Script ASP.NET MVC les

**Aansturing**

**1. Inleiding:**

- Studenten welkom heten  
 - Uitleggen waar we het over gaan hebben en hoe de les gaat verlopen

**2. Introductie/uitleg ASP.NET**

ASP.NET is een server side web applicatie framework voor web development. ASP staat voor Active Server Pages en hiermee kun je dynamische pagina’s produceren. Het is gemaakt door Microsoft om programmeurs de mogelijkheid te geven om dynamische websites, web applicaties en web services te maken. Het was released in januari 2002 met versie 1.0 van het .NET Framework en is de opvolger van Microsoft’s Active Server Pages (ASP) technologie, ook wel bekend als Classic ASP. ASP.NET is gebouwd op het Common Language Runtime (CLR), hierdoor kunnen programmeurs ASP.NET code gebruiken die ondersteund wordt door de .NET language. Wij gaan namelijk C# gebruiken.

**3. Uitleg MVC**

MVC staat voor Model View Controller. Het is een designpattern waarmee complexe applicaties leesbaarder worden en dat het veranderen van de user interface (View) niet direct invloed heeft op de code die het aanstuurt en de data die ervoor wordt gebruikt. Ditzelfde geld voor de Controller en het Model. Elk van deze componenten zou kunnen worden vervangen door een ander zolang alle naamgevingen en benodigde methodes maar aanwezig zijn. De functies van het Model de View en de Controller worden later in de presentatie behandeld met voorbeelden.

Hier zie je de workflow van een MVC framework:  
- Controller -> View

Op het moment dat je een URL aanvraagd word dit door zo’n controller afgehandeld. In de controller staat een action methode die refereerd naar de opgevraagde view. Deze action methode moet in ASP.NET MVC dezelfde naam hebben als de view, daarnaast moet de controller dezelfde naam hebben als de folder waar de view inzit.

Dus neem als voorbeeld test.com/Home/Dashboard, dan zoekt word er gezocht naar de HomeController met de Dashboard action.

Als je dat allemaal geen goede namen geeft, krijg je simpelweg een error dat de betreffende view niet gevonden kan worden.

- View -> Controller

Je vraagt bijv. in de view URL’s aan, je wilt een view partial inladen of je voert Ajax calls uit. Zoiets noemen we een User action. Deze actions worden op de view uitgevoerd en door de controllers afgehandeld.

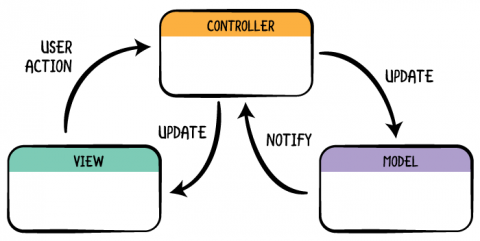
- Controller -> Model

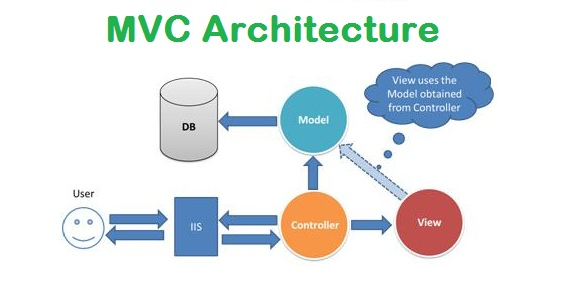
Het is logisch dat je bijv. data wilt ophalen uit een database, text, json of xml bestand. Dat soort functies doe je in het model en die data sla je ook op in het model, wat je weer meegeeft met de View.

- Model -> Controller

De model communiceert niet echt met de controller. Maar wel ongeveer op een manier als bijv. een log-in functie in de model met een boolean return waarde en aangeroepen in een if-statement in de action. Op zo’n manier kun je bijv. naar andere pagina’s navigeren op basis van true of false.

Hier zie je duidelijk geillustreerd hoe zo’n MVC framework nou communiceert met de verschillende lagen:





**4. Uitleg verschil tussen ASP.NET en ASP.NET MVC EIGENLIJK NOG TE KORT**

Tussen ASP.NET MVC en de andere ASP.NET ontwikkelmethodes zit een belangrijk verschil. ASP.NET MVC word gedreven door Models, Controllers en Views in een bepaalde workflow. Bij de meeste andere ontwikkelmethodes zoals Web Forms zit er een code behind file bij, zoals bij WPF. Dit betekend dat bij ASP.NET die view strikt verbonden is met de code behind file waaraan de view gekoppeld is. Bij ASP.NET MVC is alles apart van elkaar. Dit betekend dat je gemakkelijk views, controllers en models uit kan wisselen tussen elkaar zolang alle naamgeving en methodes maar aanwezig zijn.

