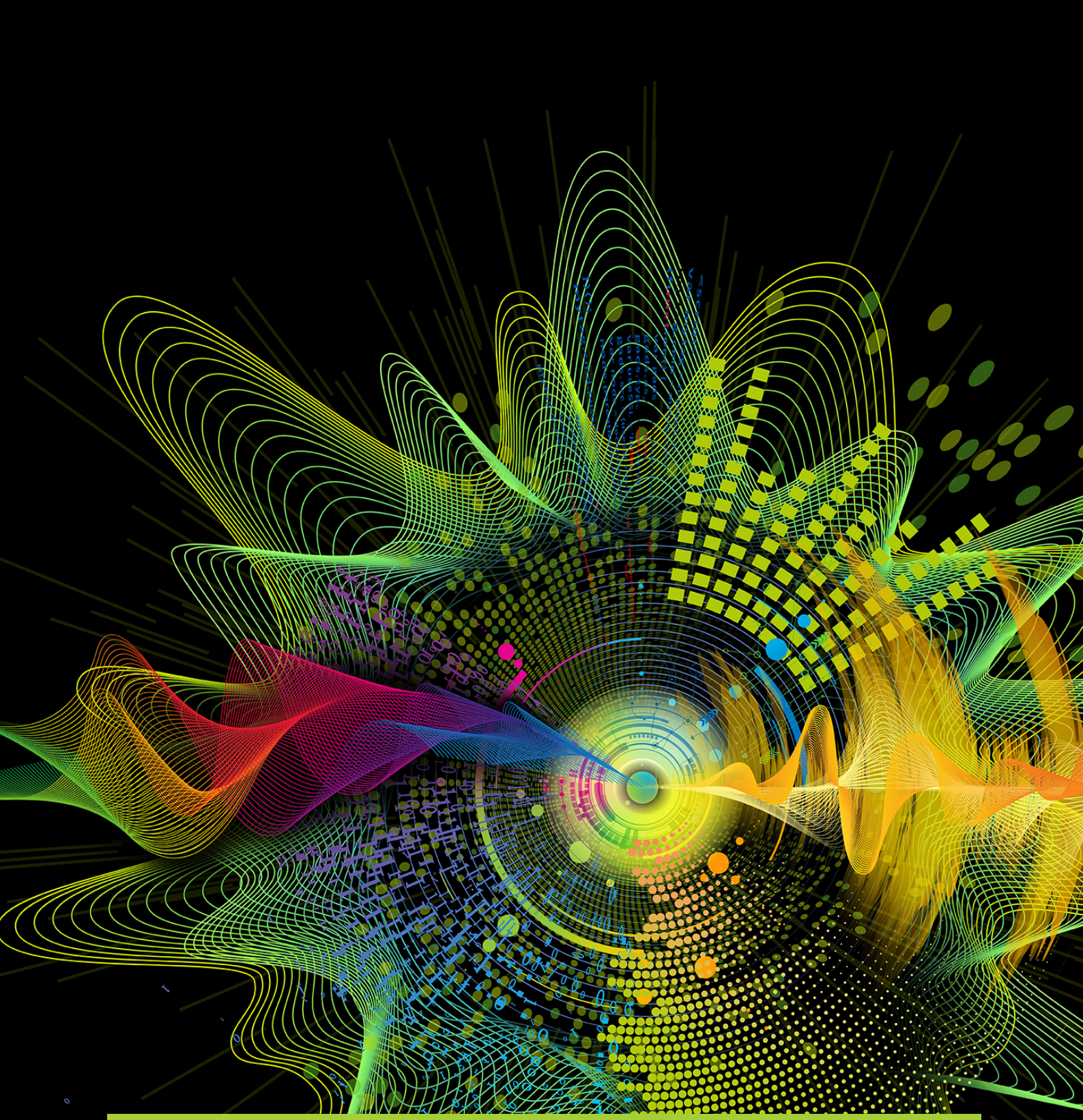


Programmeren

Module 9: C# 2

Leerjaar:	1
Crebonummer:	25604
Sector:	Software developer

INTRODUCTIE

In de vorige module over de programmeertaal c# hebben we kennis gemaakt met deze taal aan de hand van het maken van console applicaties. Met C# kunnen nog veel meer soorten applicaties gemaakt worden, waaronder ook webapplicaties. Deze module richt zich specifiek op deze webapplicaties.

Als we een webapplicatie in C# maken spreken we van een ASP.net (spreek uit als: ASP dotnet) applicatie. Deze naam is afkomstig uit de tijd van de allereerste versies van Microsofts webapplicaties. Deze heette destijds ASP-applicaties en stammen nog uit de tijd dat C# nog niet ontwikkeld was. De taal C# draait op het .NET (spreek uit als: dotnet) framework. Vanaf het moment dat C# gebruikt ging worden voor het maken van webapplicaties is de naam veranderd in ASP.net.

De allernieuwste versie van Microsofts webapplicaties heet officieel ASP.net Core. Deze laatste versie is een grote aanpassing, waarmee deze versie open-source en cross-platform geworden. Vanaf deze versie kan je ASP.net applicaties maken en hosten op het Windows, MacOS of Linux platform. In deze module gaan we hier dan ook gebruik van maken.

LEERDOELEN

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)
- Ik maak een eenvoudige webapplicatie in C# (ASP.net) die gebruik maakt van een databasekoppeling
- Ik lees in C# een array uit en manipuleer de gegevens binnen het array.
- Ik gebruik in C# de datumtypes indien nodig
- Ik verwerk in C# data die verzonden is via een formulier
- Ik geef een databasestructuur weer in een ERD
- Ik pas SQL-commando's toe voor database bewerkingen

WERKVORM

Bij een aantal oefeningen en opdrachten wordt er aanvullend een extra werkvorm aangeboden. Deze herken je aan de icoontjes aan het begin van de tekst.

Als je het lastig vindt om de opdracht direct te begrijpen bij het lezen ervan, kan je nog een extra instructiefilmpje bekijken.

Vind je het lastig om een begin te maken, dan kan je een project downloaden waarin al een start gemaakt is en waar in commentaarregels de vervolgstappen staan.

TIJDSPLANNING

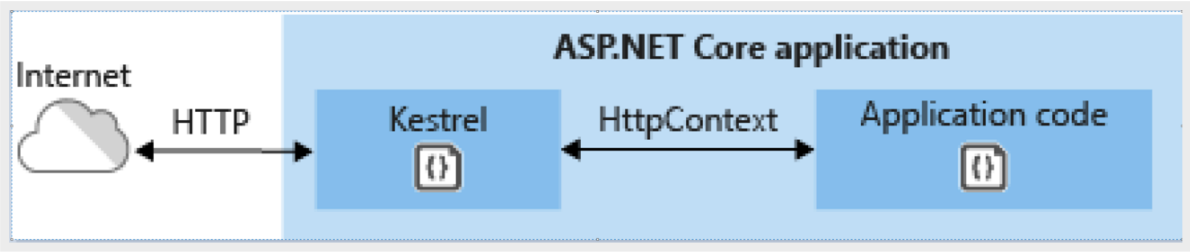
Week 1		
BASIS1, les 1	2 uur	1 De web-applicatie 2 De OnGet methode 3 Razor code 4 Opdracht 1-1
BASIS1, les 2	2 uur	1 Meerdere Pagina's 2 Opdracht 2-1
Week 2		
BASIS1, les 3	2 uur	1 De OnPost Methode: formulieren verwerken 2 Opdracht 3-1
BASIS1, les 4	2 uur	1 Array en List 2 Session variabelen 3 Opdracht 4-1
Week 3		
BASIS1, les 5	2 uur	1 Datumtypes 2 Opdracht 5-1
BASIS1, les 6	2 uur	1 Database en SQL 2 Opdracht 6-1
Week 4		
BASIS1, les 7	2 uur	1 Database en ERD 2 Opdracht 7-1 3 Opdracht 7-2
BASIS1, les 8	2 uur	Uitwerken van opdrachten die nog niet klaar zijn.

