DAG 11 – Web applikationer i C# med ASP.NET MVC

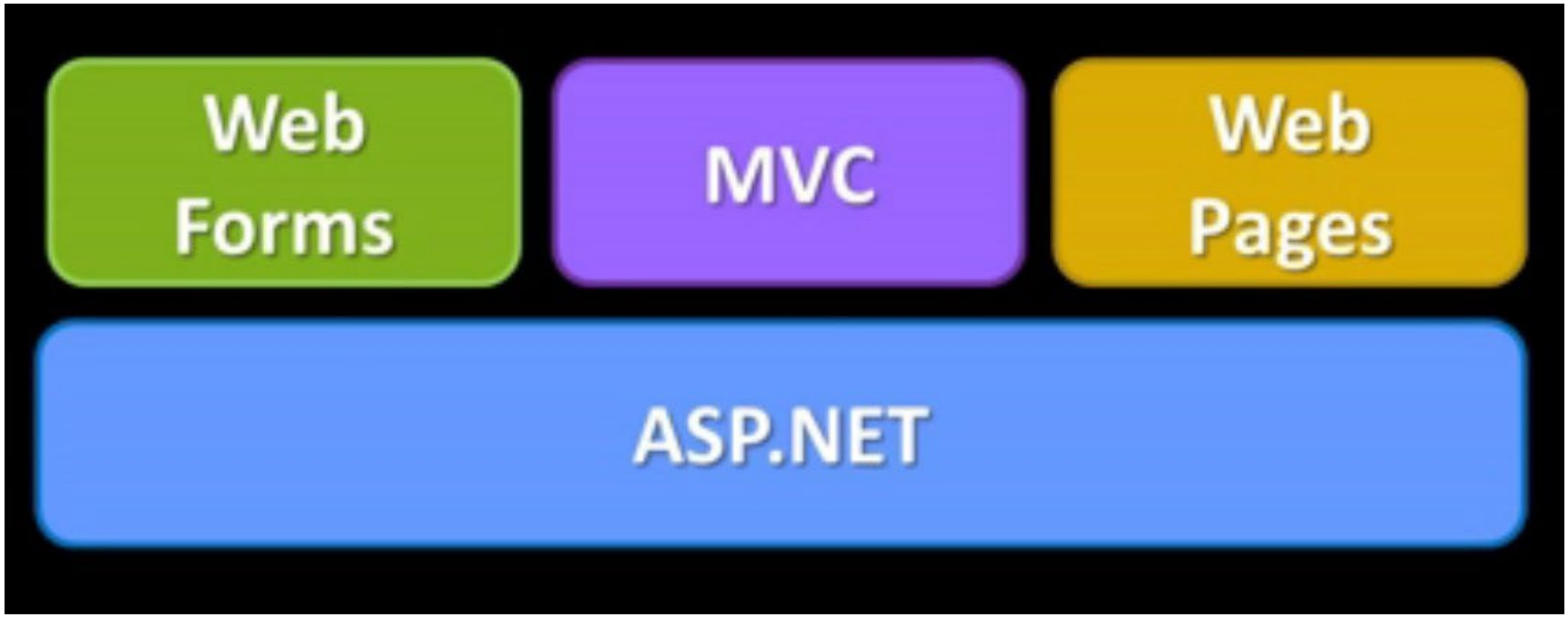
Indhold

- Introduktion til web applikationer The ASP.NET platform
  - The ASP.NET MVC framework
- Introduktion og eksempler i Razor.
  - Types, variables
  - Type conversions
  - Arrays
- Html helpers
- Opgaver

Software

- Internet Information Services (IIS). Dette er en webserver vi kan køre vores programmer i på localhost. Burde være installeret - ellers kan den installeres via Microsoft Web Platform Installer
  http://www.microsoft.com/web/downloads/platform.aspx

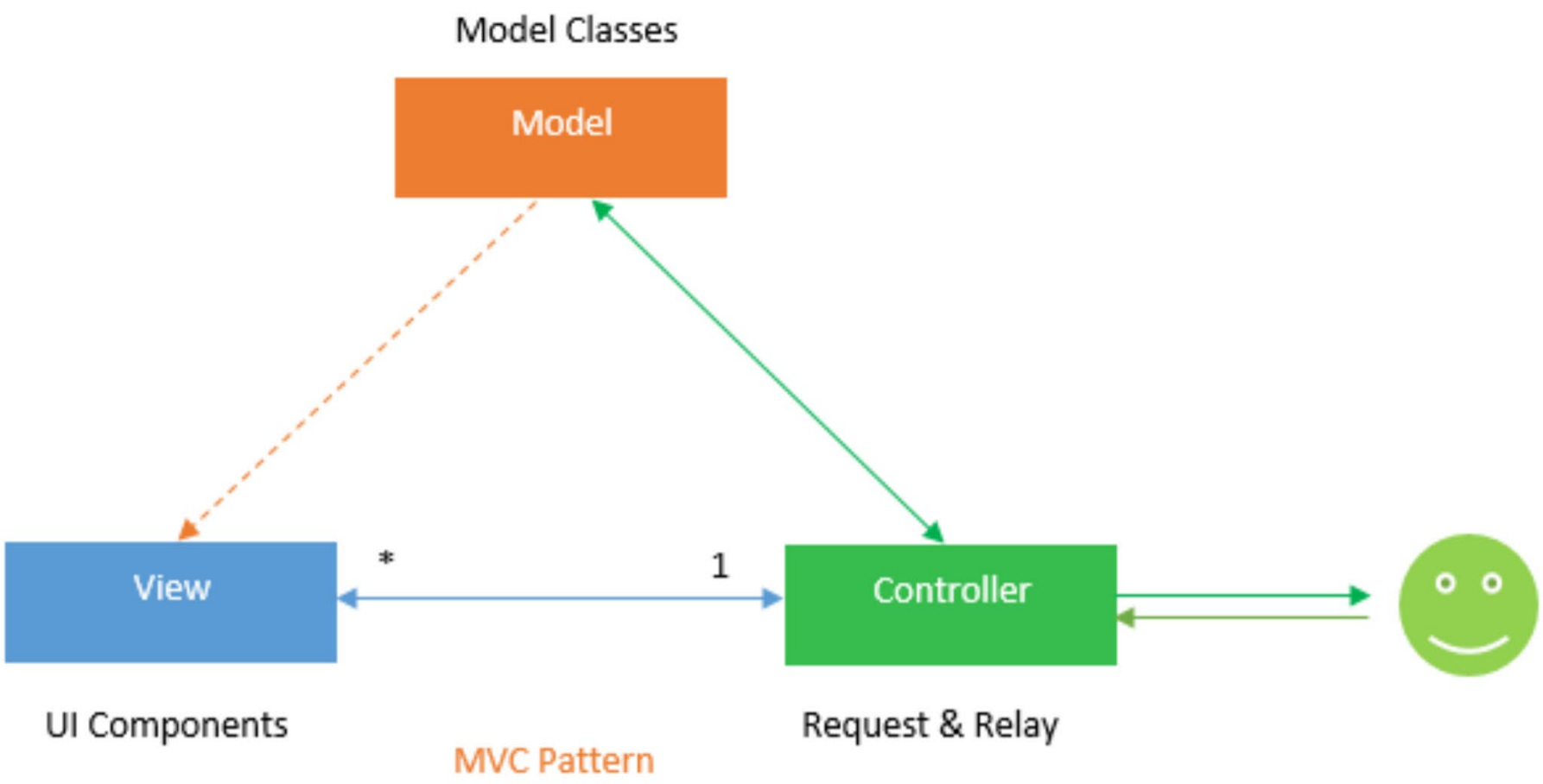
Web Sites med ASP.NET: 3
Frameworks til at vælge fra



