



# Blazor Server

## C# Web 2

### DE HOGESCHOOL MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt  
[www.pxl.be](http://www.pxl.be) - [www.pxl.be/facebook](http://www.pxl.be/facebook)



# Doelstellingen

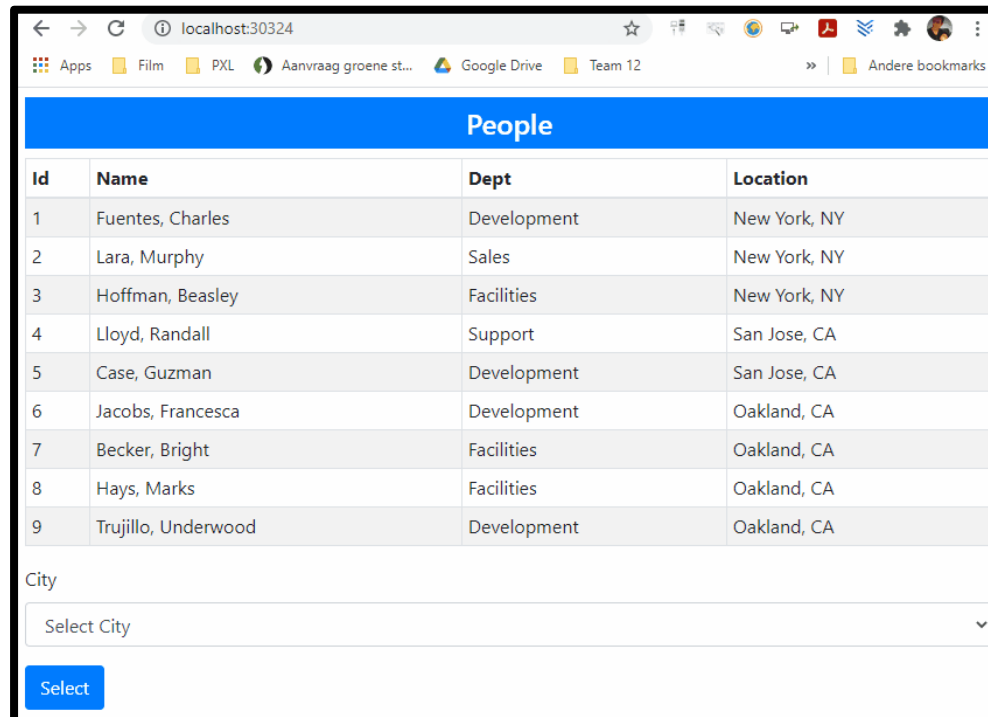
- Project - Blazor instellingen
- MVC View – Blazor razor component

# Startproject

- Visual Studio – **People-WebService (PeopleAppBlazorStartProject)**
- Github Classroom :
  - <https://classroom.github.com/a/CSnhf10k>

# Startproject

- Visual Studio – **People-WebService (PeopleAppBlazorStartProject)**
  - ASP .Net Core MVC applicatie voor een groot bedrijf met meerdere departementen
    - Toont de mensen in het bedrijf
      - Naam
      - Departement
      - Locatie (stad)
    - Na het selecteren van een stad worden de mensen met die locatie gemarkeerd



| People |                     |             |              |
|--------|---------------------|-------------|--------------|
| Id     | Name                | Dept        | Location     |
| 1      | Fuentes, Charles    | Development | New York, NY |
| 2      | Lara, Murphy        | Sales       | New York, NY |
| 3      | Hoffman, Beasley    | Facilities  | New York, NY |
| 4      | Lloyd, Randall      | Support     | San Jose, CA |
| 5      | Case, Guzman        | Development | San Jose, CA |
| 6      | Jacobs, Francesca   | Development | Oakland, CA  |
| 7      | Becker, Bright      | Facilities  | Oakland, CA  |
| 8      | Hays, Marks         | Facilities  | Oakland, CA  |
| 9      | Trujillo, Underwood | Development | Oakland, CA  |