LES 1 - 2 UUR – 1 DE WEB-APPLICATIE

INLEIDING

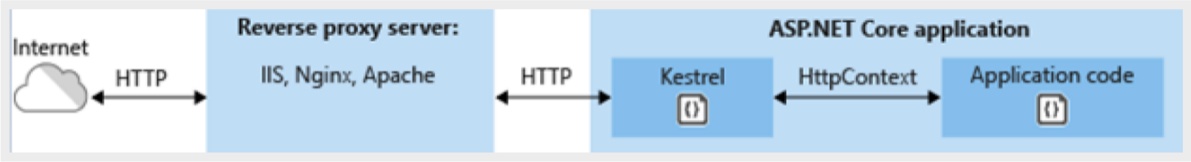
Met als bij de vorige module over C# gaan we in deze lessenserie ook weer werken met Visual Studio (Visual Studio Community 2019 voor Windows of Visual Studio for Mac).

Als we straks in de oefeningen en opdrachten een project binnen Visual Studio aanmaken voor een ASP.net webapplicatie is dat een ander type dan de console applicatie die we in de vorige C# module behandeld hebben. Toch is er een overeenkomst. De webapplicatie is in de basis een console applicatie. Er is dus ook een Program.cs file met daarin een main functie waarmee de applicatie begint bij het opstarten. Bij dit opstarten van de applicatie (kijk ook naar de video's Webapplicatie maken en opstarten (...).mp4 en ASP.net project structuur.mp4) zal er een webserver opgestart waarin de wenapplicatie gehost wordt. Dit zie je in de volgende schematische weergave.



In de afbeelding hierboven zie je dat er een webserver met de naam Kestrel opgestart wordt. Dit gebeurt dus ook als je jouw webapplicatie vanuit Visual Studio opstart met behulp van de play knop (zie video's).

Als de gehele applicatie gereed is en op een publicatieserver geplaatst moet worden, dan wordt er meestal gekozen voor onderstaande structuur. De Kestrel webserver die in de applicatie zelf zit wordt gekoppeld aan een reverse proxy server. In deze module gaan we daar niet verder op in.



EXTERNE BRONNEN

Om Visual Studio te downloaden: https://visualstudio.microsoft.com/downloads/

Als je Visual Studio thuis nog moet installeren op een Windows machine, let dan op het volgende:

- Selecteer tijdens de installatie voor de volgende optie:

Als je Visual Studio al geïnstalleerd hebt op een Windows omgeving, maar je bent bovenstaande vergeten te installeren, dan kan je de installer opnieuw gebruiken. Kies dan de optie modify, waardoor je de keuze alsnog kan maken. De installatie wordt dan aangevuld.

Op de Mac variant hoef je deze keuze niet te maken. Webdevelopment wordt automatisch mee geïnstalleerd.

Voorbeeld video's:

- Webapplicatie maken en opstarten (Mac).mp4
- Webapplicatie maken en opstarten (Windows).mp4
- ASP.net project structuur.mp4

Razor code: https://www.learnrazorpages.com/razor-syntax

THEORIE AAN DE HAND VAN OEFENINGEN

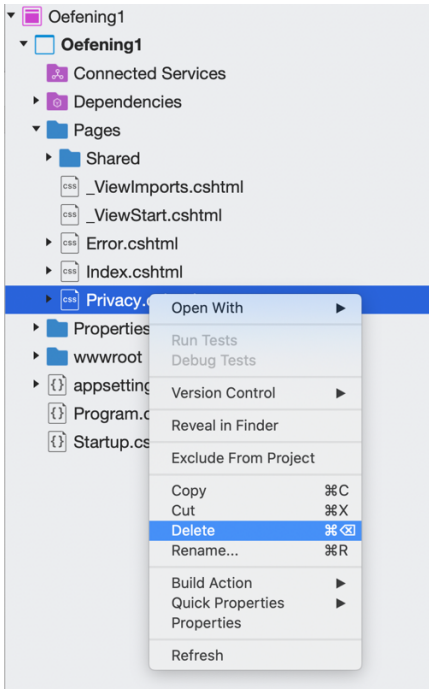
Bekijk eerst bovenstaande video's waarin een deel van de theorie wordt besproken. Na het bekijken gaan we een eerste eenvoudige webapplicatie maken in de volgende oefeningen. Deze oefeningen nemen je in kleine stappen mee tot een eerste resultaat.

OEFENING 1A:

- Maak een nieuw project aan met de naam Oefening1 zoals dat in de video is voorgedaan

We gaan eerst een aantal zaken verwijderen die nu overbodig zijn:

- Verwijder Privacy.cshtml:



- Open het bestand Index.cshtml.cs en verwijder een deel van de code zodat je alleen het volgende overhoudt:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5   using Microsoft.AspNetCore.Mvc;
6   using Microsoft.AspNetCore.Mvc.RazorPages;
7   using Microsoft.Extensions.Logging;
8
9   namespace Oefening1.Pages
10  {
11      public class IndexModel : PageModel
12      {
13          public void OnGet()
14          {
15
16          }
17      }
18  }
```

- Open de shared folder. Open daarna het bestand _Layout.cshtml en verwijder het header en footer element met inhoud. Je houdt dan het volgende over:

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8" />
5       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6       <title>@ViewData["Title"] - Oefening1</title>
7       <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
8       <link rel="stylesheet" href="~/css/site.css" />
9   </head>
10  <body>
11      <div class="container">
12          <main role="main" class="pb-3">
13              @RenderBody()
14          </main>
15      </div>
16
17      <script src="~/lib/jquery/dist/jquery.min.js"></script>
18      <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
19      <script src="~/js/site.js" asp-append-version="true"></script>
20
21      @RenderSection("Scripts", required: false)
22  </body>
23  </html>
```

- Open het bestand Index.cshtml en pas de code aan tot:

```
1   @page
2   @model IndexModel
3   @{
4       ViewData["Title"] = "Home page";
5   }
6
7   <div class="text-center">
8       <h1>Dit is mijn eerste C# webapplicatie</h1>
9   </div>
```

- Start de applicatie op. Als het goed is zie je in de browser nu het volgende

Dit is mijn eerste C# webapplicatie

LES 1 - 2 UUR – 2 DE ONGET METHODE

De functie OnGet() (officieel heet dit een methode in plaats van een functie) in het bestand Index.cshtml.cs wordt uitgevoerd als de index pagina wordt aangeroepen. Op dit moment is deze functie nog helemaal leeg, waardoor alleen de html code in Index.cshtml wordt getoond. In de volgende oefening gaan we deze OnGet() methode gebruiken voor een bepaalde actie.

OEFENING 1B:

We gaan verder met het project van oefening 1A.

- Pas de code in Index.cshtml aan tot:

```
1   @page
2   @model IndexModel
3   @{
4       ViewData["Title"] = "Home page";
5   }
6
7   <div class="text-center">
8       <h1>Dit is mijn eerste C# webapplicatie</h1>
9       <p>
10          <a href="/?nummer=1">nummer 1</a>
11      </p>
12      <p>
13          <a href="/?nummer=2">nummer 2</a>
14      </p>
15      <p>
16          <a href="/?nummer=3">nummer 3</a>
17      </p>
18      <p>
19          <a href="/?nummer=4">nummer 4</a>
20      </p>
21      <p>
22          Je hebt nummer @Model.DitNummer geklikt
23      </p>
24  </div>
```

In deze code zijn vier linkjes naar dezelfde pagina gemaakt. Achter de / wordt een parameter meegestuurd met aanroep, die we in de OnGet() methode kunnen gebruiken. Op regel 22 staat nog een stuk tekst met daarin een stuk code achter het gele @ teken. Waar dit voor is zullen we straks zien.