Vi vil kun bruge MVC frameworket

ASP.NET MVC

MVC opbygningen

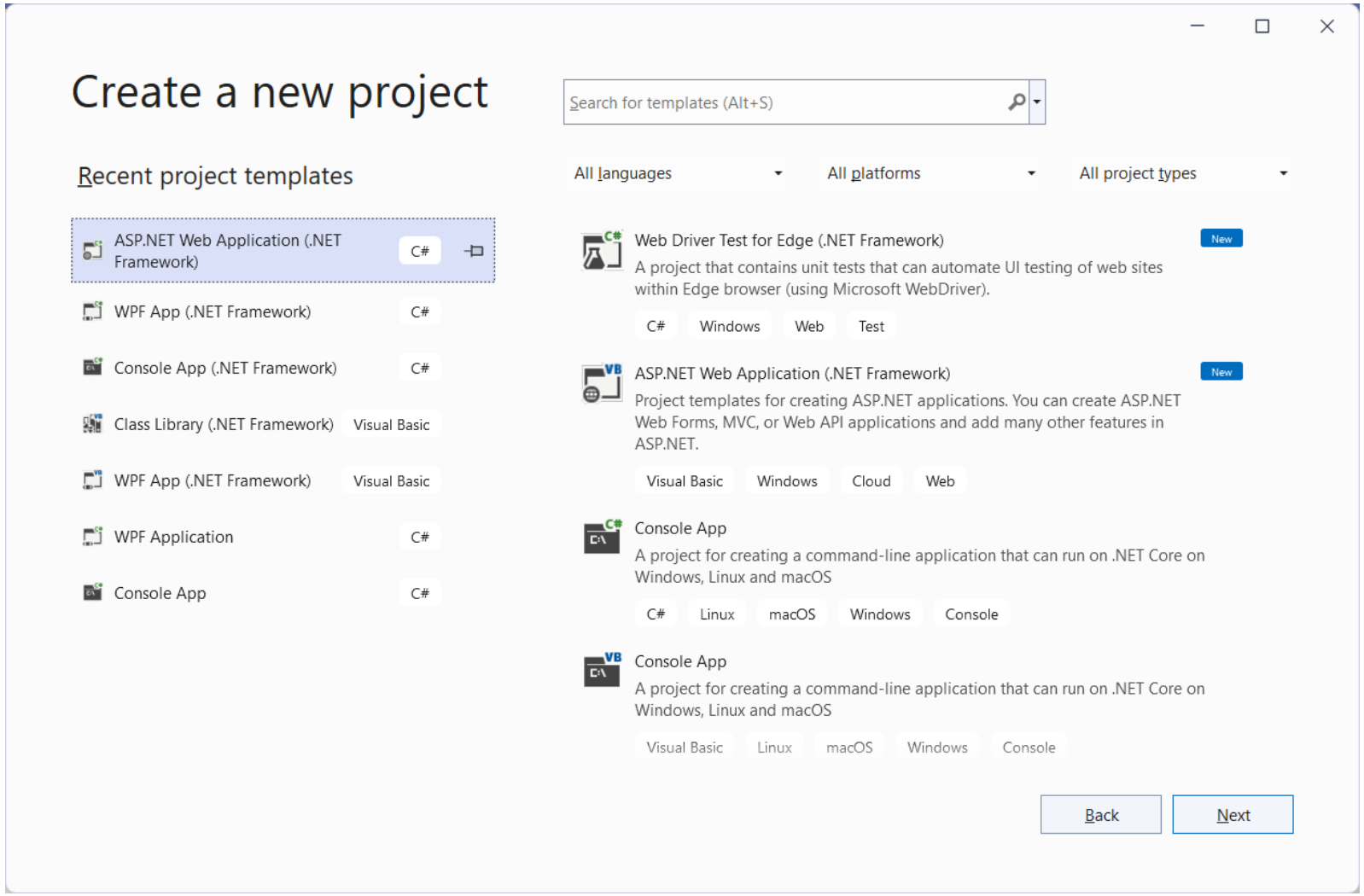


MVC anvendt i Webudvikling

- **Model**: Data Access Layer. Basalt set klasserne (og metoder til at manipulere data) som skal modellere dit system.
- **View**: En template til at generere HTML dynamisk. Definerer application UI og hvordan det vises (HTML, CSS, JavaScript på client siden, server side code: C#). Brug af template sproget Razor.
- **Controller**: Koordinere mellem View og Model, sørger for at de rigtige data kommer til view og modtager bruger input.

Hvordan kommer man i gang?

Det første ASP.NET MVC Project File -> New Project (Ctrl+Shift+N)



Create a new ASP.NET Web Application

Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application

A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication

None

Add folders & core references

- [ ] Web Forms
- [x] MVC
- [ ] Web API

Advanced

- [x] Configure for HTTPS
- [ ] Docker support

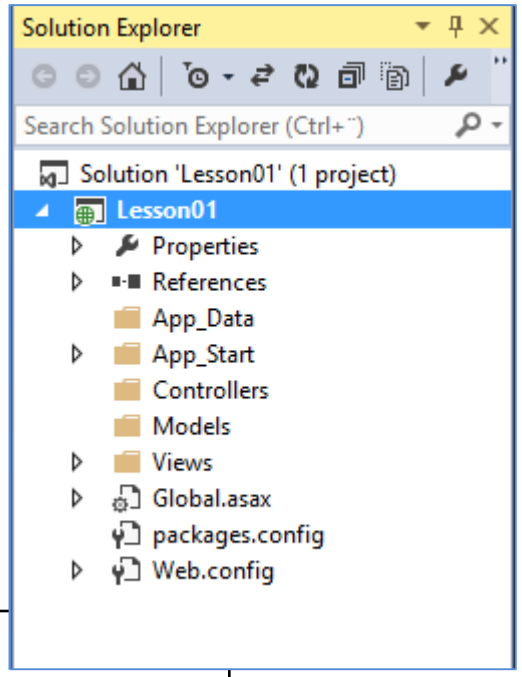
(Requires Docker Desktop)

