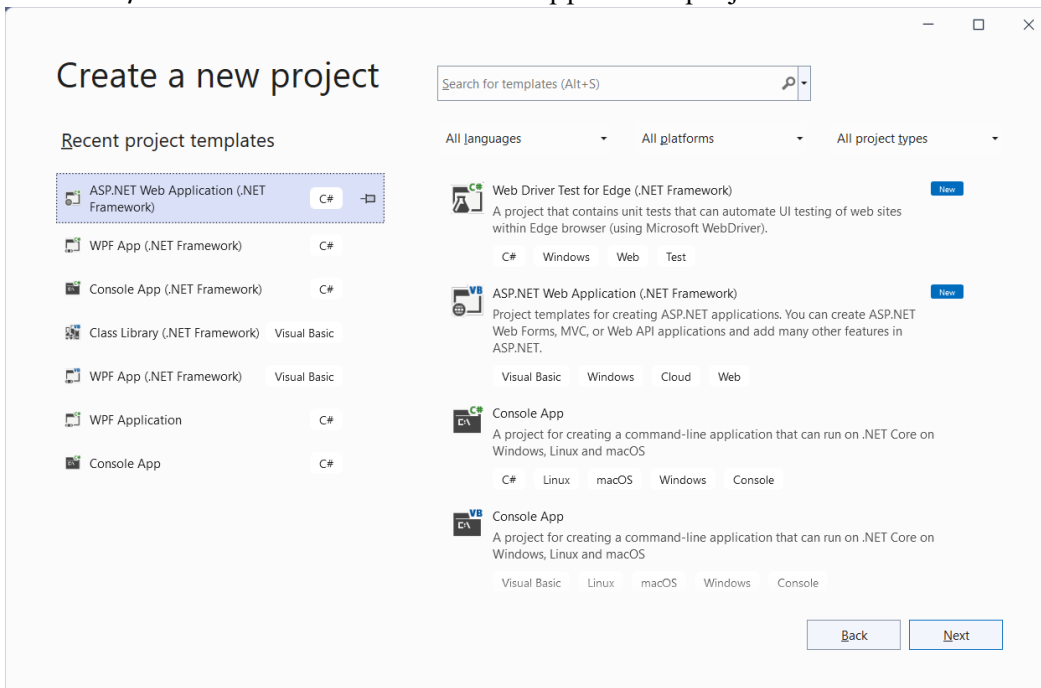
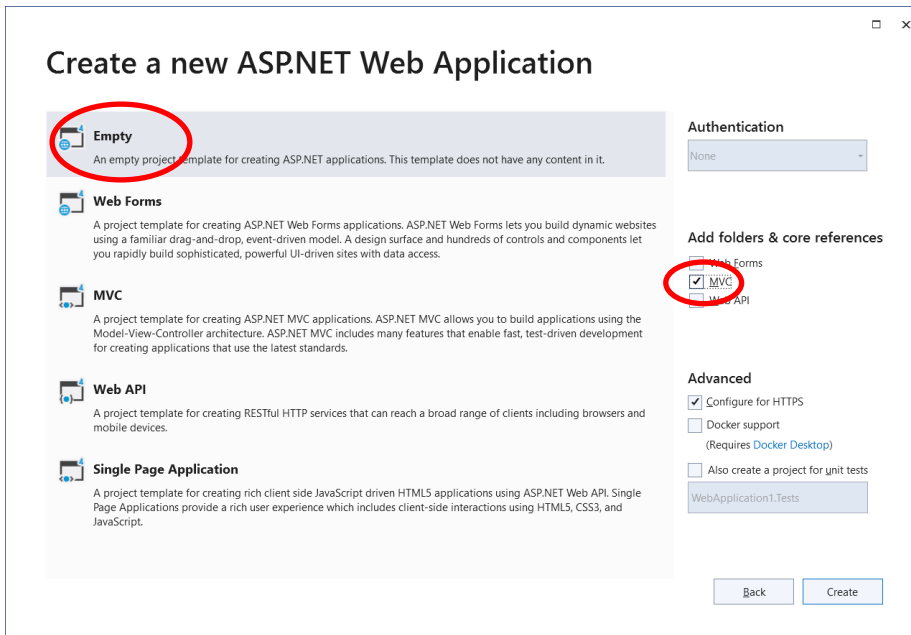


