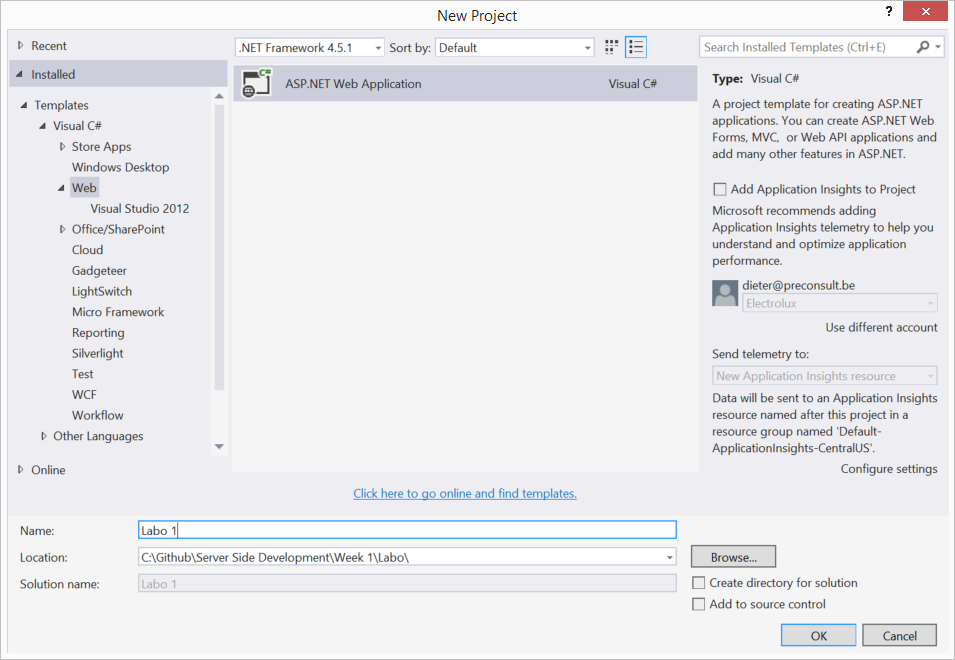
# ASP.NET MVC Labo

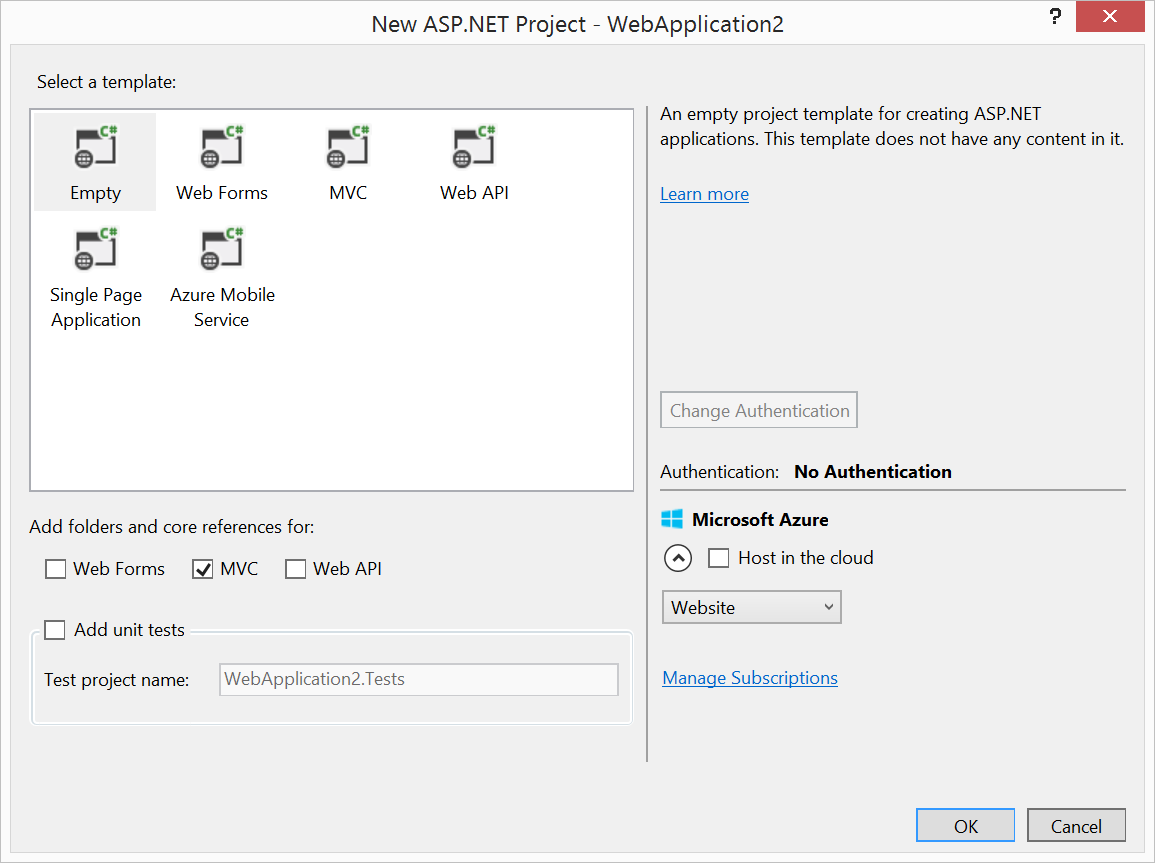
## Doelstelling:

* Kennis maken met ASP.NET MVC
* Een eerste MVC project aanmaken en verkennen

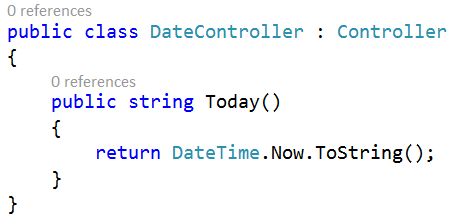
## Oefening 1: Kennismaking (Samen)

We maken een nieuwe ASP.NET MVC Applicatie via Visual Studio.

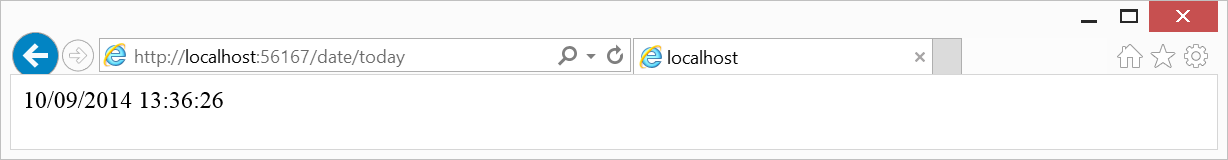




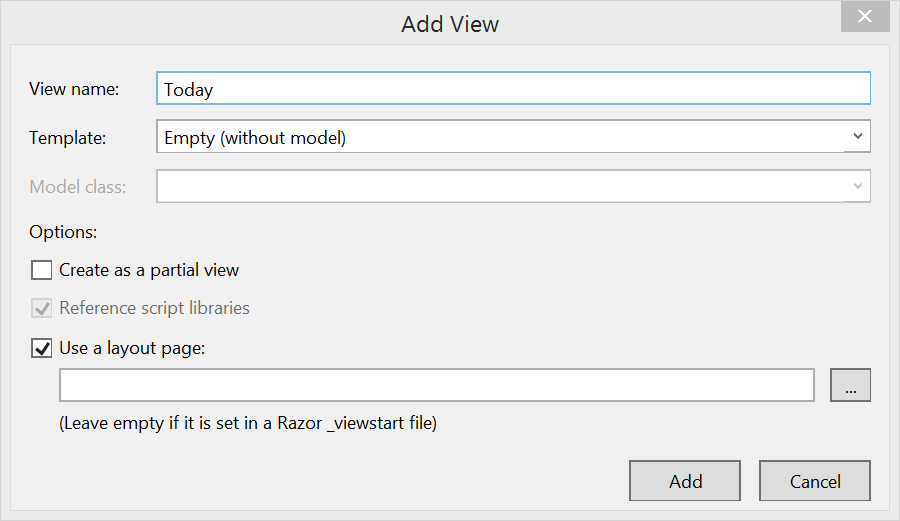
We kiezen in het eerste labo voor “Empty Project” en vinken enkel MVC aan. Voeg via rechtermuis op de map “Controllers” een file “DateController” toe aan de map controllers. In deze controller voegen we een methode (Action) Today() toe. Deze zal een string terugkeren.