- Pas de code in Index.cshtml.cs aan tot:

```
dexModel ▸ OnGet()
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5   using Microsoft.AspNetCore.Mvc;
6   using Microsoft.AspNetCore.Mvc.RazorPages;
7   using Microsoft.Extensions.Logging;
8
9   namespace Oefening1.Pages
10  {
11      public class IndexModel : PageModel
12      {
13          public string DitNummer;
14
15          public void OnGet()
16          {
17              string nummer = Request.Query["nummer"];
18
19              if (nummer != null)
20              {
21                  DitNummer = nummer;
22              }
23          }
24      }
25  }
```

In de OnGet() methode wordt de meegestuurde parameter uitgelezen en in een string variabele gezet (regel 17). Op regel 19 wordt met het if-statement gecontroleerd of de stringvariabele een waarde heeft gekregen. Dit is het geval als we op een link geklikt hebben. Als we niet op een link geklikt hebben, bijvoorbeeld als we voor het eerst op de index pagina komen, dan is de string variabele niet gevuld. In dit laatste geval wordt alle code binnen het if-statement niet meer uitgevoerd.

Als we wel op een link geklikt hebben wordt de waarde van de stringvariabele gekopieerd naar de variabele DitNummer. Deze variabele is op regel 13 als public string gedeclareerd (dus boven de OnGet() methode). Door deze declaratie kunnen we de variabele DitNummer gebruiken binnen de HTML code van Index.cshtml. Dat zie je aan @Model. DitNummer.

Wat we dus eigenlijk doen is een stukje C# code opnemen in de HTML code:

```
<p>
    Je hebt nummer @Model.DitNummer geklikt
</p>
```

- Test nu maar eens uit wat deze code doet door de applicatie op te starten

LES 1 - 2 UUR – 3 RAZOR CODE

Razor code:

Als we C# code opnemen binnen HTML code met behulp van het gele @ teken wordt de code **Razor code** genoemd. Dit geeft heel veel extra mogelijkheden waarvan we een gedeelte zullen bekijken in deze module.

Alle C# code in het Index.cshtml.cs bestand en binnen het HTML bestand wordt eerst op de server uitgevoerd voordat de pagina naar de browser gestuurd wordt.

In de volgende oefening gaan we nog meer C# code opnemen binnen de HTML code

OEFENING 1C:

We gaan verder met het project van oefening 1B.

- Breng in het onderste gedeelte van Index.cshtml de volgende aanpassing aan:

```
    <p>
        <a href="/?nummer=3">nummer 3</a>
    </p>
    <p>
        <a href="/?nummer=4">nummer 4</a>
    </p>
    @if (Model.DitNummer != null)
    {
        <p>
            Je hebt nummer @Model.DitNummer geklikt
        </p>
    }
</div>
```

- Test de code en bekijk wat het verschil is

In bovenstaande oefening zie je dat je ook een if-statement kan opnemen binnen de HTML code. Met het gele @ teken schakel je over van HTML naar C#. Zodra je binnen de accolades { } een HTML element tegen komt (bijvoorbeeld <p>) schakel je weer over naar HTML

Om Razor code te schrijven heb we de volgende opties:

- Losse variabelen die in de C# code als public gedeclareerd zijn:

```
<p>
    Je hebt nummer @Model.DitNummer geklikt
</p>
```

- Code binnen accolades behorende bij een if-statement

```
@if (Model.DitNummer != null)
{
    <p>
        Je hebt nummer @Model.DitNummer geklikt
    </p>
}
```

- Hetzelfde kan uiteraard ook met andere structuren waarbij je accolades gebruikt zoals een for-lus of while-lus:

```
@for (int i = 0; i < 10; i++)
{
    <p>
        Het nummer is: @i
    </p>
}
```

- Je kan ook een codeblok maken door alleen een @ en accolades te gebruiken:

```
@{
    int getal1 = 5;
    int getal2 = 10;
    int getal3 = getal1 + getal2;
    <p>de uitkomst is @getal3</p>
}
```

EXTERNE BRONNEN

Razor code: https://www.learnrazorpages.com/razor-syntax

LES 1 - 2 UUR – 4 OPDRACHT 1-1

JOUW GEGEVENS

In deze opdracht maak je een webapplicatie zoals in oefening1A t/m Oefening1C, maar nu met drie linkjes. In de eerste link stuur je jouw naam als een parameter mee naar de server, bijvoorbeeld:

```
<a href="/?mijnNaam=Peter Pan">mijn naam</a>
```

In de tweede link stuur je jouw studentnummer naar de server met een andere parameternaam, bijvoorbeeld:

```
<a href="/?mijnStudentnummer=12345">mijn studentnummer</a>
```

In de derde link stuur je jouw leeftijd mee in met een andere parameternaam.

In de C# code, binnen de OnGet methode, lees je de drie parameters (apart) uit. Daarna controleer je of de parameter met de link was meegestuurd en zorg je ervoor dat je juiste tekst op de pagina verschijnt. In de afbeeldingen hieronder zie je per link welke tekst moet verschijnen (uiteraard met jouw gegevens en niet die uit het voorbeeld):

Voor meer uitleg kan je nog het volgende filmpje bekijken:
Opdracht1_1.mp4

In het project Opdracht_1_1 is een begin gemaakt met de code en in commentaarregels staan de vervolgstappen aangegeven. Download voor dit project het bestand Opdracht_1_1.zip

BENODIGDE KENNIS

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)

UITBREIDING (EXTRA)

- Declareer de variabele voor mijnLeeftijd en MijnStudentnummer als een integer (in de c# code) en pas de code verder aan zodat de applicatie weer werkt.

LES 2 - 2 UUR – 1 MEERDERE PAGINA'S

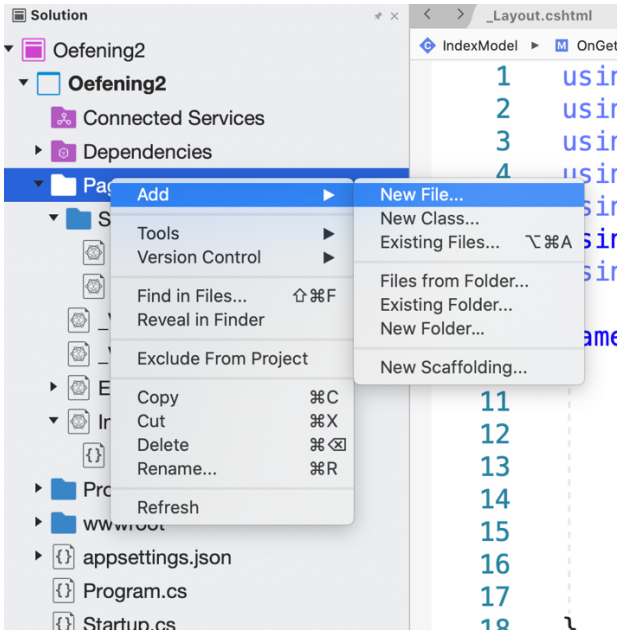
THEORIE

Een webapplicatie bestaat vaak uit meerdere pagina's. In het vorige hoofdstuk hebben we gewerkt met 1 pagina die uit een Index.cshtml file bestond en een Index.cshtml.cs. De eerste file bevatte voornamelijk HTML code met een beetje C# code (In de vorm van Razor code). De tweede file bevatte enkel C# code. In dit hoofdstuk gaan we meerdere pagina's gebruiken die ook weer uit twee delen bestaan.

