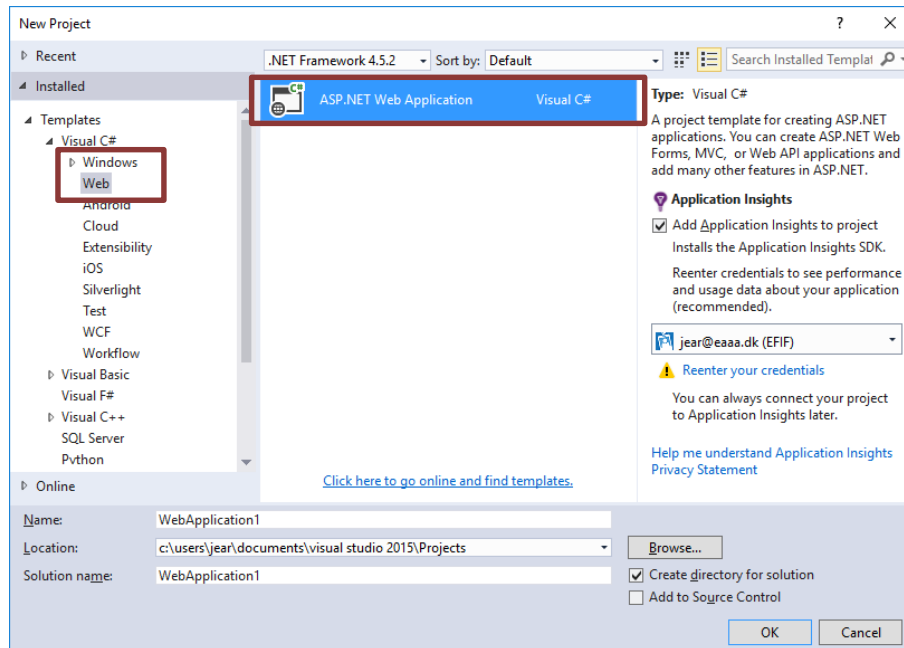


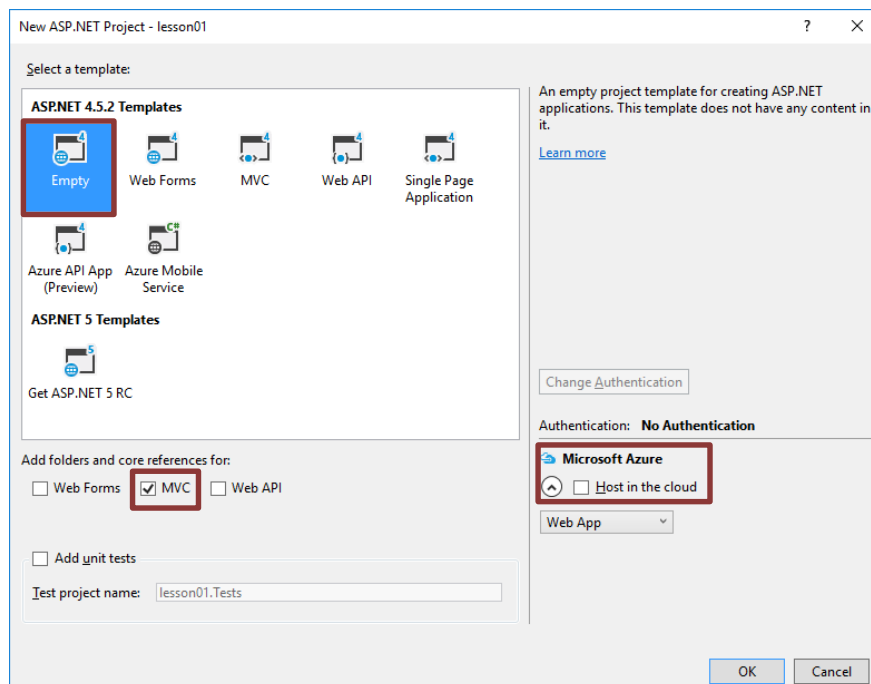
Lesson 1

Exercise 1

To start you must create a new ASP.NET Web Application project in Visual Studio (e.g. "Lesson01").