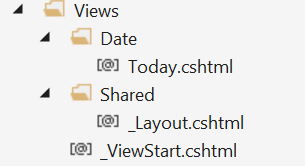
Als we nu surfen naar <http://localhost:56167/date/today> dan krijgen we volgende te zien. Hierbij hangt het poortnummer 56167 natuurlijk af van het poortnummer dat webserver je toekent. Let bovendien op de conventie hoe eerst de controller (date) en daarna de actie (today) zorgen voor een juiste routing.



Als we kijken naar de gegeneerde HTML dan zien we dat deze niet aanwezig is. De controller zal immers enkel de DateTime string terugkeren. Om een HTML terug te keren moeten we ook een view aanmaken en terugkeren. Klik in de ActionMethod Today() via rechtermuis en kies “Add View”.

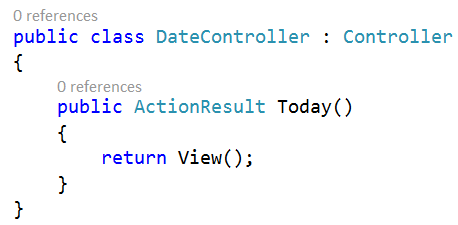


Als we nu kijken in het mapje Views dan ziet dit er als volgt uit:

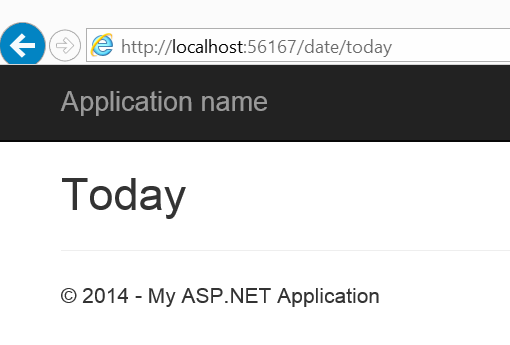


De view Today.cshtml bevat de html die zal worden weergegeven als we naar de action Today() surfen. Naast de view Today is er ook een map Shared met daarin \_Layout.cshtml. Deze file is een soort masterpage met html code die over verschillende pagina’s gebruikt zal worden.

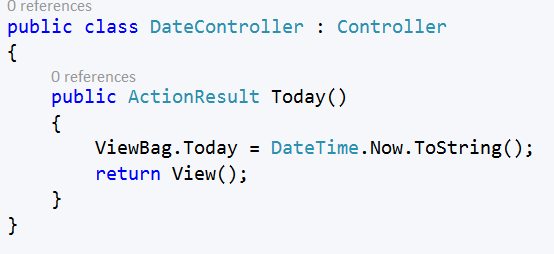
We moeten echter nog de actionmethod in de controller aanpassen. Deze method moet de View terugkeren. We wijzigen eerst de returnvalue van de method van string naar ActionResult. Daarna voegen we return View() toe. Bij het uitvoeren zal de actionmethode “Today” zoeken naar een view met dezelfde naam en deze terugkeren. Het zoeken naar een view met dezelfde naam is één van de conventies binnen het MVC patroon.



