

MVC Views

- ViewData og ViewBag
- ViewModel
- Tilføj et View
- Razor View Engine
 - Syntax
 - XSS
 - Layout & View overview
 - Layout & View relation
 - Sections i Layout
 - Sections i View
 - Partial view

ViewData og ViewBag

```
ViewData["name"] = "Thomas"
```

```
ViewBag.name = "Thomas"
```

- Kommunikerer mellem Controller og View
- Korte liv - betyder værdi bliver null, når vi er færdige med siden.
- Tildeler ikke “Strongly typed views” med mindre den funktionalitet bliver tildelt
- ViewBag er af typen dynamic
- ViewBag - kræver ikke typecasting for komplekse datatype.
- ViewData - er et Bibliotek over objekter, der er af typen ViewDataDictionary og er tilgængelig ved hjælp af strenge som nøgle.
- ViewData - kræver typecasting for komplekse datatype og kontrollere for NULL-værdier for at undgå fejl

Forskellen på ViewData og ViewBag: kompleks objekt

ViewBag

```
public ActionResult Index()
{
    var emp = new Employee
    {
        EmpId = 101,
        Name = "Thomas Hansen",
        Salary = 35000,
        Address = "Skovmarksvej 30, 4100"
    };

    ViewBag.Employee = emp;

    return View();
}
```

```
@{
    ViewBag.Title = "Home Page";
}
<h4>@ViewBag.Employee.Name</h4>
```

ViewData

```
public ActionResult Index()
{
    var emp = new Employee
    {
        EmpId = 101,
        Name = "Thomas Hansen",
        Salary = 35000,
        Address = "Skovmarksvej 30, 4100"
    };
    ViewData["emp"] = emp;
    return View();
}
```

```
@using MvcWebApplication.Models
@{
    ViewBag.Title = "Home Page";
    var viewDataEmployee = ViewData["emp"] as Employee;
}
<h2>@viewDataEmployee.Name</h2>
```

Forskellen på ViewData og ViewBag: List<T>

ViewBag

```
@foreach (var emp in ViewBag.Employee)
{
    <h2>@emp.Name</h2>
}
```

ViewData

```
@using MvcWebApplication.Models
@{
    ViewBag.Title = "Home Page";
    var viewDataEmployee = ViewData["emp"] as List<Employee>;
}
@if (viewDataEmployee != null)
{
    foreach (var emp in viewDataEmployee)
    {
        <h2>@emp.Name</h2>
    }
}
```

namespaces i web.config

Her kan du tilføje de namespaces som bliver brugt hyppigt

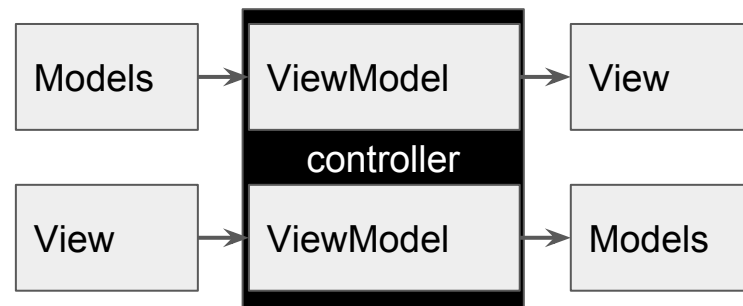
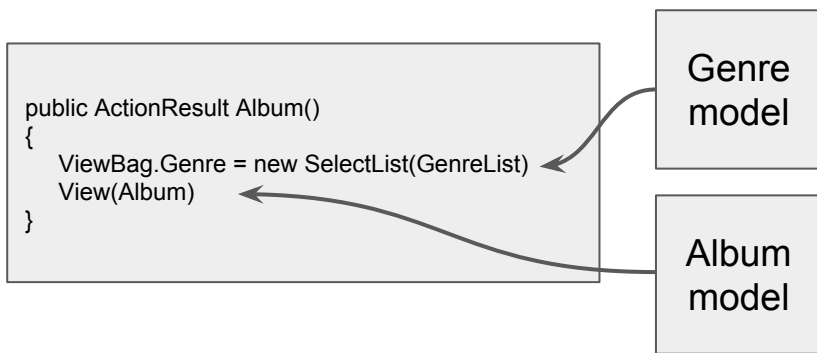
Her undgår man at skrive en using, hver eneste gang der er brug for en model

```
@using MvcWebApplication.Models
```

```
<system.web.webPages.razor>  
...  
<pages pageBaseType="System.Web.Mvc.WebViewPage">  
  <namespaces>  
    <add namespace="System.Web.Mvc" />  
    <add namespace="System.Web.Mvc.Ajax" />  
    <add namespace="System.Web.Mvc.Html" />  
    <add namespace="System.Web.Routing" />  
  
    <add namespace="MvcMusicStore.Models" />  
  </namespaces>  
</pages>  
</system.web.webPages.razor>
```