**Zelfstanding trainen**

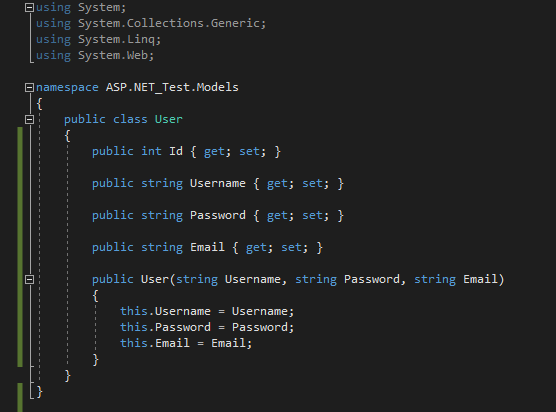
**Makkelijke oefeningen:**

**5. Model TE COMPLEXE UITLEG**

Definieert de representatie van de informatie waarmee de applicatie werkt. Aan ruwe gegevens wordt betekenis gegeven door relaties tussen de data en logica toe te voegen. De daadwerkelijke data wordt opgeslagen in een opslagmedium zoals een database. De applicatie zal de gegevens die gebruikt worden in het model ophalen en wegschrijven van en naar de dataopslag.

Een model kun je dus eigenlijk een beetje vergelijken met een tabel in een database. De properties die je er aan toe wijst zijn de kolommen de data die je hierin zet kun je dan opslaan in de database. In ASP.NET wordt de model geschreven in C# of VB.

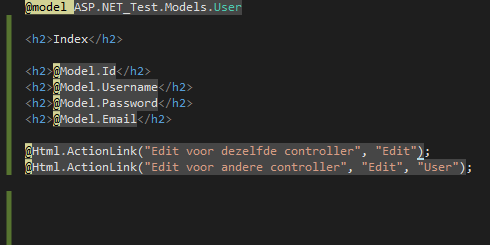
Voorbeeld in ASP.NET MVC met een model (User):



**6. View**

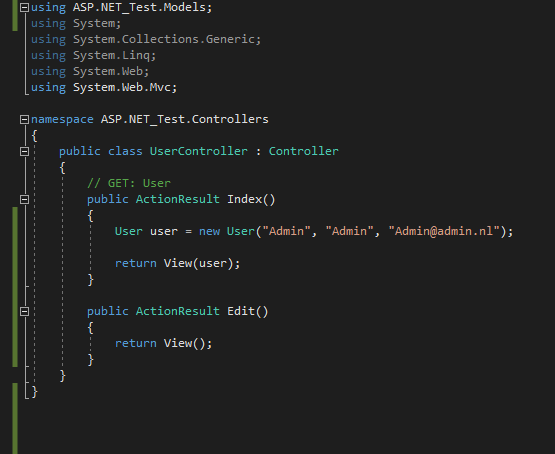
De view is de user interface van de web applicatie. Alle data die de gebruiker zal zien wordt in de view gezet. De view doet zelf geen verwerkingingen, zoals berekeningen en controles van de gegevens die getoond worden. Een view wordt in ASP.NET gewoon gemaakt door middel van HTML en word daarbij ondersteunt met CSS en Javascript.

In de view kan data aan de gebruiker worden getoond door middel van een model en de Razor syntax die bijv. uit de database gegevens heeft opgehaald. Wij komen later in de les terug op de Razor syntax en hoe deze werkt.

Voorbeeld in ASP.NET MVC met een view(Index, User):  


**7. Controller**

De controller verwerkt en reageert op request. Deze requests zijn meestal het gevolg van handelingen die de gebruiker uitvoert, zoals bijv. het navigeren naar een andere pagina. Aan een controller kunnen meerdere views worden gekoppeld door middel van meerdere actions toe te voegen. Een knop in de view kan gekoppeld worden aan verschillende methodes in de view liggend aan wat je wilt doen. Een controller kan dan verschillende ActionResults returnen waardoor er bijv. kan worden genavigeerd naar een andere view die gekoppeld is aan dezelfde controller of een andere controller. Met deze return values stuur je je gewenste model mee in de view.