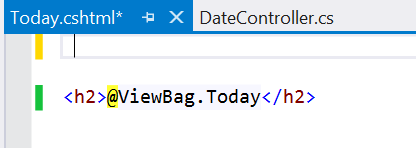
Het resultaat ziet er als volgt uit:



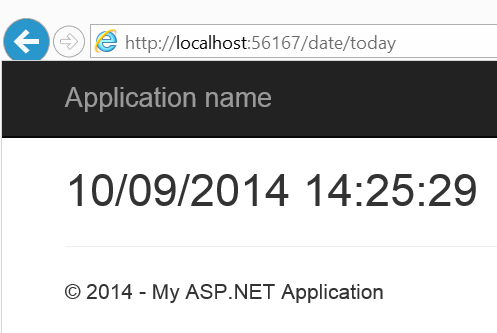
We zien echter nog geen datum staan. We moeten deze nog doorsturen naar de View zodat deze opgenomen kan worden in de teruggekeerde HTML.



Er zijn verschillende manieren om te communiceren tussen de controller en de view. Één van de manieren om te communiceren van controller naar view is de ViewBag. In die ViewBag kunnen we dynamisch properties toevoegen waarin we data opslaan. Deze properties kunnen we daarna terug uitlezen in de view. In bovenstaande code voegen we een property Today toe aan de ViewBag. In deze property slaan we de huidige datum op. In de view moeten we de ViewBag terug uitlezen. Via het @ symbool (Razor syntax) krijgen we toegang tot verschillende properties zoals de ViewBag. Daar kunnen we terug de property opvragen en afdrukken tussen HTML Tags.



Het resultaat ziet er als volgt uit:



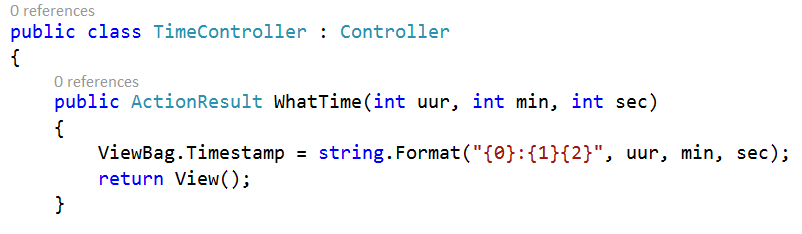
Voeg nu zelfstandig toe:

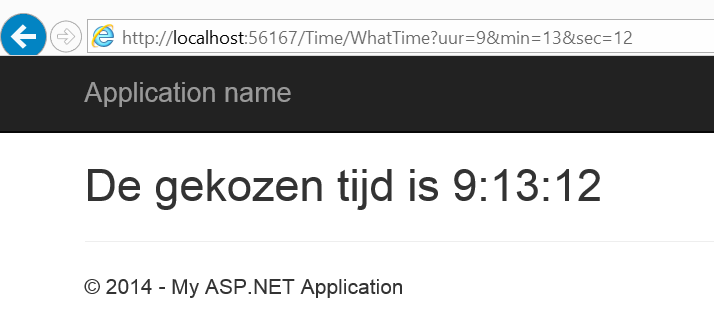
* ActionMethod Yesterday toevoegen die de datum van gisteren zal weergeven
* ActionMethod Tomorow toevoegen die de datum van morgen zal weergeven
* Zorg ervoor dat indien je geen url opgeeft hij default Today() neemt .tip: check RouteConfig.cs in de map App\_Start)
* Date nemen als controller & Action => Today nemen

## Oefening 2: De QueryString (Samen)

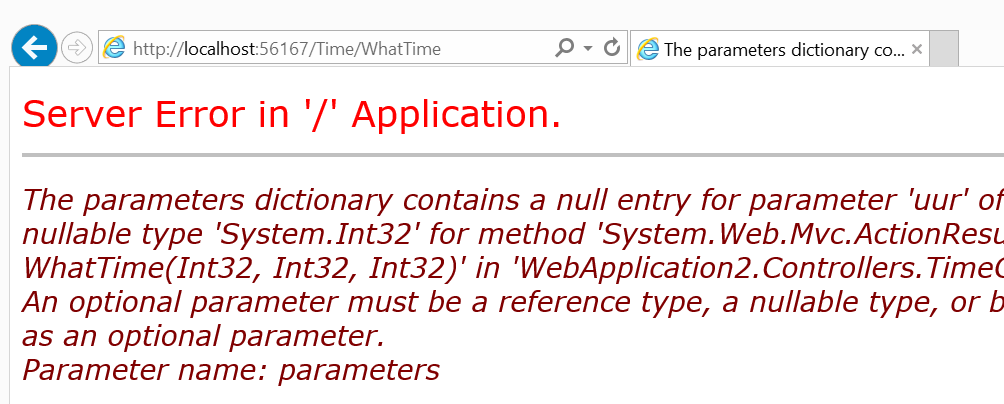
Voeg een TimeController toe aan je project in de map controllers. In de TimeController maak je een actionmethod WhatTime(int uur, int min, int sec) aan. Binnen deze methode vraag je volgende querystring waarden op:

* uur
* min
* sec

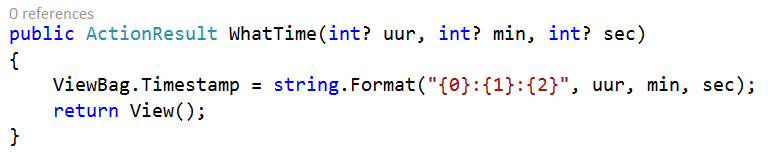




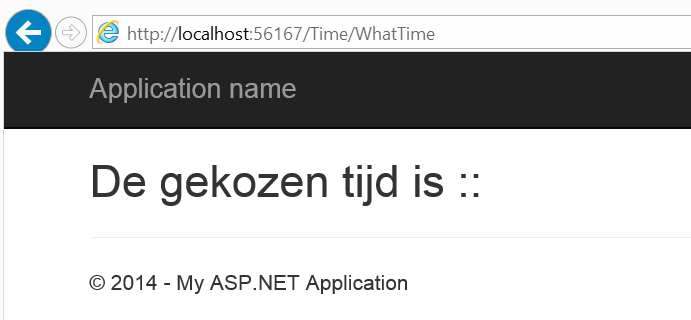
Als we nu opstarten zonder parameters in de url krijgen we een foutmelding aangezien we niks meegeven in de querystring.



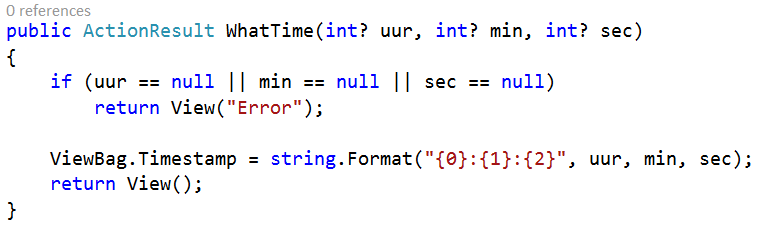
We kunnen dit opvangen door van de parameters nullable types (<http://msdn.microsoft.com/en-us/library/1t3y8s4s.aspx>) te maken. Dit doen we door na het datatype van de parameter een vraagteken te plaatsen. Nu kan een integer ook de waarde null bevatten.

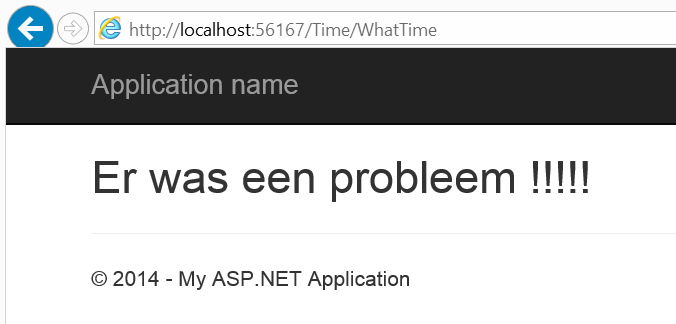


Als we dit nu opstarten dan krijgen we geen foutmelding maar ook geen tijdstip. We moeten er dus voor zorgen dat indien één van de parameters null is we een mooie foutmelding tonen.