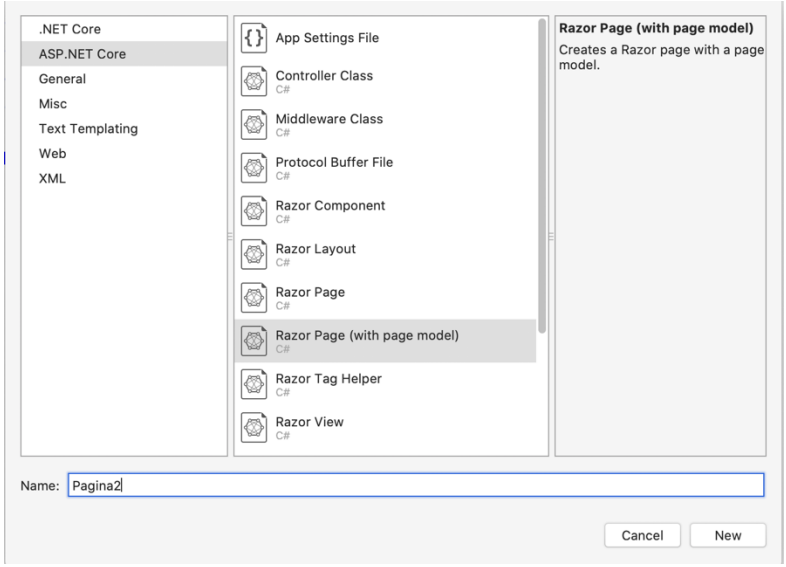
Aan de hand van de volgende oefening wordt uitgelegd hoe je met meerdere pagina's kan werken.

OEFENING 2A:

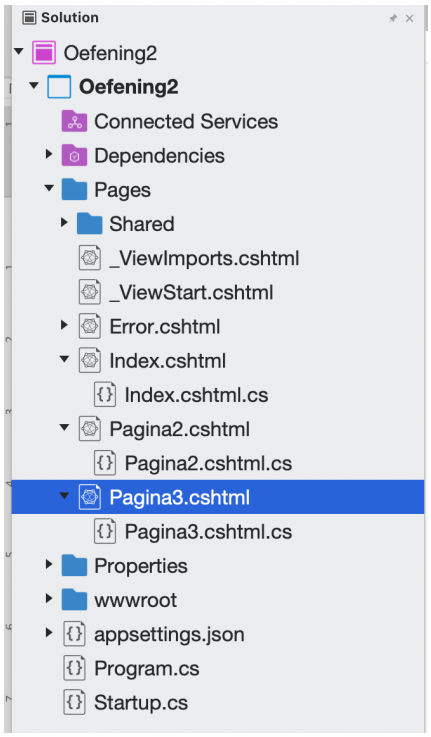
- Maak een nieuw project aan met de naam Oefening2.
- Verwijder het bestand en de code die we nu niet nodig hebben zoals je dat ook in oefening1A gedaan hebt.
- Maak nu een nieuwe pagina aan door met je rechtermuisknop te klikken op de map Pages en selecteer daarna New File.



In het venster dat verschijnt kies je de optie Razor Page (with page model) en vul je een naam voor de pagina in (pagina2)



- Doe hetzelfde voor een derde pagina. Als het gelukt is zie je het volgende in jouw project staan:



- Open nu het bestand _Layout.cshtml en zet de volgende code (header tag met alles daarbinnen) direct in de body:

```
<body>
    <header>
        <nav>
            <ul>
                <li><a href="/Index">Home</a></li>
                <li><a href="/Pagina2">Pagina 2</a></li>
                <li><a href="/Pagina3">Pagina 3</a></li>
            </ul>
        </nav>
    </header>
    <div class="container">
```

- Vervang de CSS code in site.css (in de wwwroot >> CSS map) door:

```
nav {
    margin: 0;
    background-color: #E64A19;
}

nav ul {
    padding: 0;
    margin: 0;
    list-style: none;
    position: relative;
}

nav ul li {
    display: inline-block;
    background-color: #E64A19;
}

nav a {
    display: block;
    padding: 0 10px;
    color: #FFF;
    font-size: 20px;
    line-height: 60px;
    text-decoration: none;
}

nav a:hover {
    background-color: #000000;
}
```

- Vervang de HTML code in Index.cshtml door de volgende code:

```
<div class="text-center">
    <h3 class="display-4">Dit is de Home Pagina</h3>
</div>
```

- Zet vergelijkbare code in Pagina2.cshtml en in Pagina3.cshtml

```
<div class="text-center">
    <h3 class="display-4">Dit is de Pagina 2</h3>
</div>
```

- Test nu de applicatie door deze op te starten.

Als je alles goed gedaan hebt krijg je een menubalk in beeld, waarmee je naar de verschillende pagina's kan springen:

De linkjes in oefening2A, die in de menubalk staan hebben geen parameter. De OnGet methodes op de drie pagina's doen dus nog niets. Als je naar de code kijkt van de OnGet methode op de verschillende pagina's zie je dat die ook nog leeg is. In de volgende oefening gaan we daar verandering in aanbrengen. We zetten op elke pagina nog twee extra linkjes naar de andere pagina's. In deze linkjes worden die parameters doorgegeven. Met die drie parameters geven we aan hoeveel keer er naar een bepaalde pagina gesprongen.

OEFENING 2B:

- Zet in Index.cshtml de volgende code direct onder de H3 tag:

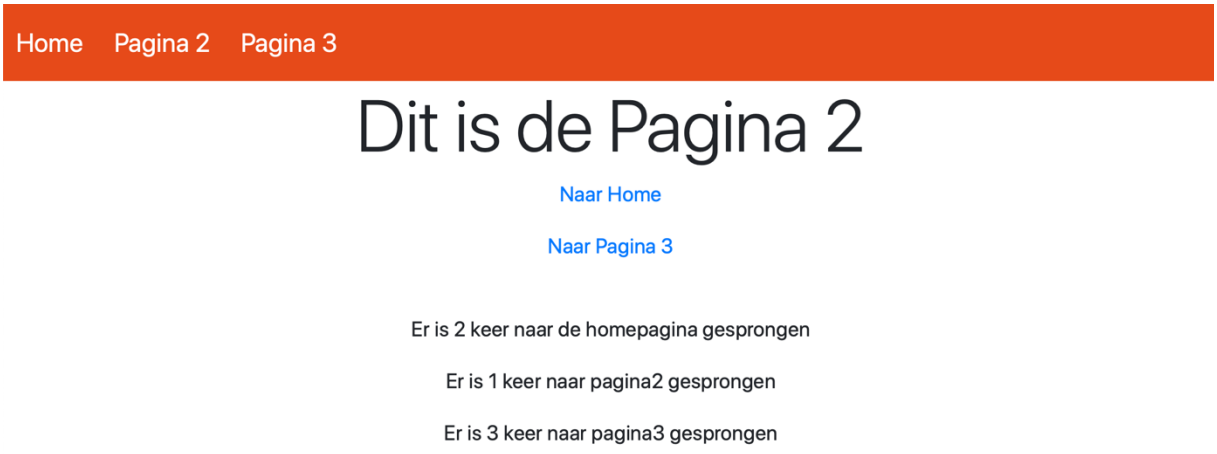
```
<p>
    <a href="/Pagina2?aantalHome=@Model.aantalKeerHome
            &aantalPagina2=@Model.aantalKeerPagina2
            &aantalPagina3=@Model.aantalKeerPagina3">Naar Pagina 2</a>
</p>
<p>
    <a href="/Pagina3?aantalHome=@Model.aantalKeerHome
            &aantalPagina2=@Model.aantalKeerPagina2
            &aantalPagina3=@Model.aantalKeerPagina3">Naar Pagina 3</a>
</p>
<br />
<p>Er is @Model.aantalKeerHome keer naar de homepagina gesprongen</p>
<p>Er is @Model.aantalKeerPagina2 keer naar pagina2 gesprongen</p>
<p>Er is @Model.aantalKeerPagina3 keer naar pagina3 gesprongen</p>
```

