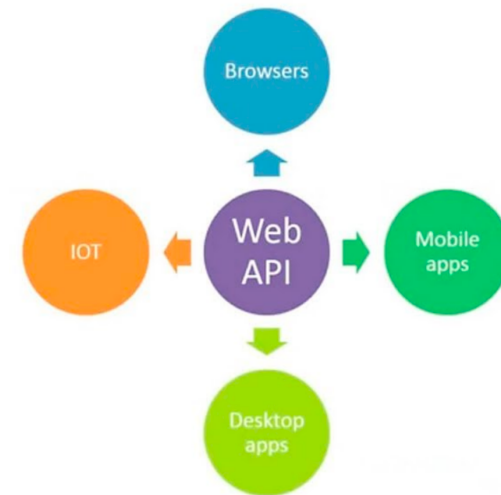
We kunnen nu ons project starten en browsen\* naar:

<https://localhost:44323/api/products>

<https://localhost:44323/api/products/1>

\* Let op! Het poortnummer kan verschillen!

Controller Methode	URI
GetAllProducts	/api/products
GetProduct	/api/products/id



# Gebruik van de Web API

Voorbeeld gebruik met HTML:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Products App</title>
</head>
<body>

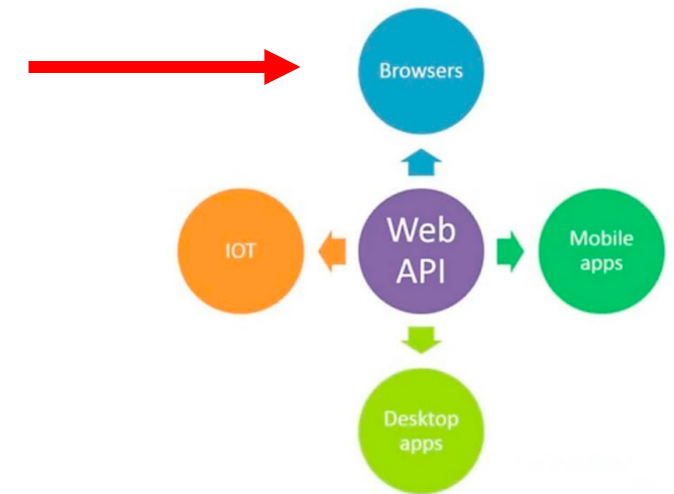
  <div>
    <h2>Alle Producten</h2>
    <ul id="products" />
  </div>
  <div>
    <h2>Search by ID</h2>
    <input type="text" id="prodId" size="5" />
    <input type="button" value="Search" onclick="find();" />
    <p id="product" />
  </div>

  <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-2.0.3.min.js"></script>
  <script>
    var uri = 'api/products';

    $(document).ready(function () {
      // Send an AJAX request
      $.getJSON(uri)
        .done(function (data) {
          // On success, 'data' contains a list of products.
          $.each(data, function (key, item) {
            // Add a list item for the product.
            $('<li>', { text: formatItem(item) }).appendTo($('#products'));
          });
        });

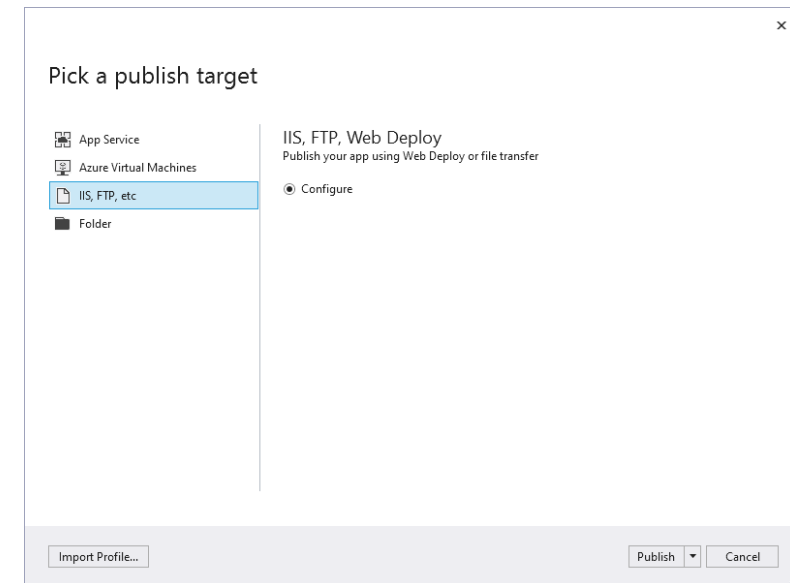
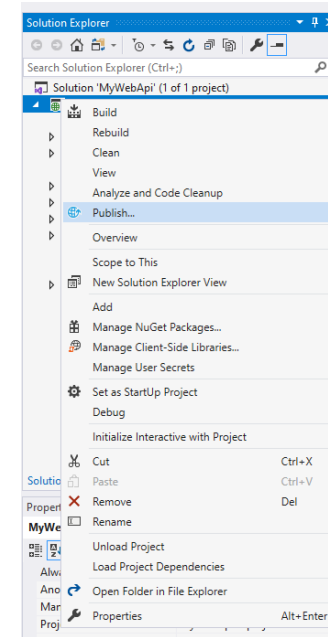
      function formatItem(item) {
        return item.Naam + ': ' + item.Prijs;
      }