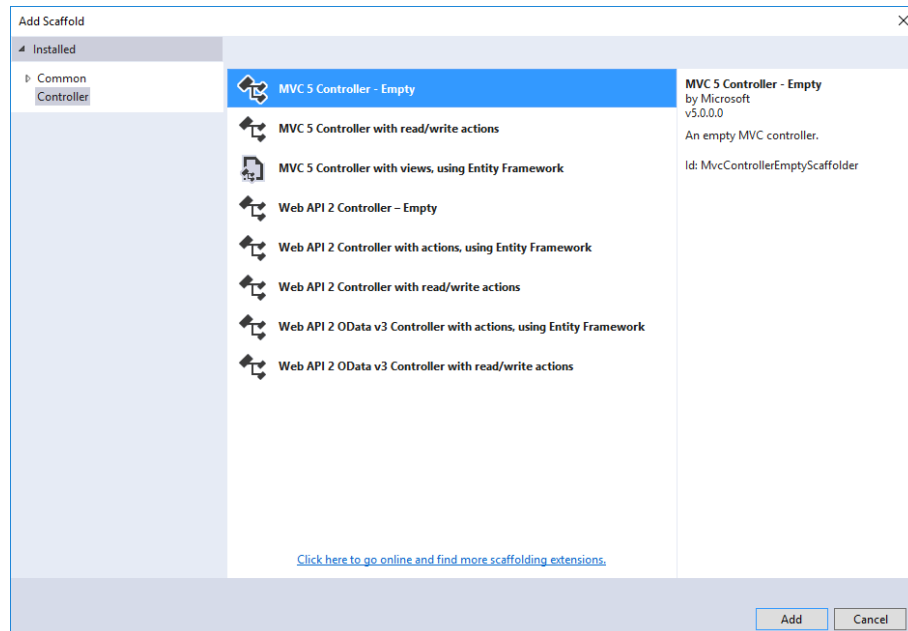


Select an Empty MVC project:

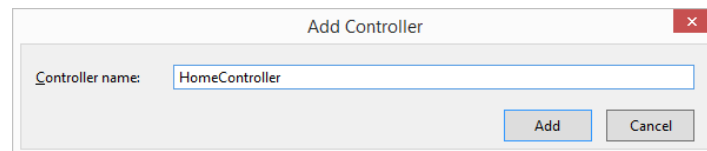


Check **Add folders and core references for MVC** and do not **Host in the cloud**.

When the project I created, you must add a new empty MVC 5 Controller to the project. Right click the *Controllers* folder in the **Solution Explorer** and choose **Add Controller...** :



and give it the name *HomeController*:



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

A variable called `name` of type `string`

A variable called `age` of type `int`

A variable called `birthday` of type `DateTime`

Assign values to the variables:

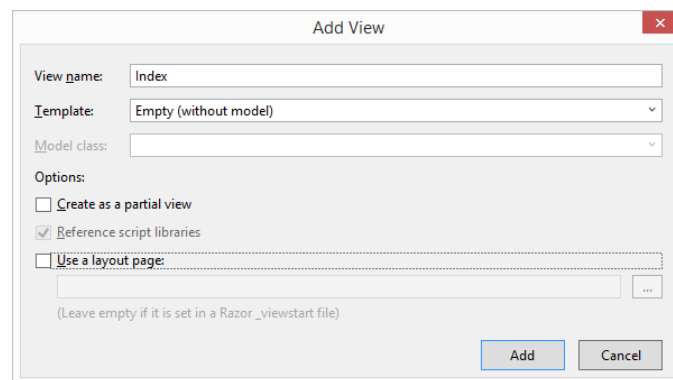
```
name = "Peter";  
etc.
```

Declare `ViewBag` properties and assign the variables to them:

```
ViewBag.Name = name;  
ViewBag.Age = age;  
ViewBag.Birthday = birthday;
```

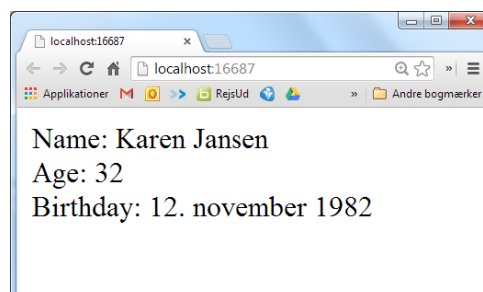
Run the code (Ctrl+F5)! Investigate and explain the error message you get. In which folders do the MVC Framework try to find the View to use? Which file types does it search for?

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.

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



Tip:

- To help you getting started with Razor syntax [C# Razor Syntax Quick Reference](#) is a helpful resource.
- 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.

Exercise 2

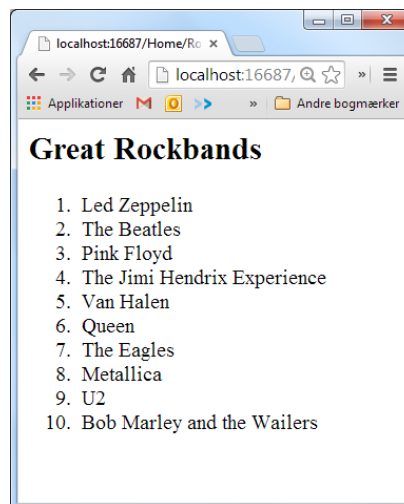
Create a new controller with the name `RockbandsController()`.