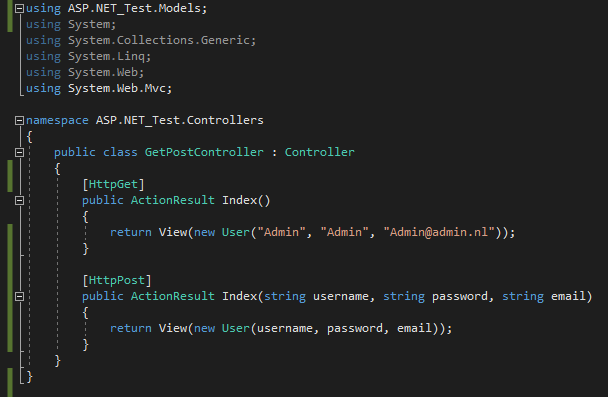
Voorbeeld in ASP.NET MVC met een controller (UserController):  


**8. Get en Post NOG TE KORT**

Wij gaan vandaag werken met twee soorten HTTP requests: Get en post. Get is voor als je data wilt opvragen en post is voor als je data wilt versturen. In ASP.NET kan je met behulp van attributes verschillende actions in de controllers zodanig afdwingen dat ze alleen get requests, of alleen post requests mogen verwerken.

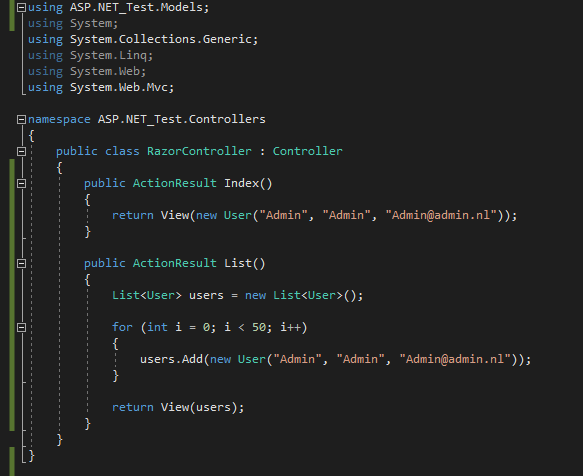
Voorbeeld in ASP.NET MVC met Get en Post:

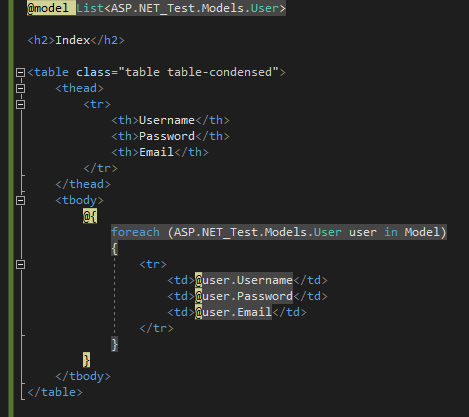
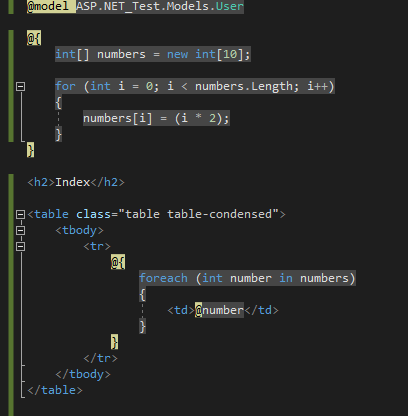




**9. Razor**

De Razor engine zorgt er als het ware voor dat je C# of Visual Basic code kan gebruiken in je view om daarmee bijv. het model te gebruiken, hiermee kan je dan bijv. een tabel vormen waar de data van het model in moet komen te staan. Je kan met Razor ook variabelen aanmaken in de view zolang de het @ teken en de scope operators { } eromheen staan. Al kun je functies met Razor gemakkelijk in de view uitvoeren is het over het algemeen niet aan te raden om complexere code in de view uit te voeren, deze complexere logica kun je beter aan de controller en het model overlaten. Als je dit wel zou doen ga je eigenlijk tegen het principe van MVC in.