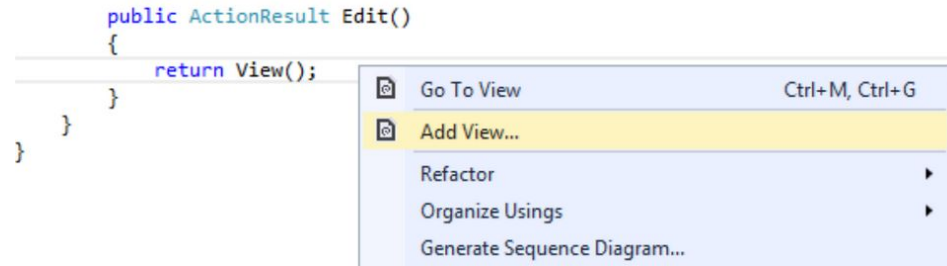
Kort om ViewModel *el. view specific model*

- Ikke det samme som i Model View ViewModel (MVVM) design pattern
- Strongly typed views
- Kan styre behandling af, model til views & view til models



Tilføj et View #1

Højreklik på en ActionResult og vælg Add View



Tilføj et View #2

Udfyld View name og vælg en template

Add View

View name:

Template:

Model class:

Data context class:

View options:

- ☐ Create as a partial view
- ☒ Reference script libraries
- ☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Add **Cancel**

Add View

View name:

Template:

Model class:

Data context class:

View options:

- ☐ Create as a partial view
- ☒ Reference script libraries
- ☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Add **Cancel**