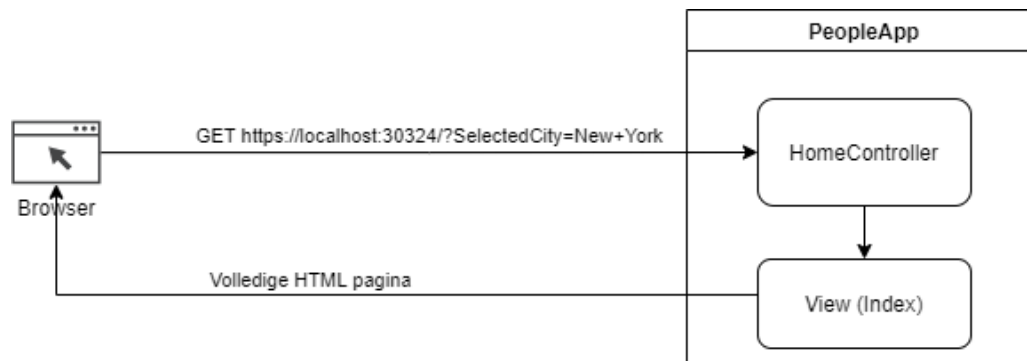
City

Select City

Select

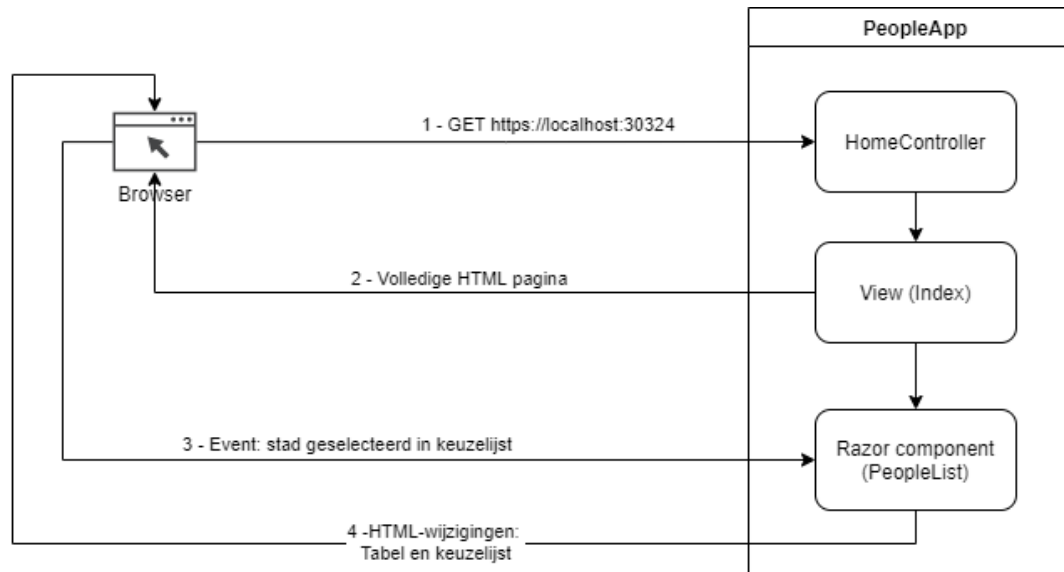
# Startproject

- Klik op *select* knop = browser stuurt HTTP GET request naar applicatie  
(bv. <https://localhost:30324/?SelectedCity=New+York>)
  - Index actie in *HomeController* maakt model en geeft deze door aan *Index.cshtml* View.
  - De View wordt door de razor view engine omgezet naar html (de rijen voor de personen uit de geselecteerde stad krijgen een specifieke css class).
  - **De volledige html pagina wordt dus bij elke klik opnieuw naar de browser gestuurd**
- ⇒ Veel overbodige (dubbele) data wordt over het netwerk gestuurd



# Blazor

- Met **Blazor** (server) wordt het mogelijk om **enkel de data (html) die wijzigt over het netwerk te sturen**
  - Als er een locatie (stad) geselecteerd wordt, dan wordt enkel de tabel en de keuzelijst aangepast
    - Blazor doet dit door m.b.v. javascript een persistente connectie open te houden met de applicatie op de server. Over deze connectie worden events gestuurd waardoor de applicatie zal antwoorden met de html (elementen, geen volledige pagina) die aangepast moet worden. M.b.v. javascript wordt de html in het browservenster aangepast.
- **Voordelen**
  - Je kan vermijden dat een volledige html pagina geladen wordt bij een actie van een gebruiker
  - Je kan blijven werken met C# en moet dus niet werken met javascript frameworks zoals Angular, React, ...
- **Nadelen**
  - Permanente connectie nodig tussen browser en Blazor applicatie. Geen offline functionaliteiten mogelijk