Voorbeeld in ASP.NET MVC met Razor:  




**10. Html helpers**

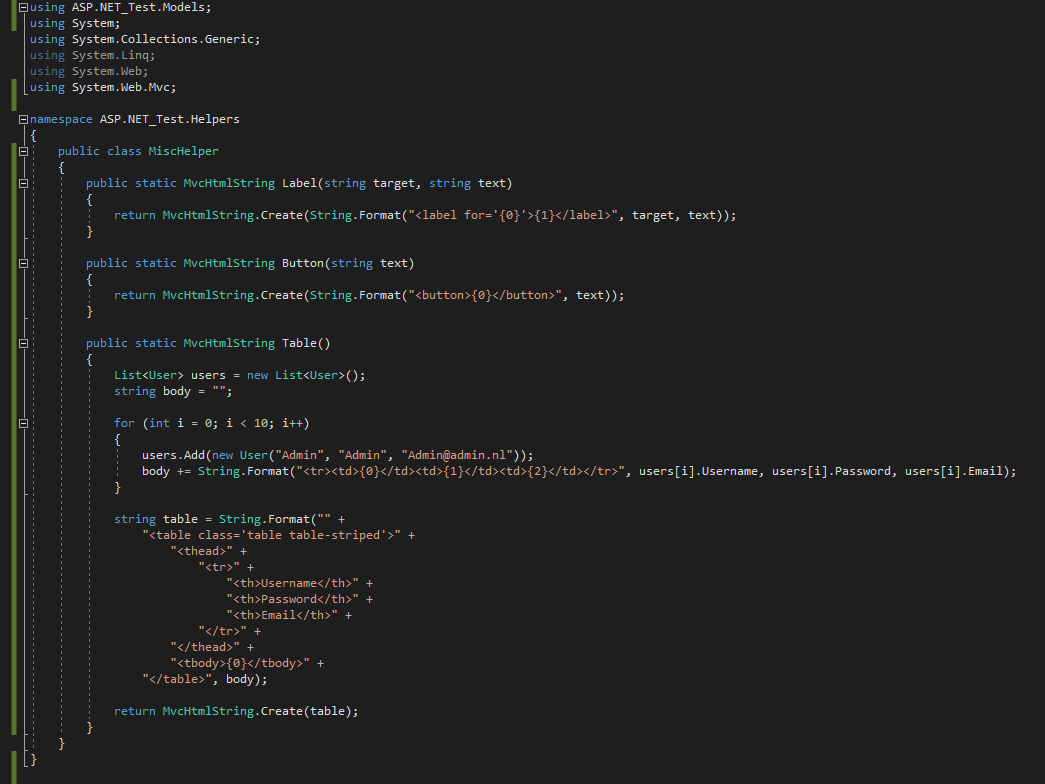
ASP.NET heeft standaard een HTML helper class in je project zitten. Deze HTML helpers helpen je bij het maken van de view door de hoeveelheid typen te verminderen. De HTML helper geeft op basis de parameters een HtmlString of MvcHtmlString terug wat automatisch door Razor word verwerkt als HTML, zou je hierbij een normale string terugkrijgen word het niet verwerkt als HTML en krijg je lelijke tekst op webpagina. Hierdoor zou je een knop, label, tabel en allerlei andere dingen kunnen maken.

Er zit in de HTML helper standaard een aantal methodes die je kan gebruiken. Maar je kan ook je eigen helper schrijven of je kan extensies toevoegen aan de bestaande HTML helper.

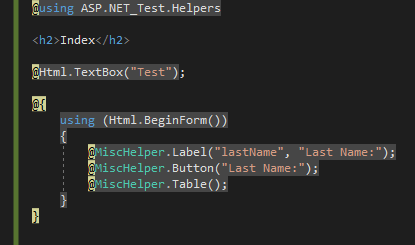
Lijst van een aantal Html helpers die standaard in ASP.NET MVC zitten:

Html.ActionLink() Html.BeginForm() Html.CheckBox() Html.DropDownList()  
Html.EndForm() Html.Hidden() Html.ListBox() Html.Password()  
Html.RadioButton() Html.TextArea() Html.TextBox()

Voorbeeld in ASP.NET MVC met een eigen HTML Helper:



Voorbeeld in ASP.NET MVC met de bestaande HTML Helper:

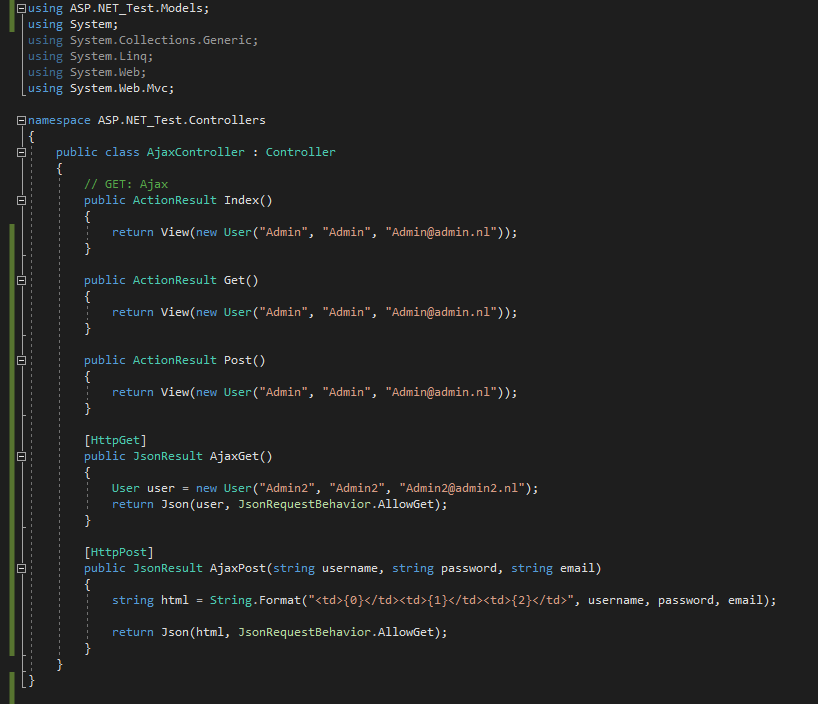


**Moeilijkere oefeningen:**

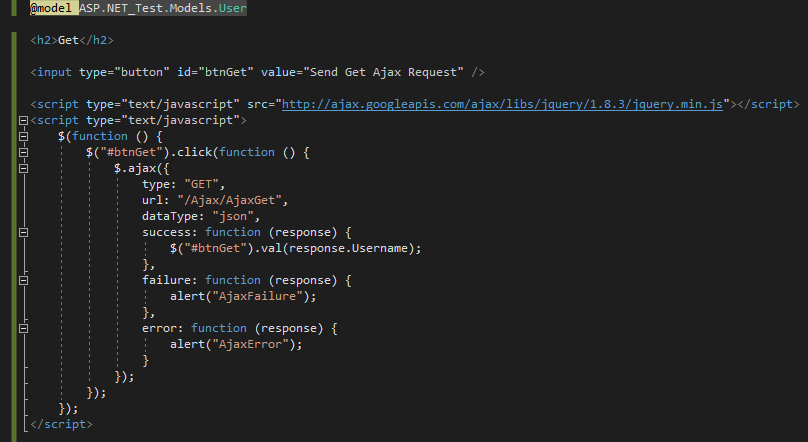
**11. jQuery Ajax (get en post) NOG TE KORT**

Met jQuery Ajax kun je calls maken via een Javascript functie of event om data uit de database te halen of een andere bepaalde functie uit te voeren zonder dat daarbij de pagina word ververst als het ware. Hierdoor kun je bijv. live feeds maken van aandelen of wat dan ook. Ajax maakt ook gebruik van HTTP requests zoals: Get en post. Voor een Ajax call kun je meerdere soorten datatypes ontvangen zoals tekst, HTML, Json enz.