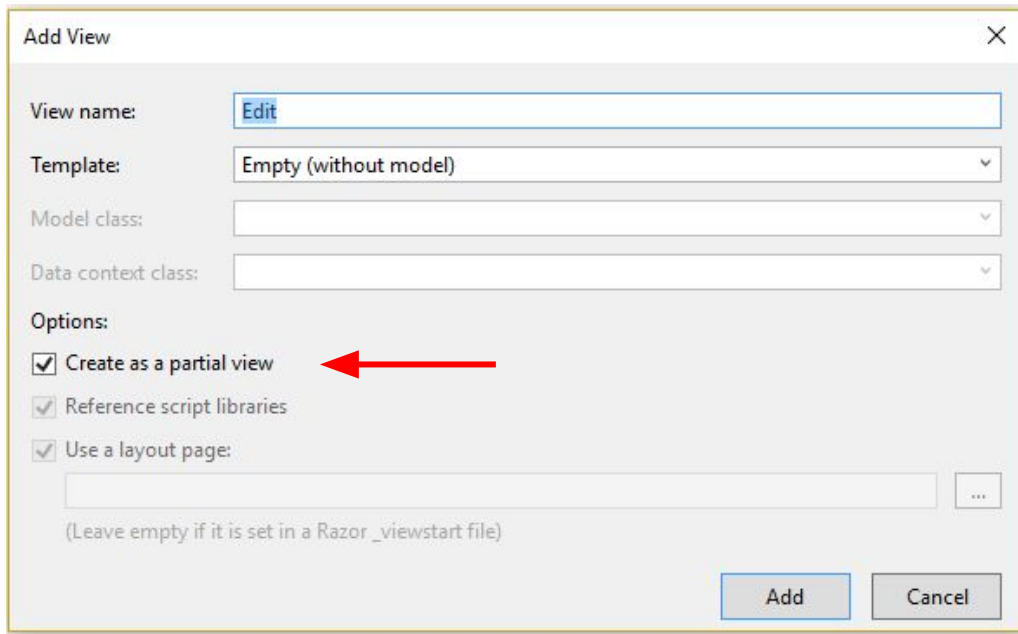
Tilføj et View: View Scaffold Types

SCAFFOLD	DESCRIPTION
Create	Creates a view with a form for generating new instances of the model. Generates a label and input field for each property of the model type.
Delete	Creates a view with a form for deleting existing instances of the model. Displays a label and the current value for each property of the model.
Details	Creates a view that displays a label and the value for each property of the model type.
Edit	Creates a view with a form for editing existing instances of the model. Generates a label and input field for each property of the model type.
Empty	Creates an empty view. Only the model type is specified using the <code>@model</code> syntax.
Empty (without model)	Creates an empty view, as with the Empty scaffold. In this case, however, there's no model so you're not required to select a model type when you select this scaffold. This is the only scaffold type which does not require you to select a model type.
List	Creates a view with a table of model instances. Generates a column for each property of the model type. Make sure to pass an <code>IEnumerable<YourModelType></code> to this view from your action method. The view also contains links to actions for performing the create/edit/delete operations.

Tilføj et View: Partial view

Skaber et view som ikke har
f.eks <html> og <body>

Et genanvendeligt View som
skal bruges i andre Views



Add View

View name:

Template:

Model class:

Data context class:

Options:

- ☒ Create as a partial view
- ☒ Reference script libraries
- ☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

Tilføj et View: Layout page

Lad den være hvis et layout er defineret i `_ViewStart.cshtml`

Et View har også muligheden for at bruge et andet layout

Vælg denne fra hvis et View ikke skal bruge layout og skal indeholde `<html>` og `<body>`

The screenshot shows the 'Add View' dialog box with the following configuration:

- View name:** Edit
- Template:** Empty (without model)
- Model class:** (empty)
- Data context class:** (empty)
- Options:**
 - ☐ Create as a partial view
 - ☒ Reference script libraries
 - ☒ Use a layout page: (indicated by a red arrow)

Below the 'Use a layout page:' checkbox is an empty text input field with a browse button ('...'). A note below the field states: '(Leave empty if it is set in a Razor _viewstart file)'. At the bottom right are 'Add' and 'Cancel' buttons.

Razor View Engine

Her vises et eksempel på en sammenligning fra en console program og Razor

Det er muligt at blande C# med html og javascript

```
<script>alert("@msg")</script>
```

```
var cars = new List<ICar>();
cars.Add(new Jeep("'74-'79 J10", "Shortbed", 4000, 80000));
cars.Add(new SportsCar("4C", "Alfa Romeo", 5000, 200000));

foreach (var car in cars)
{
    Console.WriteLine(car);
}
```

```
@foreach (var emp in Model)
{
    <h1>@emp.Name</h1>
}
```

Razor View Engine

Razer er blevet skabt for at gøre det nemmere for udviklerne

En masse ekstra støj er blevet fjernet

```
<% foreach(var item in stuff) { %>
    <li>The item name is <%= item %>.</li>
<% } %>
```

```
@foreach(var emp in Model)
{
    <h1>@emp.Name</h1>
}
```