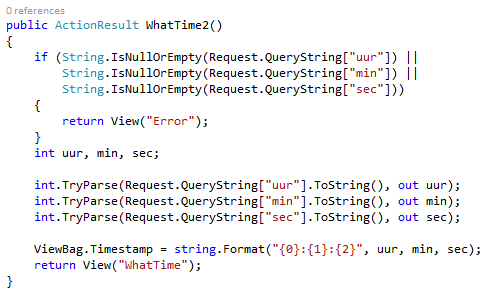


Aangezien de parameters nu nullable zijn, kunnen we controleren of ze een waarde bevatten door te testen op null. Indien één van deze parameters null is keren we een “Error” view terug. Deze view plaatsen we in de map Shared. Op deze manier kunnen we de view hergebruiken vanuit meerdere actions.



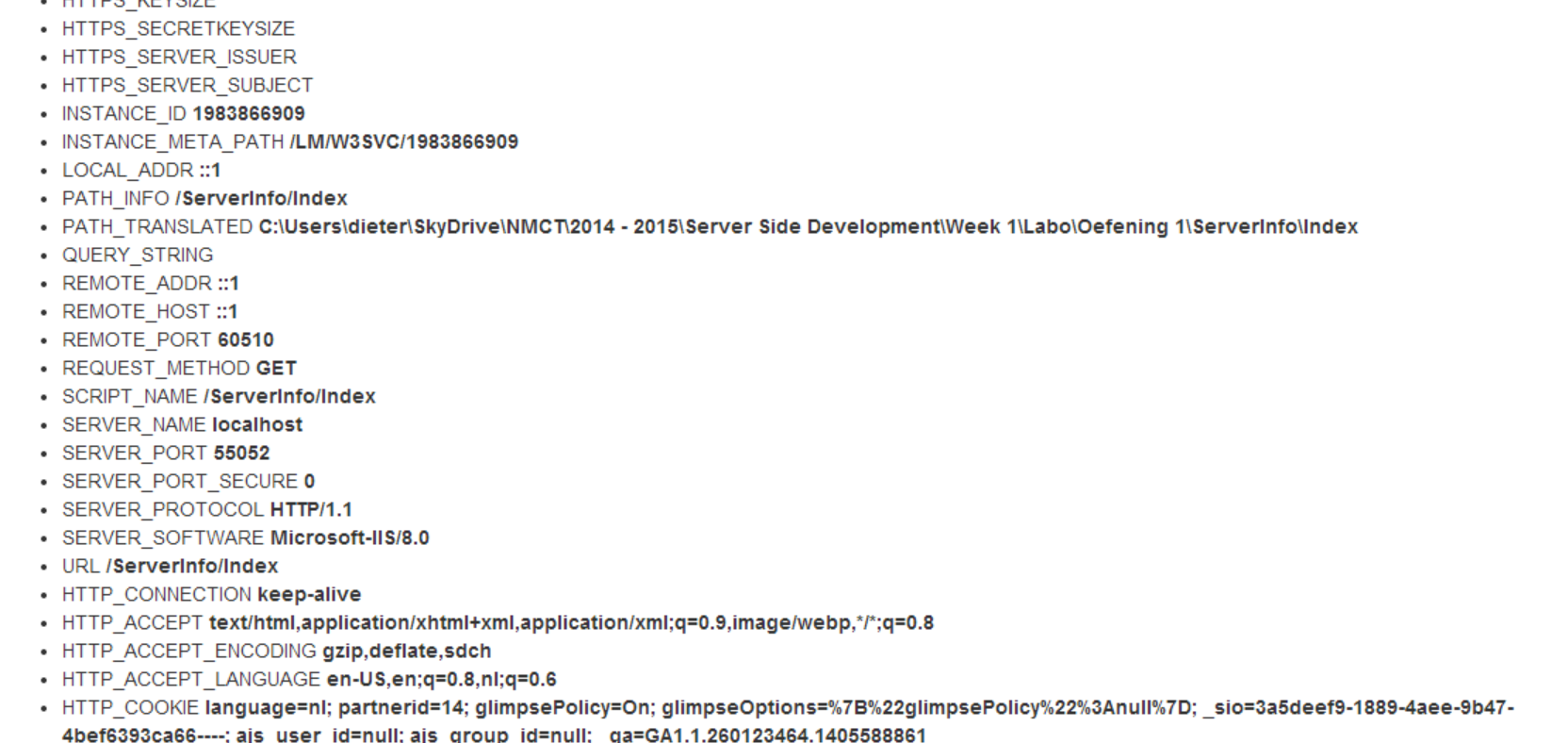


Er is nog een tweede manier om querystring waarden op te vragen, voeg een methode WhatTime2 toe. Deze tweede method maakt gebruikt van het Request object. Via dit object kunnen we allerlei waarden opvragen. Request.QueryString heeft ons toegang tot de key/value dictionary van de querystring. We moeten echter eerst controleren of de key niet leeg is. Daarna kunnen we via TryParse de juist waarde uitlezen.



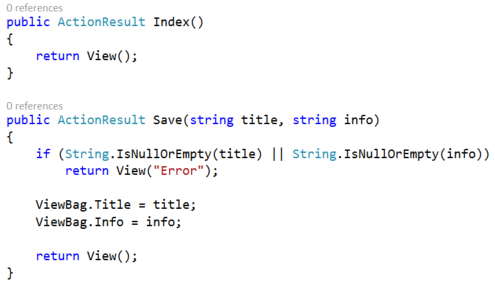
## Oefening 3: ServerVariables (Zelfstandig starten)

Maak een controller ServerVariablesController aan. Zorg ervoor dat alle server variabelen afgedrukt worden in de view (tip: Request.ServerVariables). Het resultaat ziet er ongeveer als volgt uit (**de waarde van de variabele moeten in het VET staan**): Tip: maak gebruik van een model indien nodig.

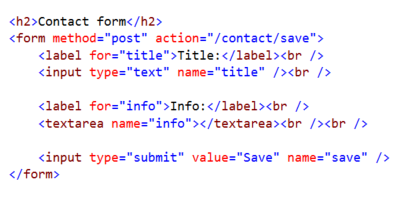


## Oefening 4:

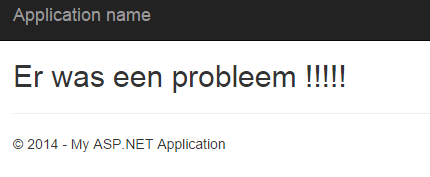