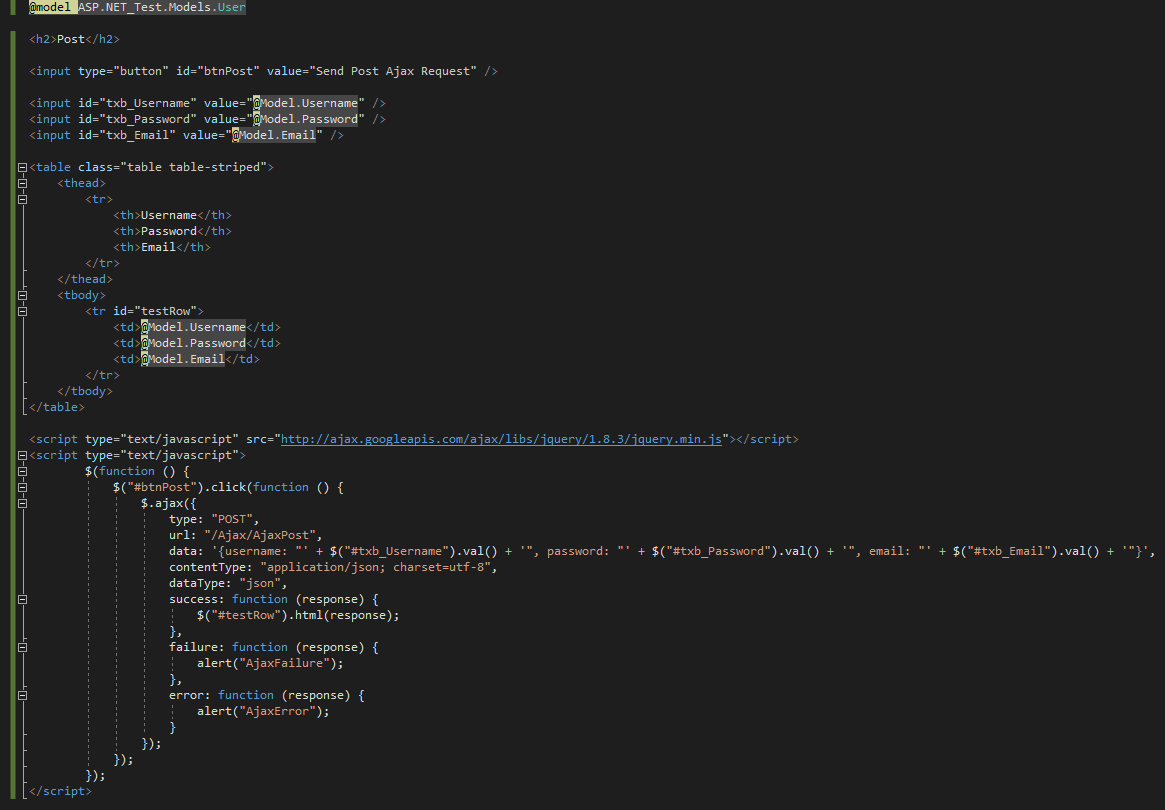
Voorbeeld in ASP.NET met jQeury Ajax (get en post):



Get:



Post:

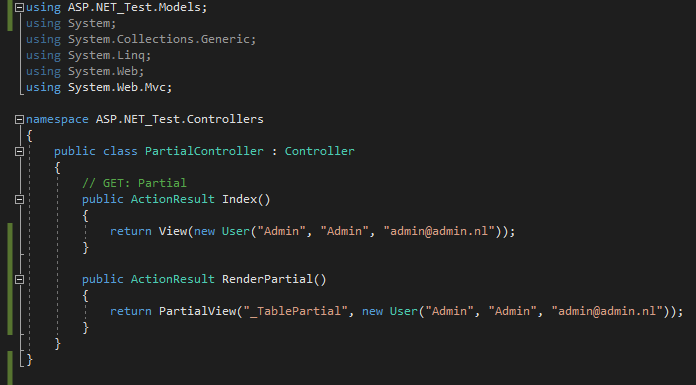


**12. Partial loading met en zonder Ajax**

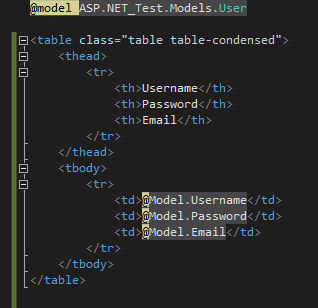
Een partial is in principe een stuk HTML die je kan laden in een andere pagina. Stel je hebt een index view en je wilt er een tabel in laden. Hiervoor zou je een aparte partial view kunnen maken waarin je de tabel vormgeeft. Later kun je dan in de index view deze partial view inladen. Door gebruik te maken van partial views worden je views overzichtelijker naar mate je meer functionaliteit aan een bepaalde pagina toevoegd. Zou je dit zonder partial views uitwerken dan wordt het al snel te veel en dus onoverzichtelijker. Je kan een partial view inladen door mate van een HTML helper of jQuery Ajax.

Met Ajax kan je een action opvragen wat als return waarde een PartialView heeft. Jquery krijgt deze return waarde als HTML binnen, waarmee je dus gemakkelijk de HTML van bijv. een div kunt invullen met de de HTML.