Razor View Engine: Syntax

<https://gist.github.com/Cosmoseyeballs/483a84b5e7166f9138be2a6a41f5b9ba>

ASP.NET Razor Syntax Reference

Code Block

```
@{
    int x = 123;
    string y = "because.";
}
```

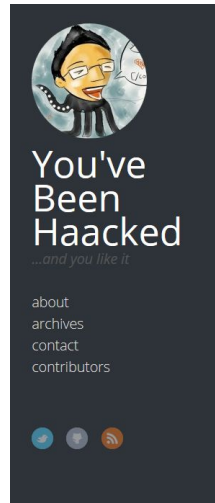
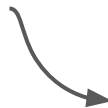
Expression (Html Encoded)

```
<span>@Model.Message</span>
```

Expression (Unencoded)

```
<span>
    @Html.Raw(Model.Message)
</span>
```

Original



Syntax/Sample	Razor
Code Block	<pre>@{ int x = 123; string y = "because."; }</pre>
Expression (Html Encoded)	<pre>@Model.Message</pre>
Expression (Unencoded)	<pre> @Html.Raw(model.Message) </pre>
Combining Text and markup	<pre>@foreach (var item in items) { @item.Prop }</pre>
Mixing code and Plain text	<pre>@if (foo) { <text>Plain Text</text> }</pre>
Using block	<pre>@using (Html.BeginForm()) { <input type="text" value="input here"> }</pre>
Mixing code and plain text (alternate)	<pre>@if (foo) { @Plain Text is @bar }</pre>
Email Addresses	<pre>Hi philha@example.com</pre>

Forked fra <https://gist.github.com/jonlabelle/8738373>

Razor View Engine : XSS

- Cross-site scripting (XSS) er en form for computer sikkerhedsbrist typisk i webapplikationer.
- XSS gør det muligt for angribere at injecte scripts kode ind i websider som bliver set af andre brugere

```
<script type="text/javascript">
    $(function () {
        var message = 'Hello @ViewBag.Username';
        $("#message").html(message).show('slow');
    });
</script>
```

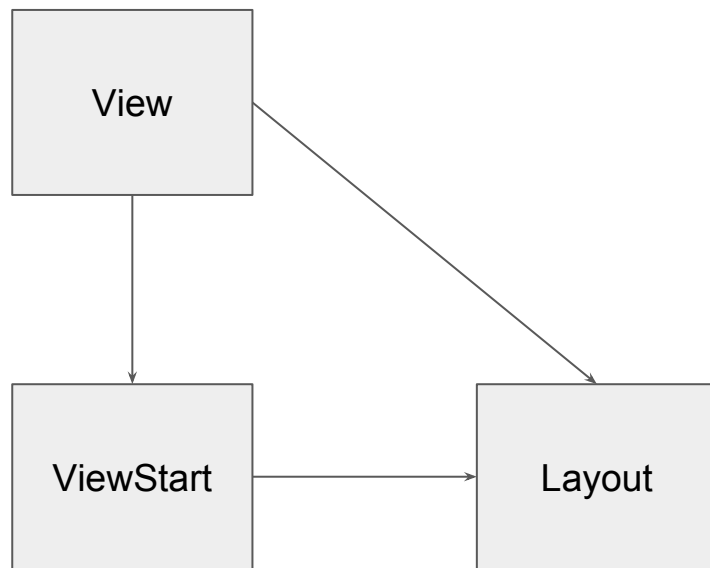
Username:
\\x3cscript\\x3e%20alert(\\x27pw
nd\\x27)%20\\x2c/script\\x3e

```
<script type="text/javascript">
    $(function () {
        var message = 'Hello @Ajax.JavaScriptStringEncode(ViewBag.Username)';
        $("#message").html(message).show('slow');
    });
</script>
```

Razor View Engine : Layout & View overview

Et View som udgangspunkt behøver ikke at angive et layout, medmindre ViewStart ikke findes