# Blazor – Program.cs

- Services

```
builder.Services.AddServerSideBlazor();
```

- Middleware

```
app.MapBlazorHub();
```

# Program.cs - Getting started

```
builder.Services.AddDbContext<DataContext>(opts =>
{
    ...
});
builder.Services.AddControllersWithViews();
builder.Services.AddServerSideBlazor();
builder.Services.AddScoped<IPersonRepository, PersonDbRepository>();
...
app.UseStaticFiles();
app.UseRouting();

app.MapDefaultControllerRoute();
app.MapBlazorHub();

SeedData.SeedDatabase(app);
app.Run();
```



# Blazor - Getting started

- Views/Shared/\_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>@ViewBag.Title</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <base href="~/ " />
</head>
<body>
  <div class="m-2">
    @RenderBody()
  </div>
  <script src="_framework/blazor.server.js"></script>
</body>
</html>
```

# Blazor - Getting started

We maken gebruik van de location repository service

```
public interface ILocationRepository
{
    IEnumerable<Location> GetLocations();
}

public class LocationRepository : ILocationRepository
{
    private readonly DataContext _context;

    public LocationRepository(DataContext context)
    {
        _context = context;
    }

    public IEnumerable<Location> GetLocations()
    {
        return _context.Locations.ToList();
    }
}
```

Deze service is gestart in onze program.cs class

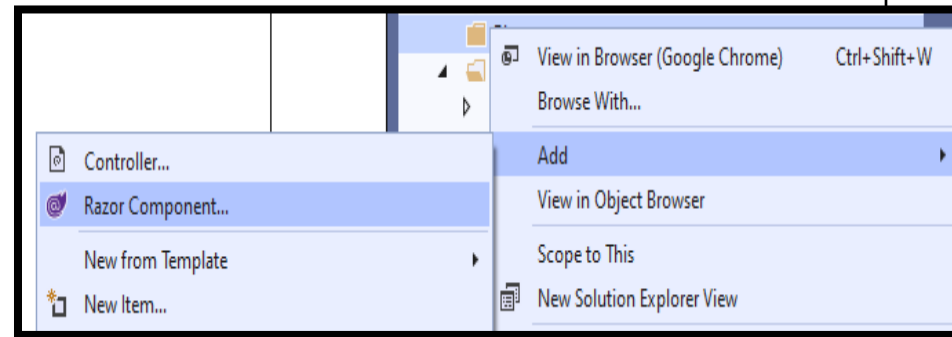
```
services.AddScoped<ILocationRepository, LocationDbRepository>();
```

# Blazor - \_Imports.razor

Voeg een bestand **\_Imports.razor** toe in de project folder!

Geef het bestand de volgende inhoud:

```
@using Microsoft.AspNetCore.Components
@using Microsoft.AspNetCore.Components.Forms
@using Microsoft.AspNetCore.Components.Routing
@using Microsoft.AspNetCore.Components.Web
@using Microsoft.JSInterop
@using PeopleApp.Models
@using PeopleApp.Data
@using PeopleApp.Services
```



Zonder de eerste 5 usings zal Blazor niet werken.

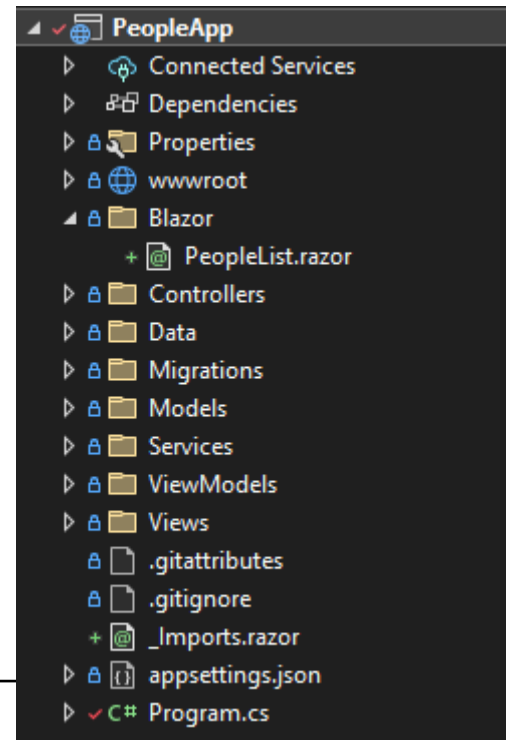
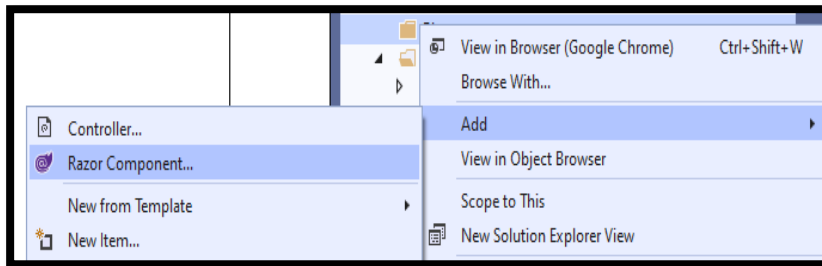
De laatste usings zijn voor ons gemak.

We zullen deze namespaces namelijk gebruiken in onze Razor componenten.

# Blazor - Razor component

Componenten in Blazor worden **Razor componenten** genoemd (niet te verwarren met Razor pages).

- Voeg een folder *Blazor* toe aan het project
- Voeg een Razor Component toe aan deze folder met de naam *PeopleList.razor*



# Blazor – PeopleList.razor

```
@code {  
  
    [Inject]  
    public IPersonRepository PersonRepository { get; set; }  
    [Inject]  
    public ILocationRepository LocationRepository { get; set; }  
    public IEnumerable<Person> People => PersonRepository.GetAll();  
    public IEnumerable<Location> Locations => LocationRepository.GetLocations();  
    public string SelectedCity { get; set; }  
    public string GetCssClass(string city)  
    {  
        return SelectedCity == city ? "bg-info text-white" : "";  
    }  
}
```

- Door het *[Inject]* attribuut krijgen de properties *PersonRepository* en *LocationRepository* een instantie toegekend door het dependency injection systeem.
- De *People* property haalt alle personen op en de *Locations* property haalt alle locaties op
- De *SelectedCity* property gaat bijhouden welke stad er geselecteerd is
- De *GetCssClass* methode gaat gebruikt worden om een rij in de table van personen een css class toe te kennen als de locatie van de person overeenkomt met de geselecteerde stad.

# Blazor – PeopleList.razor

- Door het *[Inject]* attribuut krijgen de properties *PersonRepository* en *LocationRepository* een instantie toegekend via dependency injection.
- De *People* property haalt alle personen op en de *Locations* property haalt alle locaties op
- De *SelectedCity* property gaat bijhouden welke stad er geselecteerd is
- De *GetCssClass* methode gaat gebruikt worden om een rij in de table van personen een css class toe te kennen als de locatie van de person overeenkomt met de geselecteerde stad.

# Blazor – PeopleList.razor

- Voeg de onderstaande html toe in *PeopleList.razor* boven de code sectie:

```
<table class="table table-sm table-bordered table-striped">
  <thead>
    <tr>
      <th>Id</th><th>Name</th><th>Dept</th><th>Location</th>
    </tr>
  </thead>
  <tbody>
    @foreach (Person p in People)
    {
      <tr class="@GetCssClass(p.Location.City)">
        <td>@p.Id</td>
        <td>@p.Surname, @p.Firstname</td>
        <td>@p.Department.Name</td>
        <td>@p.Location.City, @p.Location.State</td>
      </tr>
    }
  </tbody>
</table>