- [ ] Also create a project for unit tests

WebApplication1.Tests

Back | Create

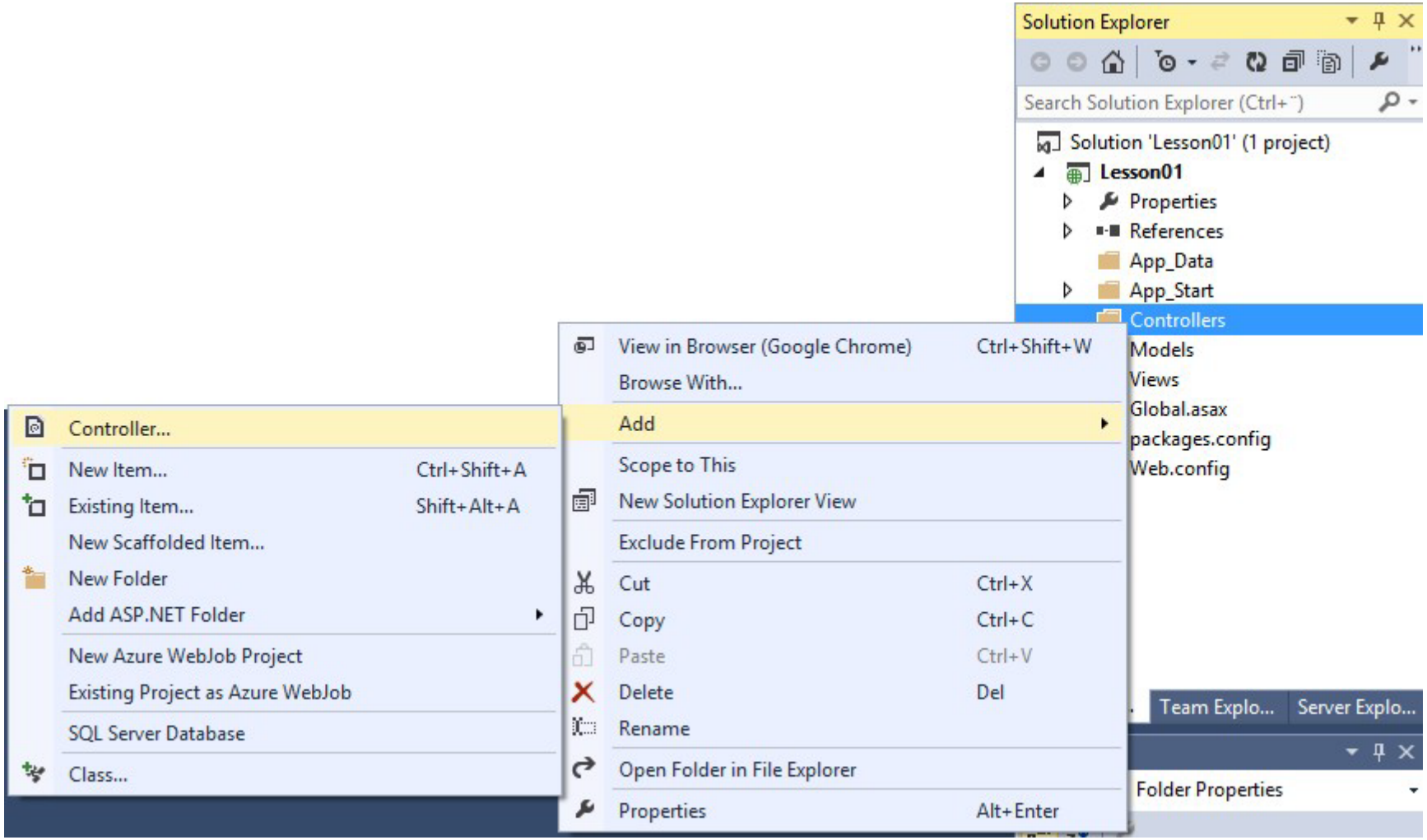
MVC
Application struktur



Top-Level Directories

Directory	Purpose
/Controllers	Where you put Controller classes that handle URL requests
/Models	Where you put classes that represent and manipulate data and business objects
/Views	Where you put UI template files that are responsible for rendering output such as HTML
/App_Data	Where you store data files you want to read/write
/App_Start	Where you put configuration code for features like Routing, bundling, and Web API
/Scripts	Where you put Java Script library files and scripts (.js)
/Content	Where you put CSS, images, and other site content , other than scripts