- Zet in Index.cshtml.cs de volgende code binnen de public class ....

```
public int aantalKeerHome = 0;
public int aantalKeerPagina2 = 0;
public int aantalKeerPagina3 = 0;

public void OnGet()
{

    string aantalHome = Request.Query["aantalHome"];
    if (aantalHome != null) {
        aantalKeerHome = int.Parse(aantalHome);
        string aantalPagina2 = Request.Query["aantalPagina2"];
        aantalKeerPagina2 = int.Parse(aantalPagina2);
        string aantalPagina3 = Request.Query["aantalPagina3"];
        aantalKeerPagina3 = int.Parse(aantalPagina3);

        aantalKeerHome++;
    }
}
```

- Doe het zelfde op de andere pagina's met de volgende aanpassingen:
    - Pas de linkjes aan zodat ze naar der juiste pagina's springen en de juiste tekst wordt weergegeven
    - Pas de code aantalKeerHome++; aan. De variabele die verwijst naar de pagina waarin de code staat moet met 1 opgehoogd worden.
- Test de applicatie

Als de applicatie goed werkt worden de aantal kliks steeds naar de volgende pagina doorgegeven en getoond op het scherm:



Wat opvalt is dat bij elke link alle waarden doorgegeven moeten worden naar de volgende pagina als we die willen bewaren. Dit zie je ook als je een link in de rode menubalk aanklikt. Er worden dan geen waarden doorgegeven en staan de getallen in de onderste drie regels weer op nul.

LES 2 - 2 UUR – 2 OPDRACHT 2-1

OPTELLEN

Maak een webapplicatie met drie pagina's. Op de index pagina staat vijf linkjes met als tekst de getallen van 1 t/m 5:

Kies een getal

1 2 3 4 5

Als de gebruiker op een link klikt wordt dat getal als parameter meegestuurd naar pagina2 die er als volgt uitziet:

Kies een tweede getal

6 7 8 9

De vier getallen zijn ook weer linkjes die naar de derde pagina verwijzen. Bij een klik op deze linkjes wordt het geklikte getal van de index pagina en het geklikte getal van deze pagina als parameter naar de derde pagina gestuurd. Op de derde pagina wordt dan de optellingen van die getallen getoond. Als er dus eerst op 3 en daarna op 8 geklikt wordt ziet pagina 3 er zo uit:

De optelling is 11

terug naar Home

Op de derde pagina is ook een link naar de homepagina.

Voor meer uitleg kan je nog het volgende filmpje bekijken:
Opdracht2_1.mp4

In het project Opdracht_2_1 is een begin gemaakt met de code en in commentaarregels staan de vervolgstappen aangegeven. Download voor dit project het bestand Opdracht_2_1.zip

BENODIGDE KENNIS

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)

UITBREIDING (EXTRA)

- Zet tussen pagina 2 en 3 een extra pagina. Op deze pagina kan de gebruiker kiezen wat er met de getallen moet gebeuren: optellen, aftrekken, vermenigvuldigen of delen. Op de laatste pagina wordt dan ook aangegeven welke bewerking is uitgevoerd en wat het antwoord is.

LES 3 - 2 UUR – 1 DE ONPOST METHODE: FORMULIEREN VERWERKEN

INLEIDING

In de vorige hoofdstukken hebben we gezien dat de OnGet methode reageert op een link naar de pagina, waarbij meegezonden parameters gebruikt kunnen worden bij de weergave op de pagina. In dit hoofdstuk gaan we bekijken wat er moet gebeuren als er een formulier wordt ingevuld en die verzonden wordt naar een pagina.

Als er data binnenkomt op een pagina die verzonden is vanuit een HTML formulier, dan wordt de OnPost methode gebruikt. Dit zie je in de volgende oefening:

OEFENING 3A:

- Maak een nieuw project aan met de naam Oefening3.
- Verwijder het bestand en de code die we nu niet nodig hebben zoals je dat ook in oefening1A gedaan hebt.
- Zet in Index.cshtml de volgende code:

```
<div class="text-center">
    <h5 class="display-4">Vul de volgende gegevens in</h5>
    <form method="post">
        <p>
            <label for="naam">Vul een naam in:</label>
            <input type="text" name="naam" />
        </p>
        <p>
            <label for="leeftijd">Vul een leeftijd in:</label>
            <input type="number" name="leeftijd" />
        </p>
        <p>
            <button type="submit">verstuur</button>
        </p>
    </form>

    <p>@Model.resultaat</p>
</div>
```

De code bevat een formulier, die naar dezelfde pagina gestuurd wordt (er is geen action binnen de form tag), met twee inputvelden en een submit button. Beide inputvelden hebben een attribuut (eigenschap) name. De waarde hiervan is belangrijk. Die hebben we nodig in de OnPost methode. De OnPost methode bestaat nog niet in Index.cshtml.cs. Die gaan we nu dus aanmaken:

- Zet de volgende code in Index.cshtml.cs (onder de OnGet methode):

```
public void OnPost(string naam, int leeftijd)
{

}
```

In de OnPost methode nemen we de twee parameters op. Dit zijn de waarden van de name attributen uit het formulier. Hiermee geven we de waarde, die door de gebruiker ingevoerd wordt, door aan de OnPost methode.

- Zet in Index.cshtml.cs de volgende public variabele (Let goed op waar je het neerzet):

```
public string resultaat;
```

- Zet nu in de OnPost methode de volgende code:

```
resultaat = $"Jouw naam is {naam} en je bent {leeftijd} jaar oud.";
```

- Test de applicatie:

Na versturen:

In de vorige oefening werd het formulier naar dezelfde pagina verstuurd. Je kan ook het formulier naar een andere pagina sturen, bijvoorbeeld Pagina2. In dit geval zal de OnPost methode van Pagina2 de verwerking uitvoeren.

In de HMTL modules heb je gezien dat je met het action attribuut kan aangeven dat het formulier naar een andere pagina gestuurd moet worden:

```
<form method="post" action="/Pagina2">
```

In de door ons gebruikte Razor pages werkt dit (door ingebouwde security) niet. Het formulier wordt wel verzonden, maar niet verwerkt. Je krijgt een lege pagina. De juiste manier van verzenden is:

```
<form method="post" asp-page="Pagina2">
```

OEFENING 3B:

Bouw oefening 3A om zodat het formulier naar Pagina2 gestuurd wordt. Deze pagina moet je uiteraard nog erbij maken. Na verzending en verwerking toont Pagina2 de volgende tekst:

Jouw naam is Peter en je bent 18 jaar oud.

Tip: verplaats bepaalde stukken code van de Index pagina naar Pagina2

In de video terugspringen.mp4 zie je nog een extra mogelijkheid, namelijk na verwerking van de OnPost methode kan je ook automatisch terugspringen naar de Index pagina.

EXTERNE BRONNEN

Voorbeeld video: terugspringen.mp4

Hoe neem je afbeeldingen op in een project: Opdracht_3_1.mp4

LES 3 - 2 UUR – 2 OPDRACHT 3-1

BANK BESTELLEN

Bij deze opdracht krijg je een project waarin je verder gaat werken. Download hiervoor het bestand Opdracht_3_1.zip met daarin het project. Als je het project opent en opstart zie je het volgende in de browser:

De afbeeldingen staan al in het project. Hoe je zelf afbeeldingen in een project kan opnemen en gebruiken zie je in de video afbeeldingen.mp4. De knoppen zijn linkjes (die met CSS opgemaakt zijn als knoppen) die naar een andere pagina springen. Eén van de linkjes ziet er als volgt uit:

```
<a class="btn btn-danger" href="/Bestel?bank=Armour">Bekijk</a>
```

In deze link zie je dat deze verwijst naar de pagina met de naam Bestel en dat de naam van de gekozen bank als parameter meegestuurd wordt. De pagina met de naam Bestel bestaat nog niet. Die moet je zelf aanmaken.