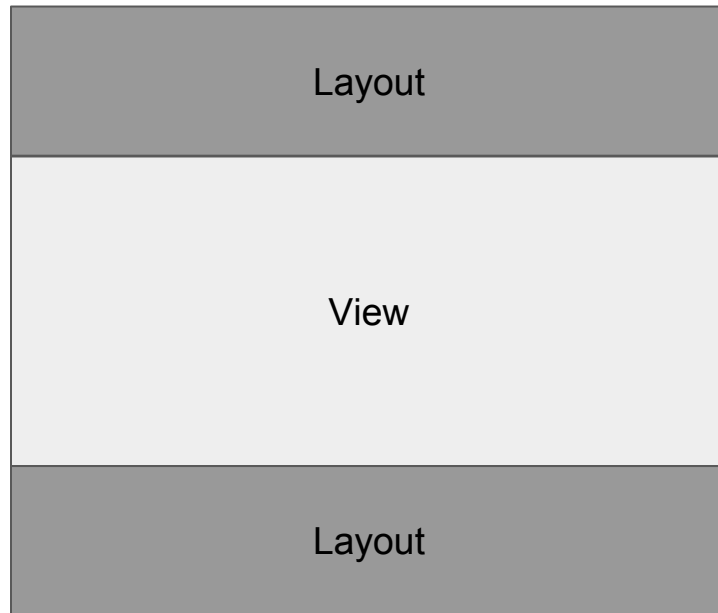
View har muligheden for at angive et andet Layout



Razor View Engine : Layout & View relation

Et Layout skal indeholde
`@RenderBody()`

Indholdet fra et View bliver placeret i
RenderBody



Razor View Engine : Sections i Layout

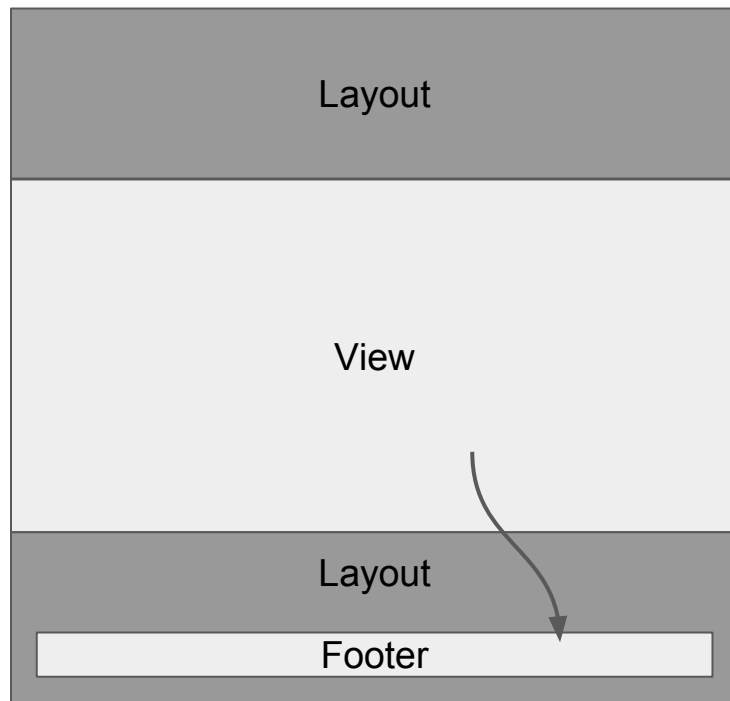
Et Layout kan have mange sections

En sections bliver defineret i et layout ved at skrive f.eks

```
@RenderSection("Footer")
```

Som default skal en section være defineret i et View. Der er mulighed for at ændre dette ved at tildele et ekstra parameter

```
@RenderSection("Footer", false)
```