11.1

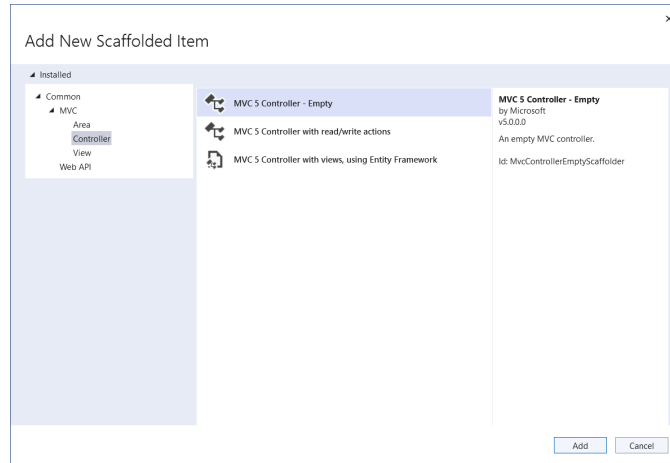
To start you must create a ASP.NET Web Application project in Visual Studio.



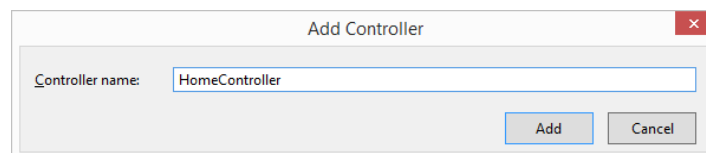
Select An Empty MVC project:



Add a new empty MVC 5 Controller:

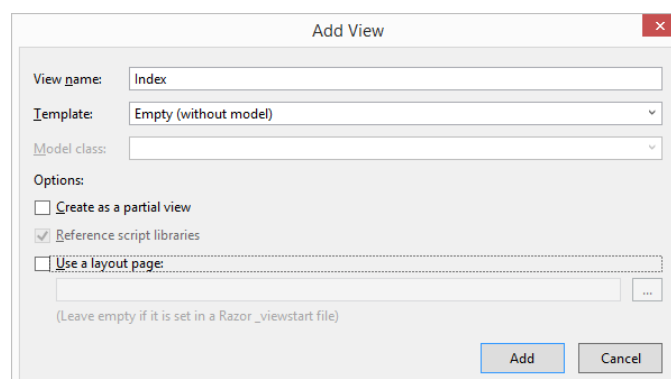


and give it the name *HomeController*:



Run the code. Investigate and explain the error message you get. In which folders do the MVC Framework try to find the View to use? Which file types does it search for?

Now, create a View for the `HomeController` by right clicking somewhere inside the `ActionResult` method `index()` and select **Add View...**:



Leave “Use a layout page” unchecked.

Write code in the view to display the text “My first view”, formatted as you like (using html)

11.2

Go to the action method `Index()` inside the `HomeController` and declare three variables in its code block:

name of type string
age of type int
birthday of type DateTime

Assign values to the variables. It could be your own data or someone else's:

```
name = "Peter";  
etc.
```

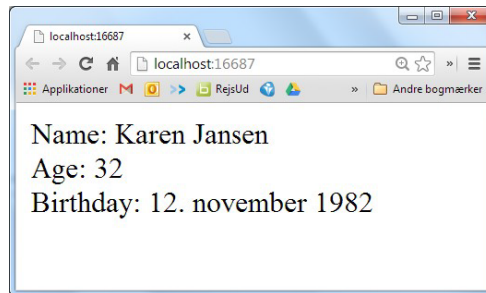
Declare ViewBag properties and assign the variables to them:

```
ViewBag.name = name;  
ViewBag.age = age;  
ViewBag.birthday = birthday;
```

Go to the View.

Insert the ViewBag data into the view in order to generate a web page similar to this:

Tip:



- Look up the documentation (<http://msdn.microsoft.com/en-us/library/system.datetime.aspx>) for the `DateTime` structure to see which methods to use in order to display the date the way you want.

Lav om, så du ikke bruger ViewBag, men Model til at overføre data fra controller til view.