<div class="form-group">
  <label for="city">City</label>
  <select name="city" class="form-control" @bind="SelectedCity">
    <option disabled selected>Select City</option>
    @foreach (Location location in Locations)
    {
      <option value="@location.City" selected="@ (location.City == SelectedCity)">
        @location.City
      </option>
    }
  </select>
</div>
```

# Razor component aanmaken

Merk op in *PeopleList.razor*:

- Net zoals bij een *Razor View* kan je in een *Razor Component* html mengen met C# code.
- De properties en methodes gedefiniëerd in de code sectie kan je aanspreken in de html
- Met het *@bind* attribuut wordt de geselecteerde stad in de keuzelijst verbonden met de *SelectedCity* property.
  - Als de gebruiker een item selecteert in de keuzelijst, dan zal de *SelectedCity* property de overeenkomende waarde krijgen.
  - Als de *SelectedCity* property een waarde krijgt, dan zal de geselecteerde optie in de keuzelijst ook gezet worden.



# Views/Home -BlazorPeopleList

## HomeController – Action

```
public IActionResult Index([FromQuery] string selectedCity)
{
    ...
}
public IActionResult BlazorPeopleList()
{
    return View();
}
```

## Views/Home - BlazorPeopleList

```
<h4 class="bg-primary text-white text-center p-2">People</h4>
<hr />
<component type="typeof(PeopleApp.Blazor.PeopleList)" render-mode="Server"/>
```

# Home/Index View

Pas de *Index* view voor de *HomeController* aan zodat deze gebruik maakt van de *PeopleList* Razor component:

```
@using PeopleApp.Data
@model PeopleApp.ViewModels.PeopleListViewModel

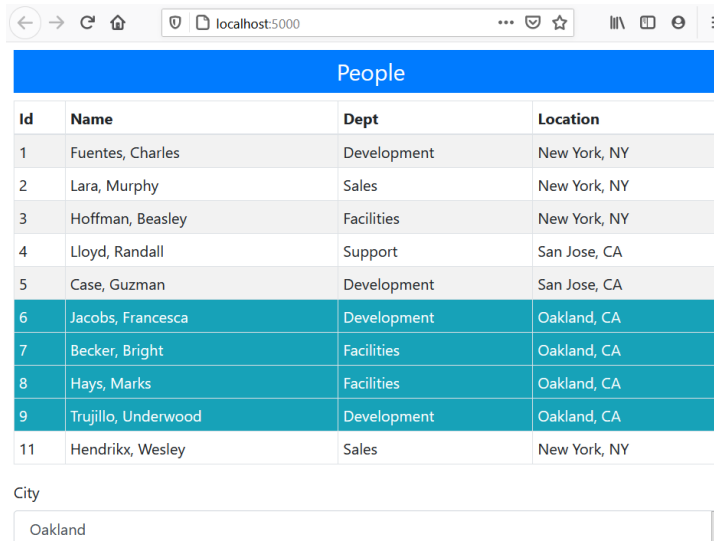
<h4 class="bg-primary text-white text-center p-2">People</h4>
<br />

<a asp-controller="Home" asp-action="BlazorPeopleList">Blazor People List</a>

<br />
<table class="table table-sm table-bordered table-striped">
```

# BlazorPeopleList

- Door een stad te selecteren gaat vanuit de browser aan de server applicatie gevraagd worden om de html (enkel de tabel en de keuzelijst) opnieuw te genereren.
- Aan server zijde wordt de html gemaakt door de Razor component en terug naar de browser gestuurd.
- In de browser word de html code vervangen (d.m.v. javascript)
  - We hebben succesvol vermeden dat de hele html pagina opnieuw geladen wordt



| People |                     |             |              |
|--------|---------------------|-------------|--------------|
| Id     | Name                | Dept        | Location     |
| 1      | Fuentes, Charles    | Development | New York, NY |
| 2      | Lara, Murphy        | Sales       | New York, NY |
| 3      | Hoffman, Beasley    | Facilities  | New York, NY |
| 4      | Lloyd, Randall      | Support     | San Jose, CA |
| 5      | Case, Guzman        | Development | San Jose, CA |
| 6      | Jacobs, Francesca   | Development | Oakland, CA  |
| 7      | Becker, Bright      | Facilities  | Oakland, CA  |
| 8      | Hays, Marks         | Facilities  | Oakland, CA  |
| 9      | Trujillo, Underwood | Development | Oakland, CA  |
| 11     | Hendrixx, Wesley    | Sales       | New York, NY |

City

Oakland

# Oefening

- **Blazor component**
  - Voeg de LocationList component toe in de solution
  - Voeg de DepartmentDetail component toe in de solution