Tilføj ny Controller – højreklik på Controllers, vælg add



En ny Controller - opbygning

HomeController.cs

Lesson01

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace Lesson01.Controllers
8  {
9      public class HomeController : Controller
10     {
11         // GET: Home
12         public ActionResult Index()
13         {
14             return View();
15         }
16     }
17 }
```

Class references

Namespace

Class name

Action method

Return value

Hello world controller

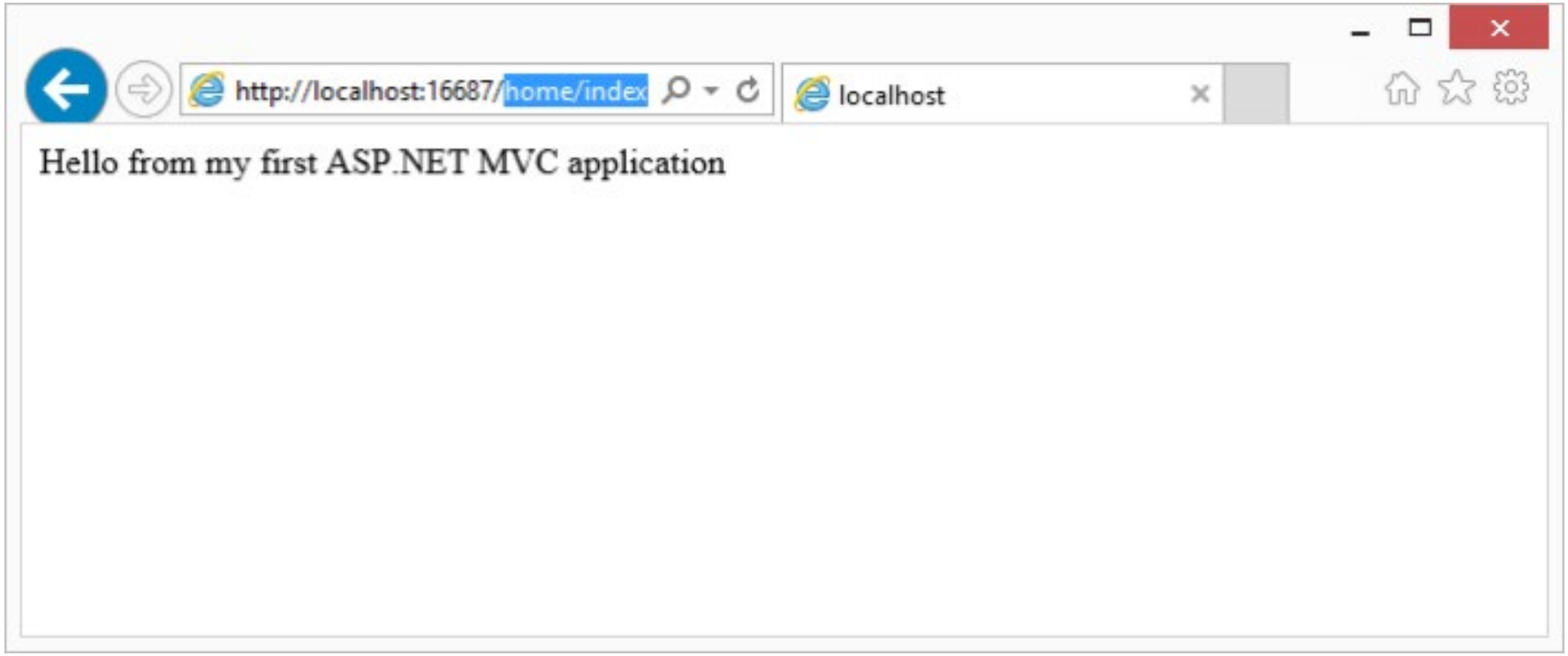
HomeController.cs

Lesson01 | Lesson01.Controllers.HomeController

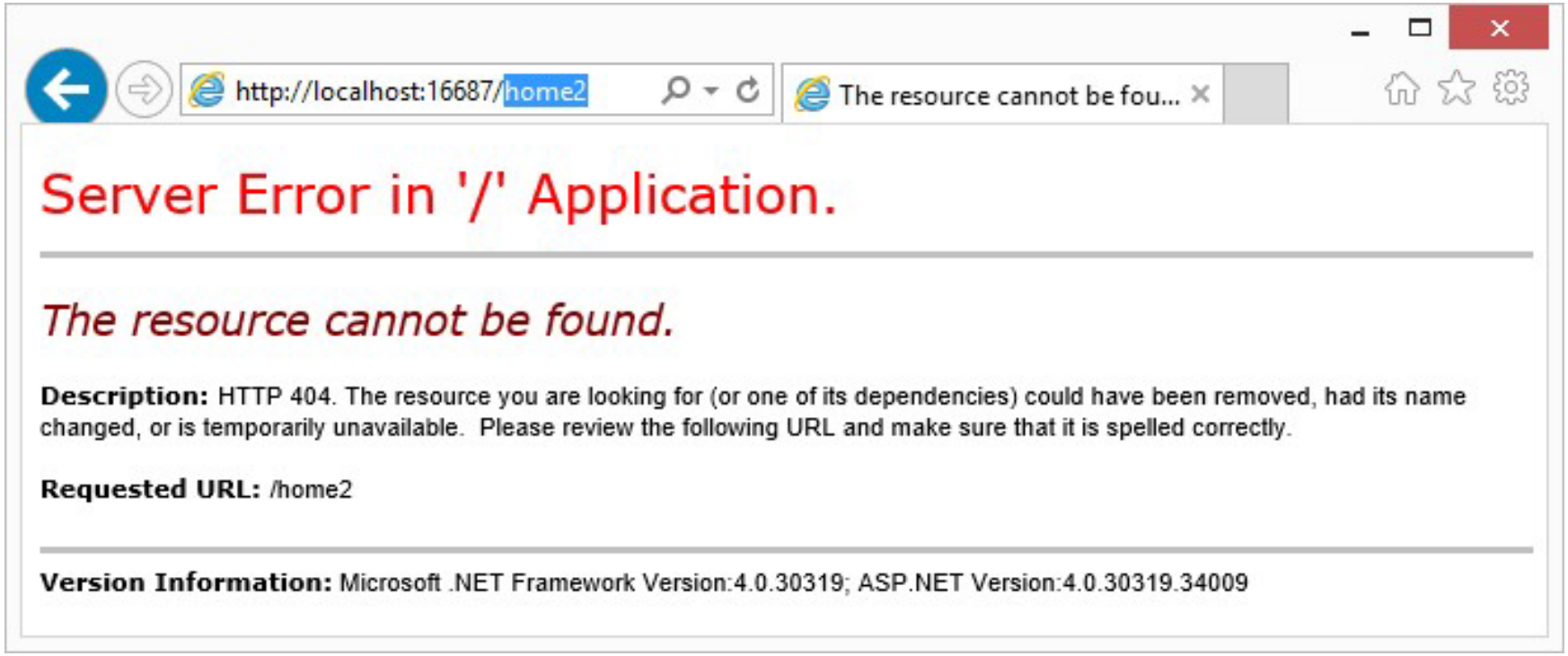
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace Lesson01.Controllers
8  {
9      public class HomeController : Controller
10     {
11         // GET: Home
12         public string Index()
13         {
14             return "Hello from my first ASP.NET MVC application";
15         }
16     }
17 }
```

Læg mærke til at retur typen af Index er ændret.

Run (Ctrl+F5 eller run knappen)



Call non-existent Controller - fejl



Routes Setting (`RouteConfig.cs`)

Lesson01 - Microsoft Visual Studio Express 2013 for Web

RouteConfig.cs | HomeController.cs

Lesson01 | Lesson01.RouteConfig | RegisterRoutes(RouteCollection routes)

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6  using System.Web.Routing;
7
8  namespace Lesson01 {
9      public class RouteConfig {
10         public static void RegisterRoutes(RouteCollection routes) {
11             routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
12
13             routes.MapRoute(
14                 name: "Default",
15                 url: "{controller}/{action}/{id}",
16                 defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
17             );
18         }
19     }
20 }
```

Solution Explorer

Solution 'Lesson01' (1 project)
- Lesson01
  - Properties
  - References
  - App_Data
  - App_Start
    - FilterConfig.cs
    - RouteConfig.cs

Output — Show output from: Build

Controllerens rolle

- Der er ikke noget **direkte forhold** mellem en URL og en fil på web serveren. Det er routing systemet som laver en mapping (senere tema)

- Der er derimod en forbindelse fra URL'en og metode navnet i en controller klasse

Ændre return typen af Index metoden tilbage til ActionResult og return a View …

HomeController.cs

Lesson01

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace Lesson01.Controllers
8  {
9      public class HomeController : Controller
10     {
11         // GET: Home
12         public ActionResult Index()
13         {
14             return View();
15         }
16     }
17 }
```