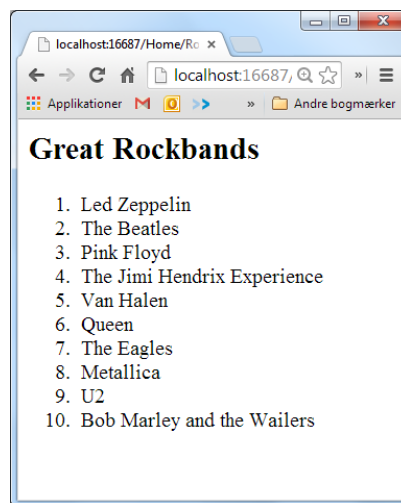
11.3

Create a new controller with the name `RockbandsController()`.

Inside the `Index` action method, declare a string array with 10 elements and assign a string with the name of your favourite rock bands to each element or other elements. Assign the `rockbands` string array to a `ViewBag` property or a model

Create a new View for the `Rockbands` Controller and use a *foreach*-loop inside the View to generate an ordered list of band names.

Run the example :



Notice:

1. You must write `[webhost]/rockbands` in the browser address bar in order to execute the Action Method `Rockbands` of the `Home` controller.

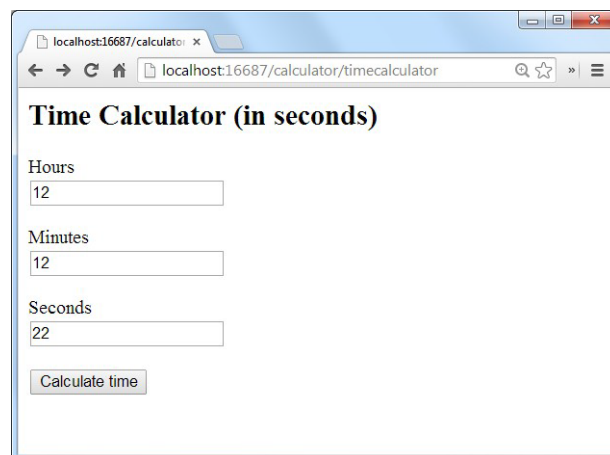
11.4

Create a new Controller named `CalculatorController`. Add an Action Result Method, `TimeCalculator` and create a new View (`TimeCalculator.cshtml`) for that controller.

The view must generate an html form, which enables the user to type in time in hours, minutes and seconds. You can write the form inside the View with plain html or you can use the MVC framework's built in Html helper methods, like:

```
@using (Html.BeginForm()) {  
<p>  
@Html.Label("Hours") <br />  
@Html.TextBox("Hours")  
</p>  
<p>  
    @Html.Label("Minutes") <br />  
    @Html.TextBox("Minutes")  
</p>  
<p>  
    @Html.Label("Seconds") <br />  
    @Html.TextBox("Seconds")  
</p>  
<p>  
<input type="submit" value="Calculate time">  
</p>  
}
```

Call the controller to test the view:



The screenshot shows a web browser window with the address bar displaying `localhost:16687/calculator/timecalculator`. The page title is "Time Calculator (in seconds)". The form contains three text input fields labeled "Hours", "Minutes", and "Seconds". The "Hours" field contains the value "12", the "Minutes" field contains "12", and the "Seconds" field contains "22". Below these fields is a "Calculate time" button.

Activate source code view in the browser and inspect the html of the form. Which controller and action method is called? What are the names of the form fields? Do they have an id?

The next step is to create a Controller which handles the form input when users click the submit button ("Calculate time"). This will lead us to the following:

When you click the button, the user will be presented for a result of the calculation:

12 hours + 12 minutes + 22 seconds = 43.942 seconds

By setting a special Action Method *attribute* and having different parameters for the Action Methods, you can use the same action method *name* for both methods:

```
[HttpGet]
public ActionResult TimeCalculator() {
...
}

[HttpPost]
public ActionResult TimeCalculator(FormCollection formCollection) {
...
}
```

These attributes determines which version of `TimeCalculator` is called. When the browser URL requests the page a get request is sent to the server, and it is then the first version of `TimeCalculator` that is called. When the user on the other hand clicks on the submit button a post request is sent to the server, and the second version of `TimeCalculator` which is called. To have access to form data from the action method the `FormCollection` is given as parameter.