      function find() {
        var id = $('#prodId').val();
        $.getJSON(uri + '/' + id)
          .done(function (data) {
            $('#product').text(formatItem(data));
          })
          .fail(function (jqXHR, textStatus, err) {
            $('#product').text('Error: ' + err);
          });
      }
    });
  </script>
</body>
</html>
```



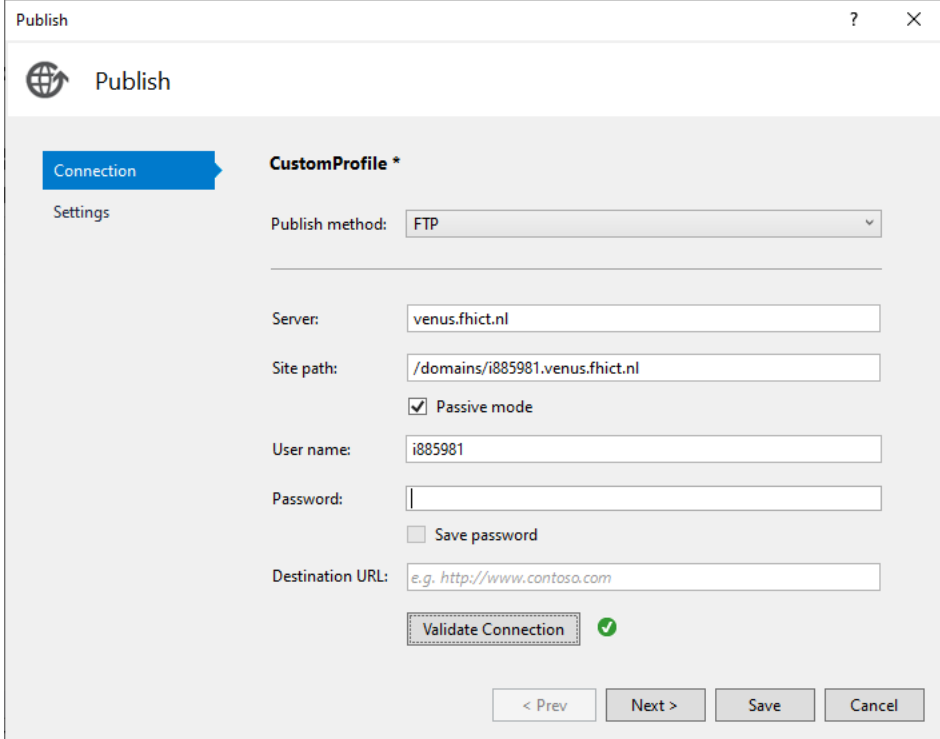
# Publiceren van de Web API

- Klik met *rechtermuisknop* op het project en vervolgens *Publish...*
- Kies voor optie *IIS, FTP, etc* en klik op *Publish* rechtsonderin



# Publiceren van de Web API

- Bij *Publish method* kies voor *FTP*
- Server is *venus.fhict.nl*
- Vul de *Site path* in
- *Passive mode* dient aangevinkt te zijn als je op school publisht
- Klik op *Validate Connection* om te testen



The screenshot shows the 'Publish' dialog box with the following settings:

- Publish method:** FTP
- Server:** venus.fhict.nl
- Site path:** /domains/i885981.venus.fhict.nl
- Passive mode:**
- User name:** i885981
- Password:** (empty)
- Save password:**
- Destination URL:** e.g. http://www.contoso.com

At the bottom, there is a 'Validate Connection' button with a green checkmark, and navigation buttons: '< Prev', 'Next >', 'Save', and 'Cancel'.