… og Add a View

Lesson01 - Microsoft Visual Studio Express 2013 for Web

FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST WINDOW HELP

Google Chrome | Debug

HomeController.cs

Lesson01 | Lesson01.Controllers.HomeController | Index()

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace Lesson01.Controllers
8  {
9      public class HomeController : Controller
10     {
11         // GET: Home
12         public ActionResult Inde
13         {
14             return View();
15         }
16     }
17 }
```

Go To View	Ctrl+M, Ctrl+G
Add View...	
Refactor	
Organize Usings	
Comment Selection	Ctrl+K, Ctrl+C
Uncomment Selection	Ctrl+K, Ctrl+U
Insert Snippet...	Ctrl+K, Ctrl+X
Surround With...	Ctrl+K, Ctrl+S
Go To Definition	F12
Find All References	Shift+F12
Breakpoint	
Run To Cursor	Ctrl+F10
Run Flagged Threads To Cursor	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Outlining	

Right click at controller and choose Add View…

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'Lesson01' (1 project)
  - Lesson01
    - Properties
    - References
    - App_Data
    - App_Start
    - Content
    - Controllers
      - HomeController.cs
    - fonts
    - Models
    - Scripts
    - Views
      - Shared
        - _Layout.cshtml
        - _ViewStart.cshtml
      - Web.config
    - Global.asax
    - packages.config
    - Web.config

Solution Ex... | Team Explo... | Server Explo...

Properties

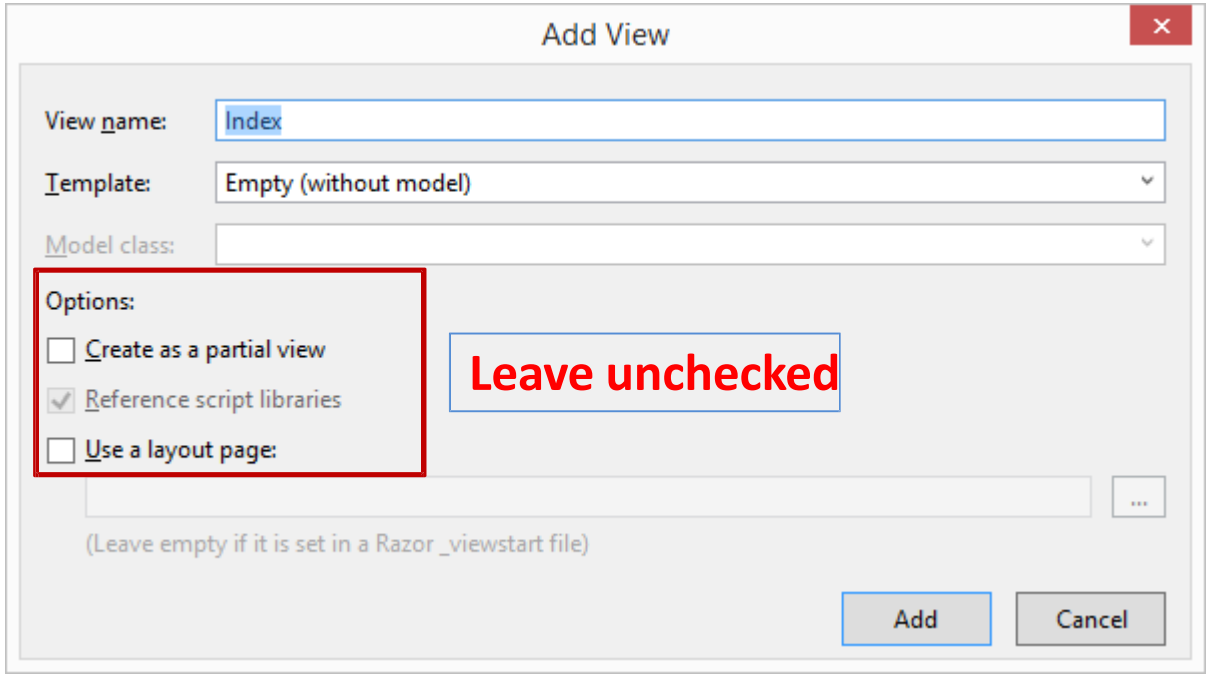
Output

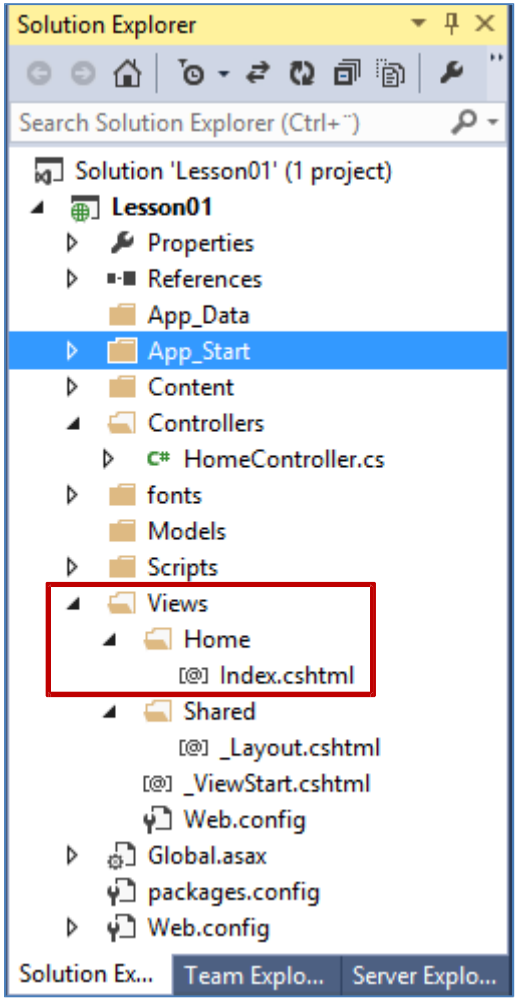
Show output from: Package Manager

Package Manager Console | Error List | Output

Ready | Ln 12 | Col 32 | Ch 32 | INS

… med MVC 5 eller højere vil der være ekstra Bootstrap filer tilføjet et projekt.





The View

Lesson01 - Microsoft Visual Studio Express 2013 for Web

FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST WINDOW HELP

Google Chrome | Debug

Index.cshtml | HomeController.cs