In het bestand Bank.txt staat voor elke bank aanvullende informatie (naam, type, beschrijving en prijs) die je moet gebruiken op de Bestel pagina.

Op de bestelpagina wordt deze aanvullende informatie getoond van de bank die op de index pagina geselecteerd is. Daaronder komt een invulformulier met bestelknop zoals in de volgende afbeelding.

Je hebt de volgende keuze gemaakt

Naam:	Armour
Type:	2-zitsbank
Beschrijving:	De Armour is een zeer voordelig geprijsde designbank met een eigenzinnige uitstraling. De Armour is verkrijgbaar in een groot aantal bekledingssoorten en kleuren.
Prijs:	449.00

Vul jouw gegegens in om te bestellen

naam:

adres:

woonplaats:

Bestel

Als het formulier ingevuld is en er wordt op de bestelknop geklikt, dan worden de gegevens uit het formulier en de naam van de bank als parameter meegestuurd met een redirect (sprong terug) naar de index pagina. Op deze indexpagina worden deze gegevens in het middelste kader getoond zoals in de afbeelding hieronder:

Te nemen stappen:

- Pagina Bestel aanmaken
- Lees in de OnGet methode de meegezonden parameter uit.
- Toon de gegevens uit bank.txt voor de geselecteerde bank op de Bestel pagina met daaronder een invulformulier.
- Zet in het formulier ook een hidden input met de geselecteerde bank, zodat deze ook meegezonden wordt met het formulier:
  ```
  <input type="hidden" name="bank" value="@Model.naam"/>
  ```
  Hierin is @Model.naam de naam van de bank.
- Lees in de OnPost methode alle velden van het formulier uit.

- Verstuur vanuit de OnPost de gegevens naar de Index pagina met een redirect die eruit ziet zoals het volgende voorbeeld:

```
return RedirectToPage("Index", new { param1 = bank, param2 = naam, ...});
```

  Op de puntjes kan een derde en vierde parameter meegestuurd worden.
- Ga nu naar de Index pagina. Lees de meegestuurde parameters weer uit.
- Declareer voor elke parameter ook een public property.
- Kopieer de inhoud van de parameters naar de public property's.
- Toon de tekst voor de bestelde bank in het middenkader door gebruik te maken van de public property's.

Voor meer uitleg over de te nemen stappen staat in het volgende filmpje: Opdracht3_1.mp4

BENODIGDE KENNIS

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)
- Ik verwerk in C# data die verzonden is via een formulier

UITBREIDING (EXTRA)

- Toon op de bestelpagina ook de afbeelding van de geselecteerde bank
- Toon in het middenkader (bij de bevestiging van de bestelling) ook de prijs van de bank.

LES 4 - 2 UUR – 1 ARRAY EN LIST

In Javascript hebben we al met array's gewerkt. We hebben gezien dat het een variabele is waarin we meerdere waarden kunnen opslaan. Elke waarde heeft dan zijn eigen index. Hieronder nog een voorbeeld in javascript hoe je een array aanmaakt en kan vullen met waarden.

```
var mijnArray = new Array();
mijnArray.push(11);      // Voeg de waarde 11 toe aan mijnArray
mijnArray.push(27);      // Voeg de waarde 27 toe aan mijnArray
mijnArray.push(8);       // Voeg de waarde 8 toe aan mijnArray
```

De waarde 11 is opgeslagen in mijnArray[0] en de waarde 27 is opgeslagen in mijnArray[1].

Met push kan je in Javascript onbeperkt waarden toevoegen aan het array. Je kan ook de waarden in één keer in het array plaatsen:

```
var mijnArray = [11, 27, 8]; // Zet in mijnArray de waardes 11, 27, 8
```

In C# gelden andere regels. Een array in C# heeft een vaste lengte en het aanmaken (declareren) van het array gaat ook iets anders:

```
string[] cars = new string[4];
cars[0] = "Volvo";
cars[1] = "BMW";
cars[2] = "Ford";
cars[3] = "Mazda";
```

Of:

```
string[] cars = new string[4] { "Volvo", "BMW", "Ford", "Mazda" };
```

Bekijk de tutorials en video's onder het kopje externe bronnen.

EXTERNE BRONNEN

Tutorials:

https://www.w3schools.com/cs/cs_arrays.asp

Video's:

https://www.youtube.com/watch?v=BDfO1ryG6Rc

https://www.youtube.com/watch?v=_kgQQmz7fak

In plaats van een array kan je ook een List gebruiken. Dit is eigenlijk ook een soort array, maar die heeft geen vaste lengte en lijkt in het gebruik meer op het array zoals in Javascript. We zullen de List in de volgende video bekijken: List.mp4

EXTERNE BRONNEN

Tutorial:

https://www.learncs.org/en/Lists

Video:

List.mp4

LES 4 - 2 UUR – 2 SESSION VARIABELEN

In de video List.mp4 in het vorige hoofdstuk heb je gezien dat een nieuwe aanroep van een pagina volledig losstaat van de vorige. Er worden geen gegevens bewaard die bij een nieuwe aanroep weer gebruikt kunnen worden. Een oplossing hiervoor is het gebruik maken van session variabelen. Met session variabelen kunnen we op de webserver (per sessie, dus per gebruiker) data opslaan en later weer teruglezen.

Om sessie variabelen te kunnen gebruiken moeten we dit eerst aan ons project toevoegen in de vorm van extra code. Deze code zetten we in het bestand StartUp.cs. In dit bestand vullen we in de ConfigureServices functie de code aan tot onderstaande code:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();

    //session
    services.AddSession(options => {
        options.IdleTimeout = TimeSpan.FromMinutes(30);
        //zonder deze optie is het standaard 20 minuten
    });
    services.AddMemoryCache();
}
```

In de functie Configure zetten we app.UseSession() op de juiste plaats ertussen:

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        // The default HSTS value is 30 days. You may want to change th
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseSession();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```

Op elke pagina waar je een sessie variabele wil gebruiken met je nog wel bovenaan de C# code een extra using opnemen:

```
using Microsoft.AspNetCore.Http;
```

Als we meer data op wil nemen in een sessie variabele zoals de inhoud van een array of List, dan wordt er vaak gebruik gemaakt van JSON. JSON is een manier om de inhoud van een array of List in een string te verpakken. Ook hiervoor moeten we een extra using opnemen in de code:

```
using System.Text.Json;
```

De List met boodschappen uit de video List.mp4 kan met JSON in de volgende string omgezet worden:

```
["kaas","thee","suiker"]
```

Hierdoor kan die string eenvoudig in de sessie opgeslagen worden. Op de volgende pagina kan die string weer uit de sessie variabele gehaald worden en teruggezet in de List. Hier dit werkt zie je in de video List_deel2.mp4.

LES 4 - 2 UUR – 2 OPDRACHT 4-1

BANK BESTELLEN – DEEL 2

In deze opdracht maken we gebruik van het resultaat van opdracht3_1. Maak hiervoor een kopie van de gehele map en hernoem alleen de buitenste map naar opdracht4_1.

In opdracht3_1 werd de bestelde bank via parameters doorgegeven van de bestel pagina naar de index pagina:

```
return RedirectToPage("Index", new { param1 = bank, param2 = naam, ...});
```

In deze opdracht gaan we dat aanpassen. We gaan niet meer de parameters gebruiken, maar een Sessie variabele om deze data op te slaan er weer in te lezen. De sprong terug van de bestel pagina naar de index pagina gaat er nu zo uitzien.