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. Assign the rockbands string array to a `ViewBag` property:

```
ViewBag.Rockbands = rockbands;
```

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 (Ctrl+F5):



Tip:

- For instructions of how to declare a string array, see <http://msdn.microsoft.com/en-us/library/0a7fscd0.aspx>
- The syntax of the foreach loop is:

```
foreach (string name in stringArray)
{
    // ...
}
```

where *stringArray* is the name of the array and *name* is the variable name of each string element in inside the array. You can use the *name* variable inside the code block whenever you want to refer band name. The *name* variable can have any name that you want. In this exercise, it could for example be `rockband`.

Notice:

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

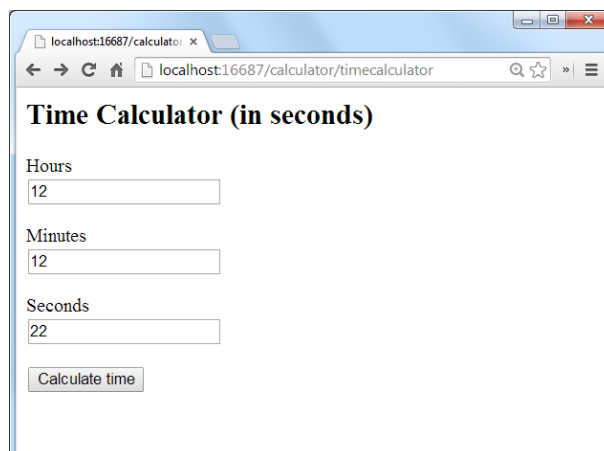
Exercise 3, *Time Calculator*

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 enter values for 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 input fields labeled "Hours", "Minutes", and "Seconds", each with the value "12". 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?

When you click the button, the user must see the total time in seconds:

12 hours + 12 minutes + 22 seconds = 43.942 seconds

To accomplish that, the next step is to create a Controller which handles the data sent from the form when users click the submit button ("Calculate time").

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 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;
```

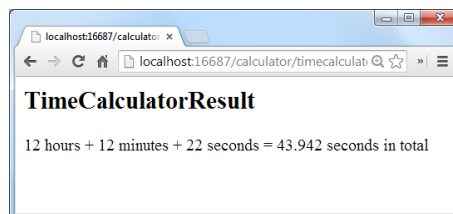
The above code gives you a new `TimeSpan` object `ts`, with the values given as arguments when the object is created. Notice that the `TimeSpan` property `TotalSeconds` returns a double. That's the reason why the variable `total` is declared as a double.

Don't worry if terms like class, objects, properties, and methods sound unfamiliar to you. We'll cover all this in much more detail in the next lesson.

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.

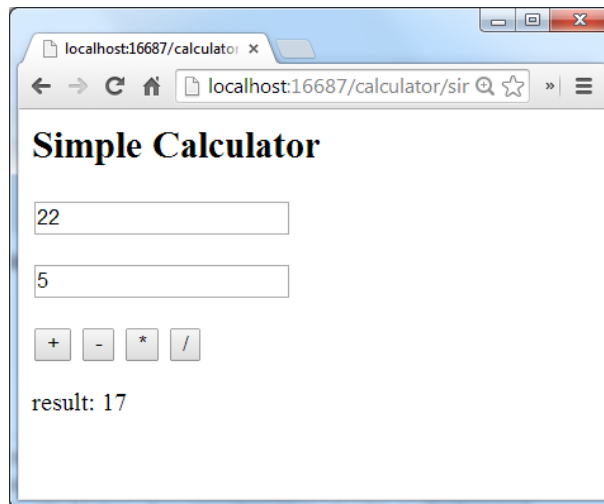
```
return View();
```

Tip:

When you write code in Visual Studio you can use the built in code format facility to format you code with line breaks and indentation. Use **Edit -> Format Document** to do that (or alternatively the shortcut **Ctrl K + Ctrl D**). Notice, to auto format the document it must be without syntax errors.

Exercise 4, Simple Calculator (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;
```

6. In the view, you can save values inside input fields by calling the `TextBox` method of `Html` helper class and give the name of the `ViewBag` property as string parameter:

```
@Html.TextBox("Number1")
```


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

// Jes Arbov, 2016-08-22