Inside the block code of the `TimeCalculator` you must extract the value of the form. You can do that by referencing each of the form elements by its name, like:

```
formCollection["Hours"]
```

In order to calculate the values you must convert them to integers. You can use the `Convert` class for that. It as has a static method `ToInt32` that convert strings, as well as other data types, to integers:

```
int hours = Convert.ToInt32(formCollection["Hours"]);
```

When you have converted the form's input fields to integers, the next task is calculate the total number of seconds. The `TimeSpan` class is an excellent choice for that:

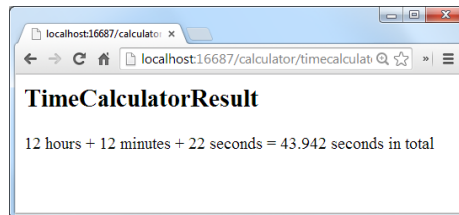
```
TimeSpan ts = new TimeSpan(0, hours, minutes, seconds);
double total = ts.TotalSeconds;
```

The above code gives you a new `TimeSpan` object `ts`, with the values given as arguments when the object is created. Notice that the `TimeSpan` property `TotalSeconds` returns a double. That's the reason why the variable `total` is declared as a double.

To proceed you must now have variables holding values for `Hours`, `Minutes`, `Seconds`, and `Total` and assigned values to these variables. To give access to these variables from a view you must save them as `ViewBag` properties:

```
ViewBag.Hours = hours;  
...
```

The last step in this exercise is to create a view for displaying the result:



You already have a View, `TimeCalculator` with the name of the action method. Instead of calling the default View it's possible to call a View with a different name by giving the name of the View as argument when you call the view:

```
return View("TimeCalculatorResult");
```

Create the View `TimeCalculatorResult` and display the result similar to the screenshot above.

Modify the default view to show both the form and the result, and call the default view from inside the `TimeCalculator` Action Method.

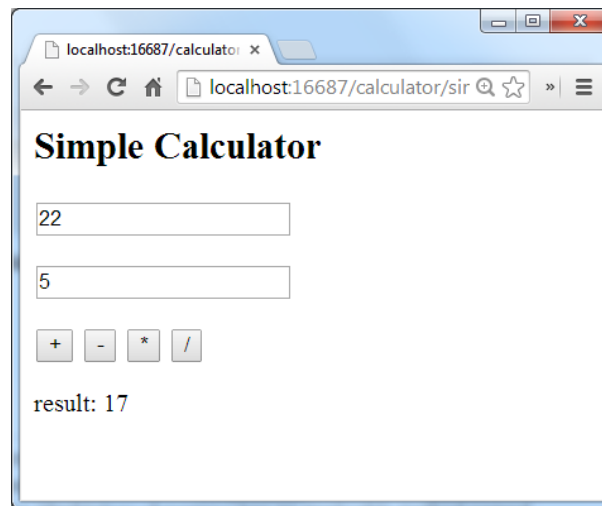
```
return View();
```

11.5 Lav en MVC klient til din tre lags applikation

Lav en MVC klient til din trelags applikation. Brug så meget du kan af det, du har lært om MVC.

11.6 (Optional)

Add a new Controller `SimpleCalculator` with an Action Method and View in order to build a basic calculator as indicated in the screenshot below:



1. The numbers typed into the form must be integers
2. In order to handle the result correctly for division the result variable must be of type `double`
3. If you give all the submit buttons the same name, like for example `operator`, and different values (like "+", "-", "*", and "/"), you can read value of the operator by referencing `FormCollection["operator"]`.
4. Use the `switch` control to determine which operator to use in the calculation. You can see the documentation for `switch` at <http://msdn.microsoft.com/en-us/library/06tc147t.aspx>
5. To display the numbers typed in after the calculation you can save them to a `ViewBag` property as you convert the number to a string:

```
ViewBag.Number1 = number1.ToString();
```

6. In the view, you can save values inside the input fields by giving an object as second parameter for the `TextBox` method of `Html` helper class:

```
@Html.TextBox("Number2", (string)ViewBag.Number2)
```

Notice that `ViewBag.Number2` is casted to a string in order to convert it to a strongly typed object.

Test that your application works correct. Does a division by zero raise an error? If so, how do you fix it?