Waar we tot nu toe de ganse tijd met een http GET werkten, gaan we in deze oefening de http POST introduceren. Een http GET gebruiken we om een resource op te vragen en eventueel via de querystring parameters mee te geven. Een POST gebruiken we om data via de http body door te geven aan de server. Maak een nieuwe controller ContactController aan. In deze controller gaan we minstens twee actionmethods hebben, een Index() (http GET) die de form zal terugkeren en een Save() (http POST) die we gebruiken om de formwaarden op te slaan.



De view ziet er als volgt uit: zoals gezien in PHP moeten we ook een method en action opgeven.



Voeg nog zelf een tweede view toe waarin we de ingevoerde gegevens weergeven als bevestiging Wanneer we niks invullen dan moeten we dit opvangen in de controller en moeten we een “Error” View afdrukken.

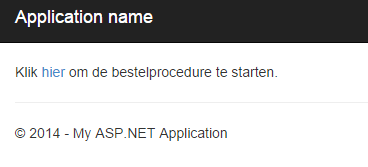


## Oefening 5: Order formulieren

Maak een eenvoudige applicatie voor het bestellen van een tablet. De applicatie bestaat uit 4 schermen. Tip: maak gebruik van hidden fields om gegevens van de view naar de controller over te brengen en zo geen waarden te verliezen.



### Start scherm:



### Stap 1:



### Stap 3:

Keuze uit: iPad Mini, iPad Air, Nexus 7, Surface 2



### Stap 4