```
1
2   @{
3       Layout = null;
4   }
5
6   <!DOCTYPE html>
7
8   <html>
9   <head>
10      <meta name="viewport" content="width=device-width" />
11      <title>Index</title>
12  </head>
13  <body>
14      <div>Hello from a ASP.NET MVC 5 View
15      </div>
16  </body>
17  </html>
18
```

100 %

Output

Show output from: Build

```
1>------ Build started: Project: Lesson01, Configuration: Debug Any CPU ------
1>  Lesson01 -> C:\Users\jear\Google Drev\Documents\eaaa-it\2014e\backend\projects\Lesson01\
```

Package Manager Console | Error List | Output

Solution Explorer

Search Solution Explorer (Ctrl+¨)

- Solution 'Lesson01' (1 project)
  - Lesson01
    - Properties
    - References
    - App_Data
    - App_Start
    - Controllers
      - HomeController.cs
    - Models
    - Views
      - Home
        - Index.cshtml
      - web.config

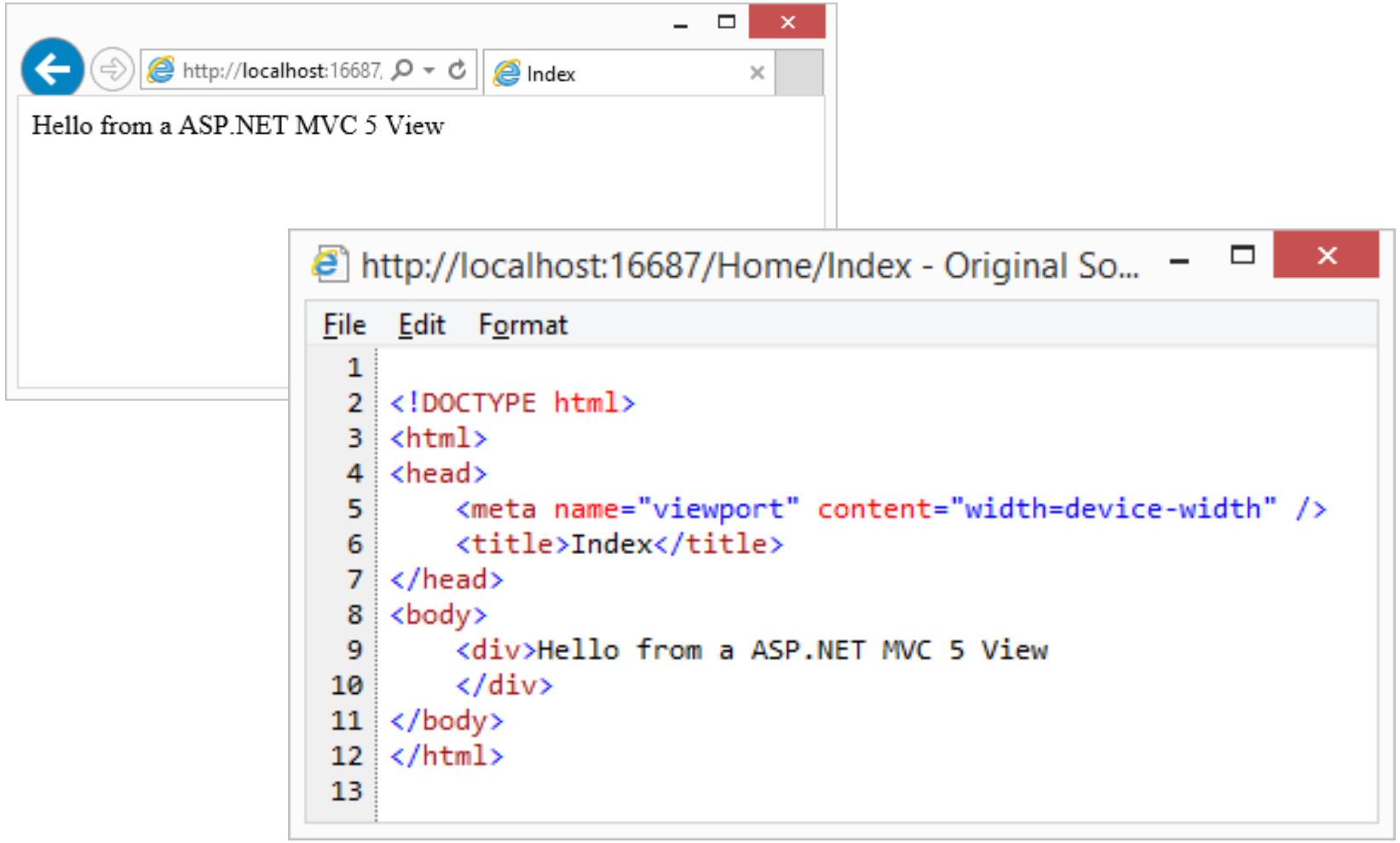
Solution Ex... | Team Explo... | Server Explo...

Properties

Document

Item(s) Saved | Ln 18 | Col 1 | Ch 1 | INS

Executing the web page



ASP.NET MVC Views

- I nogle frameworks – som PHP er Views direkte tilgængelige via en URL. **Men i ASP.NET MVC kan du ikke direkte angive et view som URL – det vil give en fejl**.
- I stedet for, er et view altid generet af en controller som giver de data som viewet skal renderes med. Så en url skal matche de ruter som er defineret i routing configuration – typisk /controllername/actionmethodname. F.eks. /home/Index

Lille opgave

Lav opgave 11.1

Der er flere måder at sende data fra controller til View

- ViewData (key/value pairs)
  - `ViewData["movie"] = "One Flew Over the Cuckoo's Nest";`
  - `ViewData["movie"]["year"] = 1975;`
- ViewBag (properties, same underlying object as ViewData)
  - `ViewBag.Movie = "Forrest Gump";`
  - `ViewBag.Movie.Year = 1994;`
- Model (objects)
  - `new Movie{Title = "Intouchables", Year = 2012};`

Der er flere måder at sende data fra controller til View

- ViewData (key/value pairs)
  - **Ikke typestærk**
- ViewBag (properties, same underlying object as ViewData)
  - **Ikke typestærk, ikke check for, om attribut er sat**
- Model (objects)
  - **Typestærk**

Eksempel: Controller'en

Lesson01 - Microsoft Visual Studio Express 2013 for Web

Index.cshtml | Movie.cs | MovieController.cs | Index.cshtml | HomeController.cs

Lesson01 | Lesson01.Controllers.MovieControll | Index()