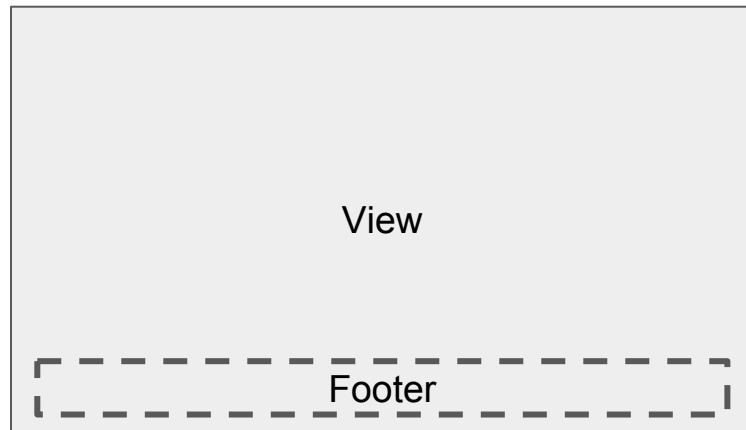


Razor View Engine : Sections i View

Et Layout kan have mange sections

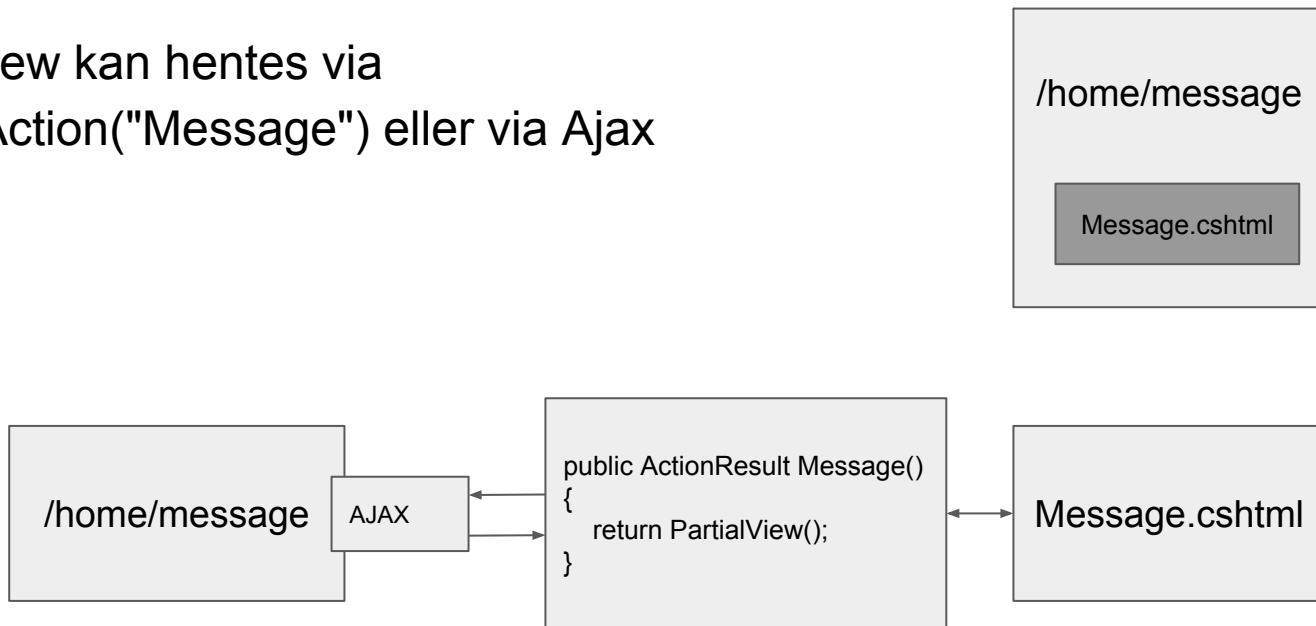
en sections bliver defineret i et view ved
f.eks at skrive

```
@section Footer {  
    This is the FOOTER  
}
```



Razor View Engine : Partial view

Partial view kan hentes via
`@Html.Action("Message")` eller via Ajax



Note:

I bogen står der at StartView laver rod i ajax kald når man henter PartialView. Dette skulle ikke være et problem mere.