Voorbeeld in ASP.NET MVC met Partials (met en zonder Ajax):





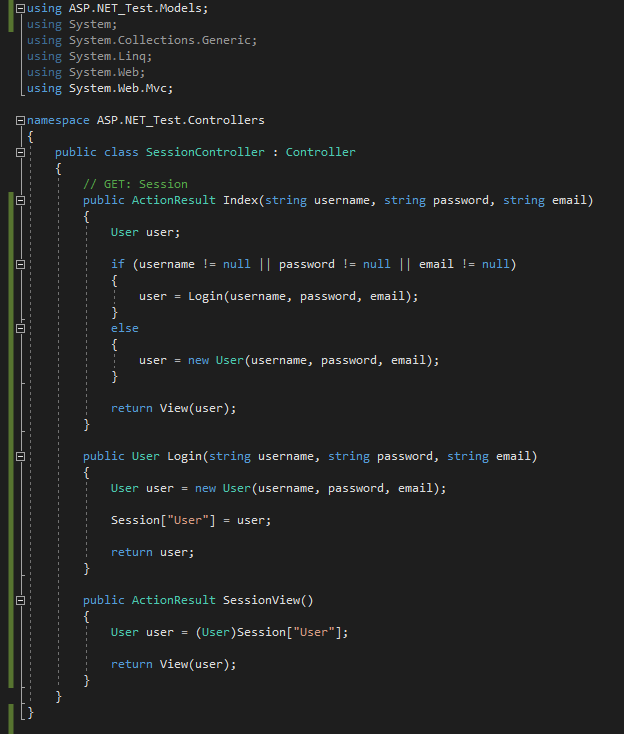


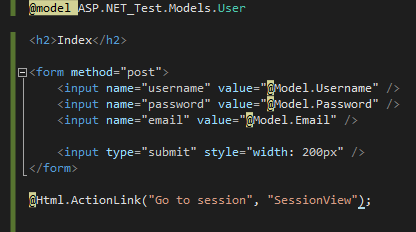
**13. Sessions**

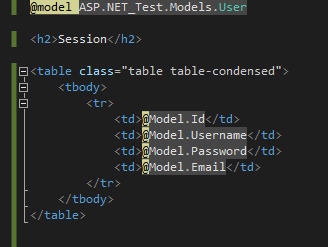
ASP.NET beschikt over session state, waarmee je variabelen kan opslaan en kan opvragen voor later gebruik. Deze variabelen zijn server side opgeslagen in het geheugen en worden aangesproken door middel van een session key. Deze session key word opgeslagen in een session cookie en is dus altijd identiek per gebruiker.

HTTP is een staatloos protocol. Dit betekent dat de webserver elk HTTP request behandeld als een onafhankelijke request. De server heeft hierbij geen weten van waardes van variabelen die zijn gebruikt in een vorige request. ASP.NET session state indentificeerd requests van dezelfde browser als een session, en biedt een mogelijkheid om waardes van variablen te gebruiken in de tijd dat de session actief is. De session state is standaard geactiveerd voor elke ASP.NET MVC applicatie.

In een session kun je alle soorten .NET-framework datatypes gebruiken. Deze variabelen worden opgeslagen in een SessionStateItemCollection en kan aangeroepen worden door de HttpContext.Session property.

Voorbeeld in ASP.NET MVC met Sessions:  






**Terugkoppeling**

Feedback geven over hoe wij de les vonden gaan en vragen aan de studenten hoe zij de les ervaarden.

**Bronnen:**

<https://en.wikipedia.org/wiki/ASP.NET>   
<https://nl.wikipedia.org/wiki/Model-view-controller-model>   
<https://www.codeproject.com/Articles/668182/Difference-betweeen-ASP-NET-WebForms-and-ASP-NET-M>   
<https://www.quora.com/What-is-the-difference-between-ASP-NET-MVC-and-ASP-NET>   
<https://docs.microsoft.com/en-us/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c>   
<https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/views/creating-custom-html-helpers-cs>   
<https://www.aspsnippets.com/Articles/ASPNet-MVC-jQuery-AJAX-and-JSON-Example.aspx>   
<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial>   
<https://msdn.microsoft.com/en-us/library/ms178581.aspx>

<https://stackoverflow.com/questions/623815/what-is-the-difference-between-a-session-and-a-cookie>