```
10  public class MovieController : Controller
11  {
12      // GET: Movie
13      public ActionResult Index()
14      {
15          ViewData["movie"] = "One Flew Over the Cuckoo's Nest";
16          ViewData["year"] = 1975;
17
18          ViewBag.Movie = "Forrest Gump";
19          ViewBag.Year = 1994;
20
21          return View(new Movie { Title = "Intouchables", Year = 2011 });
22      }
23  }
24 }
```

Output

Show output from: Build

```
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Solution Explorer

- References
- App_Data
- App_Start
- Controllers
  - HomeController.cs
  - MovieController.cs
- Models
  - Movie.cs
- Views
  - Home
    - Index.cshtml
  - Movie
    - Index.cshtml

Item(s) Saved | Ln 21 | Col 13 | Ch 13 | INS

Eksempel: View'et

Lesson01 - Microsoft Visual Studio Express 2013 for Web

Index.cshtml* | Movie.cs | MovieController.cs | Index.cshtml | HomeController.cs

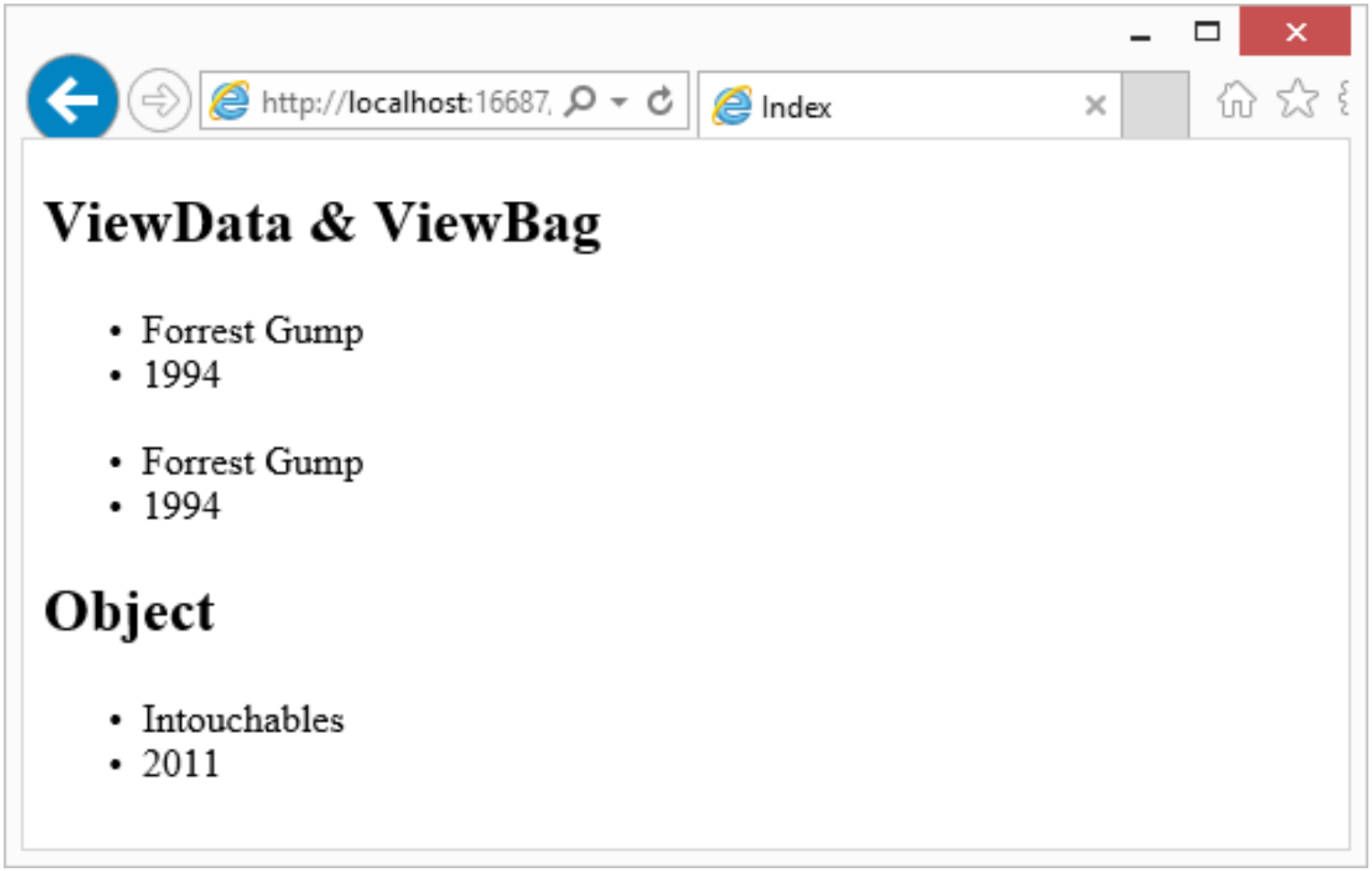
```
@using Lesson01.Models;
@model Movie
@{
    Layout = null;
}
    <h2>ViewData & ViewBag</h2>
    <ul>
        <li>@ViewData["movie"]</li>
        <li>@ViewData["year"]</li>
    </ul>
    <ul>
        <li>@ViewBag.Movie</li>
        <li>@ViewBag.Year</li>
    </ul>

    <h2>Object</h2>
    <ul>
        <li>@Model.Title</li>
        <li>@Model.Year</li>
    </ul>
```

Output

```
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Eksempel: output i browseren



Controller.View

Adskillige overloads:

View(String, String, Object)	Creates a ViewResult object using the view name, master-page name, and model that renders a view.
View()	Creates a ViewResult object that renders a view to the response.
View(Object)	Creates a ViewResult object by using the model that renders a view to the response.
View(String)	Creates a ViewResult object by using the view name that renders a view.
View(IView)	Creates a ViewResult object that renders the specified IView object.
View(String, Object)	Creates a ViewResult object that renders the specified IView object.
View(String, String)	Creates a ViewResult object using the view name and master-page name that renders a view to the response.
View(IView, Object)	Creates a ViewResult object that renders the specified IView object.

Controller.View

Parametre:

C# Copy

```
protected internal System.Web.Mvc.ViewResult View (string viewName, object model);
```

Parameters

viewName String

The view that is rendered to the response.

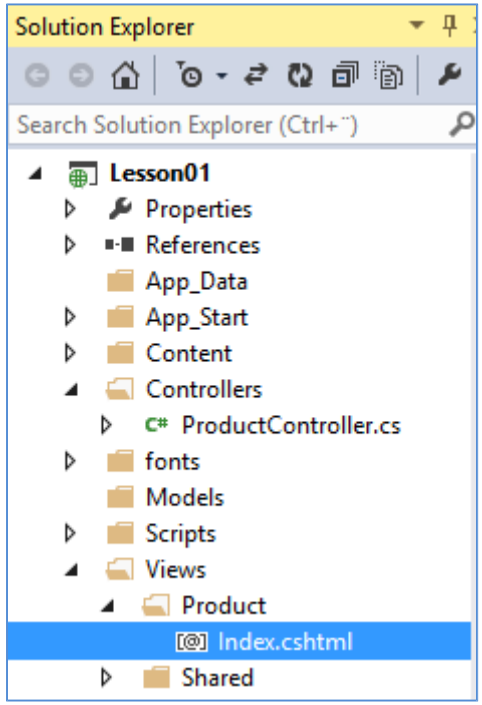
model Object

The model that is rendered by the view.

ASP.NET MVC Conventions

Convention over configuration

- Directories
  - Controllers
  - Models
  - Views
- Eksempler:
  - Hver Controllers navn skal slutte med *Controller*: `ProductController`
  - Views som en Controller bruger er i en undermappe, som er navngivet efter Controlleren og filnavnet er navngivet efter Action metoden. (Hvis ikke viewname er angivet i view()-kald) For eksempel: **`/Views/Product/Index.cshtml`**



Lille opgave

Lav opgave 11.2

Razor

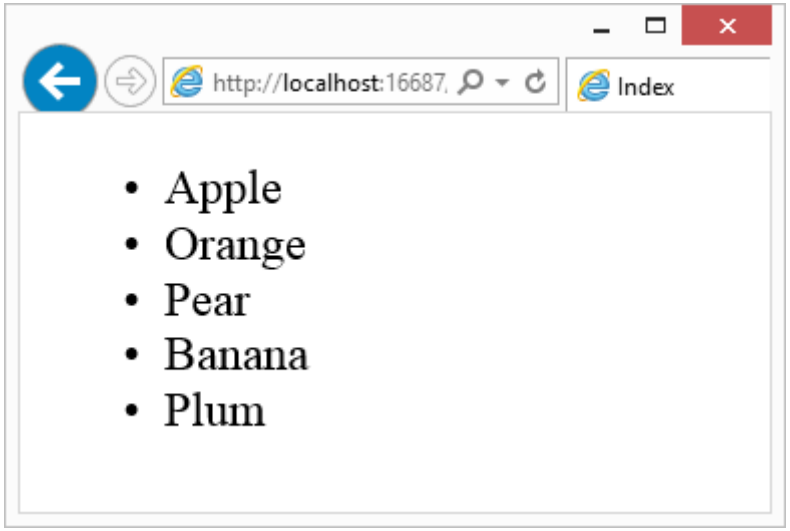
HTML og C#-kode blandet sammen

FruitController

```
7   namespace Lesson01.Controllers
8   {
9       public class FruitController : Controller
10      {
11          // GET: Fruit
12          public ActionResult Index()
13          {
14              string[] fruits = new string[] { "Apple", "Orange", "Pear", "Banana", "Plum" };
15              ViewBag.Fruits = fruits;
16
17              return View();
18          }
19      }
20  }
21
```

Fruit View

```
11  <body>
12      <ul>
13          @foreach(string fruit in ViewBag.Fruits) {
14              <li>@fruit</li>
15          }
16      </ul>
17  </body>
```



Razor eksempel - arrays

```
@{
    int[] nums = { 1, 7, 9, 20 };
    // add numbers in array
    int sum = 0;
    for (int i = 0; i < nums.Length; i++)
    {
        sum = sum + nums[i];
    }
}
@sum //37
```

Razor eksempel - Arrays af type string

```
<ul>
    @{
        string[] colorNames = new string[5];
        colorNames[0] = "Yellow";
        colorNames[1] = "Green";
        colorNames[2] = "Red";
        colorNames[3] = "Blue";
        colorNames[4] = "White";

        for (int i = 0; i < colorNames.Length; i++) {
            <li>@colorNames[i]</li>
        }
    }
    @s
</ul>
```

The foreach loop i Razor

```
<ul>
    @{
        string[] colorNames = new string[5];
        colorNames[0] = "Yellow";
        colorNames[1] = "Green";
        colorNames[2] = "Red";
        colorNames[3] = "Blue";
        colorNames[4] = "White";

        foreach (string color in colorNames) {
            <li>@color</li>
        }
    }
</ul>
```

Lille opgave

Lav opgave 11.3

Parametre i url

I url'en, efter action, kan sættes parametre

Eks.:https://localhost:44383/home/index/3

Defineres i route.config:

```
 url: "{controller}/{action}/{id}",
 defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional}
```

Inkluder i action-metode:

```
 public ActionResult Index(int? id)
```

Parameternavne skal helst matche, især hvis de er optional.

https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/asp-net-mvc-routing-overview-cs

Lave Forms i Views – the old way

```
<form>
    <p>
        <label for="firstname">Firstname</label><br />
        <input type="text" id="firstname" name="firstname" />
    </p>
    <p>
        <label for="lastname">Lastname</label><br />
        <input type="text" id="lastname" name="lastname" />
    </p>
    <input type="button" value="Register" />
</form>
```

Lave Forms i Views med Html Helpers

```
@
...
using (Html.BeginForm()) {
    <p>
        @Html.Label("Firstname") <br />
        @Html.TextBox("Firstname")
    </p>
    <p>
        @Html.Label("Lastname") <br />
        @Html.TextBox("Lastname")
    </p>
    <input type="submit" value="Register" />
}
```

The HTML Output

```
<form action="/FormHandler/Index" method="post">
            <p>
                <label for="Firstname">Firstname</label> <br />
                <input id="Firstname" name="Firstname" type="text" value="" />
            </p>
            <p>
                <label for="Lastname">Lastname</label> <br />
                <input id="Lastname" name="Lastname" type="text" value="" />
            </p>
            <input type="submit" value="Register" />
</form>
```

Håndtere form data

```
public class FormHandlerController : Controller
{
    // GET: FormHandler
    public ActionResult Index()
    {
        return View();
    }

    // POST: FormHandler
    [HttpPost]
    public ActionResult Index(FormCollection formCollection) {
        ViewBag.Firstname = formCollection["Firstname"];
        ViewBag.Lastname = formCollection["Lastname"];
        return View();
    }
}
```

The View: et eksempel

```
@if(ViewBag.Firstname == null || ViewBag.Lastname == null) {
    <h2>Register</h2>
    using (Html.BeginForm()) {
        <p>
            @Html.Label("Firstname") <br />
            @Html.TextBox("Firstname")
        </p>
        <p>
            @Html.Label("Lastname") <br />
            @Html.TextBox("Lastname")
        </p>
        <input type="submit" value="Register" />
    }
}
else {
    <p>Your name:</p>
    <p>@ViewBag.Firstname @ViewBag.Lastname </p>

}
```