```
return RedirectToPage("Index");
```

De te nemen stappen zijn:

- Verander de RedirectToPage zoals hierboven aangegeven (op de bestel pagina).
- Maak in de OnPost() van de bestelpagina een List aan van het type string.
- Zet hieronder alle binnenkomende gegevens vanuit het ingevulde formulier in deze List (dus banknaam, naam van de besteller, adres, en woonplaats).
- Zet de List met gegevens in een sessie variabele (maak ook gebruik van JSON)
- Lees in de OnGet() van de Index pagina de sessie variabele uit en zet de data in de juiste public property's, voorbeeld:

  ```
  gekozenBank = bestelling[0];
  ```

  Hierbij is bestelling[0] het eerste element uit het List object die uit de Sessievariabele gelezen is.
  Maak aan het einde van de OnGet() (van de Index pagina) de sessie variabele leeg. Dit doe je met het volgende commando:

  ```
  HttpContext.Session.Clear();
  ```

Als je alles goed uitgevoerd hebt, werkt de applicatie weer zonder fouten.

BENODIGDE KENNIS

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)
- Ik lees in C# een array uit en manipuleer de gegevens binnen het array

UITBREIDING (EXTRA)

De extra's zijn hetzelfde als bij opdracht3_1, maar nu in de structuur van deze opdracht:

- Toon op de bestelpagina ook de afbeelding van de geselecteerde bank
- Toon in het middenkader (bij de bevestiging van de bestelling) ook de prijs van de bank.

LES 5 - 2 UUR – 1 DATUMTYPES

INLEIDING

In de laatst gemaakte opdrachten wordt er elke keer een bank besteld. Bij zo'n bestellingen willen we eigenlijk ook de besteldatum weten en wanneer de bank geleverd wordt, bijvoorbeeld op een datum 6 weken later. Om met datums te werken is er in C# een apart type variabele. Het is geen int, float of string, maar een DateTime.

Bekijk de video over DateTime.

EXTERNE BRONNEN

Video over Datetime:
https://www.youtube.com/watch?v=KKzSQ6r93dY

LES 5 - 2 UUR – 2 OPDRACHT 5-1

BANK BESTELLEN – DEEL 3

In deze opdracht maken we gebruik van het resultaat van opdracht4_1. Maak hiervoor een kopie van de gehele map en hernoem alleen de buitenste map naar opdracht5_1.

Als er een bestelling getoond wordt op de Index pagina, dan wordt ook de besteldatum getoond (dat is vandaag) en de verwachte leverdatum over 6 weken. Let erop dat de data in de juiste volgorde staat (dag – maand – jaar) :

BENODIGDE KENNIS

- Ik gebruik in C# de datumtypes indien nodig

LES 6 - 2 UUR – 1 DATABASE EN SQL

INLEIDING

Een goede webapplicatie maakt meestal gebruik van een database, waarin de informatie staat die op de site getoond moet worden. In de opdrachten die wij tot nu toe gemaakt hebben stond de informatie (de gegevens van de verschillende banken) direct in onze code opgenomen. In dit hoofdstuk gaan we deze informatie uit een database uitlezen met behulp van een SQL-commando en daarna tonen op de pagina.

THEORIE

Een database bestaat uit één of meer tabellen. Deze tabellen kunnen we uitlezen met behulp van SQL-commando's. Een voorbeeld hier is:

CustomerID	CustomerName	ContactName	Address	City	PostalCode
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22

Gegevens uit deze tabel kunnen we uitlezen met:

```
SELECT CustomerName, City FROM Customers;
```

In dit commando selecteren we de namen en de woonplaats van alle klanten.

Meer informatie over databases en SQL commando's vind je in het externe bestand Database_info.pdf.

In de opdracht gaan wij een SQLite database gebruiken. Dit is een bestand die we direct in ons C# project kunnen plakken en er gebruik van maken. De database heeft twee tabellen waarvan we er in dit hoofdstuk één van gaan gebruiken. In de tabel (met de naam: meubels) staan de gegevens van de verschillende banken uit de vorige opdracht en:

artikelnr	naam	type	omschijving	prijs
Filter	Filter	Filter	Filter	Filter
10084532	Armour	2_zitsbank	De Armour is ee...	449
10085305	Baltica	2_zitsbank	De Baltica is een ...	599
10108335	Bari	3_zitsbank	De Bari is een L_...	720
40074107	Canada	6_zitsbank	De Canada is ee...	1299
40084724	Colorado	3_zitsbank	Dat is nog eens ...	799
50121189	Hawaii	5_zitsbank	De Hawaii is een ...	999.99

In de volgende video wordt getoond hoe je deze tabel in een C# applicatie kan uitlezen en kan tonen op de pagina: Database_voorbeeld.mp4

EXTERNE BRONNEN

Bestand: Database_info.pdf

Video: Database_voorbeeld.mp4

LES 6 - 2 UUR – 2 OPDRACHT 6-1

BANK BESTELLEN – DEEL 4

In deze opdracht maken we gebruik van het resultaat van opdracht5_1. Maak hiervoor een kopie van de gehele map en hernoem alleen de buitenste map naar opdracht6_1.

Op de bestelpagina staan alle gegevens van een bank in de code opgenomen en bij een selectie van een bank worden deze gegevens in de public variabele gezet, zodat deze getoond kunnen worden in de browser:

```
if (bank == "Armour")
{
    naam = "Armour";
    type = "2-zitsbank";
    beschrijving = "De Armour is een zeer voordelig gepri
    prijs = "449.00";
}
```

In deze opdracht verwijder je de gegevens van de banken uit de code. Als een bepaalde bank geselecteerd is, lees je de gegevens van die bank uit de database en zet je de ingelezen waarden in de public property's. De werking van de applicatie blijft dus hetzelfde, alleen komt de informatie nu uit de database in plaats vanuit de code.

Tip: je hebt geen List of Array's nodig zoals in de video. Je kan meteen een ingelezen waarde in een public property zetten:

```
naam = reader.GetString(1);              //naam bank
type = reader.GetString(2);              //type bank
beschrijving = reader.GetString(3);      //beschrijving
prijs = reader.GetInt32(4).ToString();   //prijs
```

BENODIGDE KENNIS

- Ik maak een eenvoudige webapplicatie in C# (ASP.net) die gebruik maakt van een databasekoppeling
- Ik pas SQL-commando's toe voor database bewerkingen

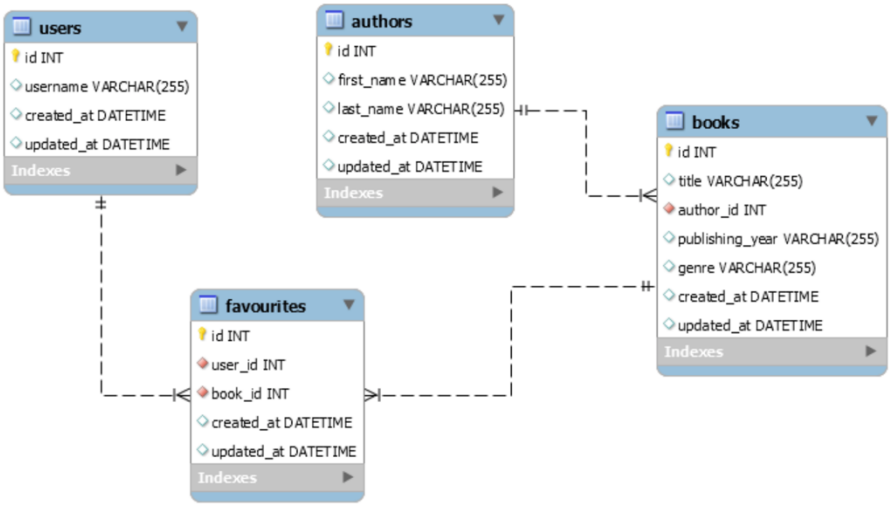
UITBREIDING (EXTRA)

Op de index pagina staan alleen de afbeeldingen van de banken. Lees ook op deze pagina de database uit en toon onder of boven het plaatje de naam en de prijs van de bank.

LES 7 - 2 UUR – 1 DATABASE EN ERD

THEORIE ERD

Als een database meerdere tabellen bevat, is er bijna altijd ook een bepaalde relatie tussen de verschillende tabellen. Deze relatie wordt dan vaak in een ERD (Entiteit-Relatie-Diagram) weergegeven. Een voorbeeld van een ERD is:



Dit ERD geeft vier tabellen weer. In de bovenste blauwe balk staat de naam van de tabel en daaronder in het wit staan de kolomnamen. De tabel users heeft dus 4 kolommen met de namen: id, username, created_at en updated_at. Elke tabel heeft ook een primairy key. Dit is de eerste kolom waar het sleuteltje bij staat. Vaak heeft deze kolom de naam id (zoals in de afbeelding). In die kolom wordt voor elke rij in de tabel een unieke waarde (bijvoorbeeld een nummer) ingevuld die de rij uniek maakt. Het nummer mag dus niet twee keer voorkomen. Met de lijnen worden de relaties tussen de tabellen weergegeven. Je ziet dat er een lijn loopt van de tabel authors naar de tabel books. In de tabel books staat ook een kolom met de naam author_id. Dit is een verwijzing naar de id kolom in de tabel authors. Door het nummer van de auteur in de tabel books op te nemen kan je achterhalen wie de auteur van dat book is. In leerjaar 2 zullen we hier dieper op ingaan.

THEORIE SCHRIJVEN NAAR DATABASE

In dit hoofdstuk gaan we ook bekijken hoe je gegevens in een database kan opslaan vanuit C# code. Op dezelfde manier als bij het inlezen van data moet er eerst een verbinding gemaakt worden met de database, waarna het wegschrijven kan plaatsvinden. We gebruiken dezelfde database uit de video Database_voorbeeld.mp4 en de laatste opdracht. Alleen gebruiken we nu niet de tabel meubels, maar de tabel bestellingen:

bestelnummer	naam	adres	woonplaats	artikelnr

Hierboven zie je de namen van de kolommen. Verder is de tabel nu nog helemaal leeg.

In de video Database_insert_voorbeeld.mp4 zie hoe je gegevens in deze tabel kan wegschrijven.

EXTERNE BRONNEN

Informatie over ERD:
https://www.lucidchart.com/pages/nl/wat-is-een-entity-relationship-diagram
https://www.youtube.com/watch?v=QpdhBUYk7Kk

Video: Database_ insert_voorbeeld.mp4

LES 7 - 2 UUR – 2 OPDRACHT 7-1

ERD

Maak van de database die we gebruiken (met de twee tabellen meubels en bestellingen) een ERD. Het diagram mag je met een software-tool maken, maar mag je ook met de hand tekenen en daarna inscannen. Indien je een software-tool gebruikt, maak er dan een screenshot van. De afbeelding zet je in de Word-document met de naam opdracht7-1.docx.

BENODIGDE KENNIS

- Ik geef een databasestructuur weer in een ERD

UITBREIDING (EXTRA)

Pas het ERD aan als je de klanten en de bestellingen in aparte tabellen wil registreren.

LES 7 - 2 UUR – 2 OPDRACHT 7-2

BANK BESTELLEN – DEEL 5

In deze opdracht maken we gebruik van het resultaat van opdracht6_1. Maak hiervoor een kopie van de gehele map en hernoem alleen de buitenste map naar opdracht7_2.

In het resultaat van opdracht6_1 wordt op de index pagina een bank geselecteerd, die op de bestelpagina getoond wordt. Er is ook al een bestelformulier waar de gebruiker naam en adres kan invoeren. Na het klikken op de bestelknop wordt er weer teruggesprongen naar de index pagina, waarop een bevestiging van de bestelling getoond wordt.

De bestelling is alleen nog niet opgeslagen in de database. Pas in deze opdracht de code van de bestelpagina zo aan dat de ingevoerde naam en adres met de bestelde bank (artikelnummer) in de databasetabel bestellingen wordt weggeschreven.

BENODIGDE KENNIS

- Ik maak een eenvoudige webapplicatie in C# (ASP.net) die gebruik maakt van een databasekoppeling
- Ik pas SQL-commando's toe voor database bewerkingen

BEOORDELING

De module wordt beoordeeld op de leerdoelen. Per opdracht staat (bij de opdracht) aangegeven om welk(e) leerdoel(en) het gaat. Een leerdoel van met de volgende waardering beoordeeld worden:

O = onvoldoende
M = matig (je doet iets nog niet helemaal goed)
V = voldoende
G = je doet het extra goed, bijvoorbeeld door een extra onderdeel te maken.

LEERDOELEN

De leerdoelen zijn:

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)
- Ik maak een eenvoudige webapplicatie in C# (ASP.net) die gebruik maakt van een databasekoppeling
- Ik lees in C# een array uit en manipuleer de gegevens binnen het array.
- Ik gebruik in C# de datumtypes indien nodig
- Ik verwerk in C# data die verzonden is via een formulier
- Ik geef een databasestructuur weer in een ERD
- Ik pas SQL-commando's toe voor database bewerkingen

WERKPROCESSEN

De werkprocessen zijn:

- B1-K1-W2 : Ontwerpt software
- B1-K1-W3 : Realiseert (onderdelen van) software

De werkprocessen zijn gekoppeld aan de leerdoelen. De volgende koppelingen zijn in deze reader opgenomen:

B1-K1-W2 :

- Ik geef een databasestructuur weer in een ERD

B1-K1-W3 :

- Ik maak een eenvoudige webapplicatie in C# (ASP.net)
- Ik maak een eenvoudige webapplicatie in C# (ASP.net) die gebruik maakt van een databasekoppeling
- Ik lees in C# een array uit en manipuleer de gegevens binnen het array.
- Ik gebruik in C# de datumtypes indien nodig
- Ik verwerk in C# data die verzonden is via een formulier
- Ik pas SQL-commando's toe voor database bewerkingen

Op de volgende pagina staat het gebruikte beoordelingsformulier afgedrukt.

BEOORDELINGSFORMULIER

Module					
Naam beoordelaar					
Naam student					
Datum					
Leerdoelen:	Beoordeeld bij opdrachten :	Punten:			
		O	M	V	G
Ik maak een eenvoudige webapplicatie in C# (ASP.net)	1_1 Jouw gegevens 2_1 Optellen 3_1 Bank bestellen 4_1 Bank bestellen deel 2				
Ik maak een eenvoudige webapplicatie in C# (ASP.net) die gebruik maakt van een databasekoppeling	6_1 Bank bestellen deel 4 7_2 Bank bestellen deel 5				
Ik lees in C# een array uit en manipuleer de gegevens binnen het array.	4_1 Bank bestellen deel 2				
Ik gebruik in C# de datumtypes indien nodig	5_1 Bank bestellen deel 3				
Ik verwerk in C# data die verzonden is via een formulier	3_1 Bank bestellen				
Ik pas SQL-commando's toe voor database bewerkingen	6_1 Bank bestellen deel 4 7_2 Bank bestellen deel 5				
Ik geef een databasestructuur weer in een ERD	7_1 ERD				

Opmerking: