

ASP.NET MVC Framework

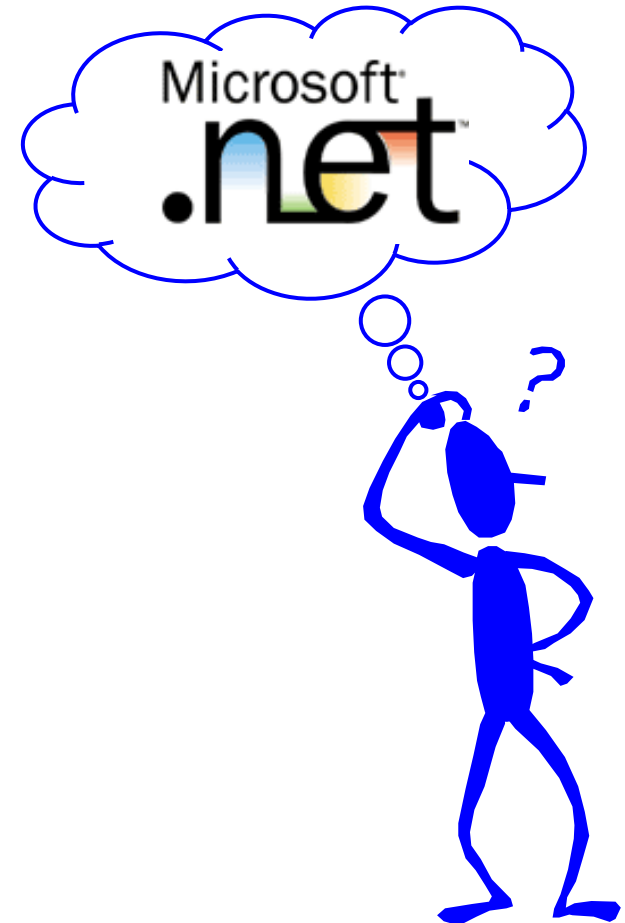
- en del af ASP.NET 3.5 Extensions

Captator

Tlf: 8620 4242
www.captator.dk

Carsten Juel Andersen

Softwarearkitekt
juel@captator.dk
Mobil: 2348 0003



- ◆ **MVC overblik**
 - Hvorfor?, Hvad?, Hvordan?
- ◆ **Fra Controller, over Model til View**
 - Sammenhængen mellem Controller, Model og View
- ◆ **URL routing**
 - Fleksibel og generel URL routing
- ◆ **Håndtering af view data**
 - Hvordan flyttes data fra Controller til View
- ◆ **Redigering af data**
 - Håndtering af Form redigering og postbacks

- ◆ **Dette foredrag er (stort set) forældet inden I kommer hjem ...**
 - Dette foredrag bygger på
 - ASP.NET 3.5 Extensions Preview af 9. dec 2007
 - MVC Toolkit
 - ADO.NET Entity Framework beta 3
- ◆ **Nyt preview i forbindelse med MIX08 (5. - 7. marts)**
 - Efter MIX08 releasen vil der formentlig komme en række CTP'er frem mod den endelige release
 - Se Scott Gu's blog (weblogs.asp.net/scottgu) med titlen: "ASP.NET MVC Framework Road-Map Update"
- ◆ **Men mon ikke koncepterne forbliver nogenlunde som de er på nuværende tidspunkt ?!**

◆ ASP.NET web forms

- bygger på en "Windows forms - weboplevelse"
- bygger på en form per side
- bygger på event mekanisme
- bygger på et kontroltræ, hvor kontroller har navn efter, hvor i træet de er placeret
- benytter ViewState
- er i modsætning til "traditionel" web udvikling, hvor siden ofte vil indeholde en form for hver inputformular på siden

◆ De fleste public websites har kun et fåtal af sider, hvor alt dette er nødvendigt

◆ Det er svært at skille især kontrol- og user interface-logik

◆ Mål med Model View Controller frameworket

- At sikre en klar adskillelse mellem model, view, controller

◆ Model

- Model / entitetsobjekter / forretningsobjekter indeholder applikationens tilstand og persisteres ofte i eksempelvis en database

◆ View

- Ansvarlig for at displaye brugergrænsefladen

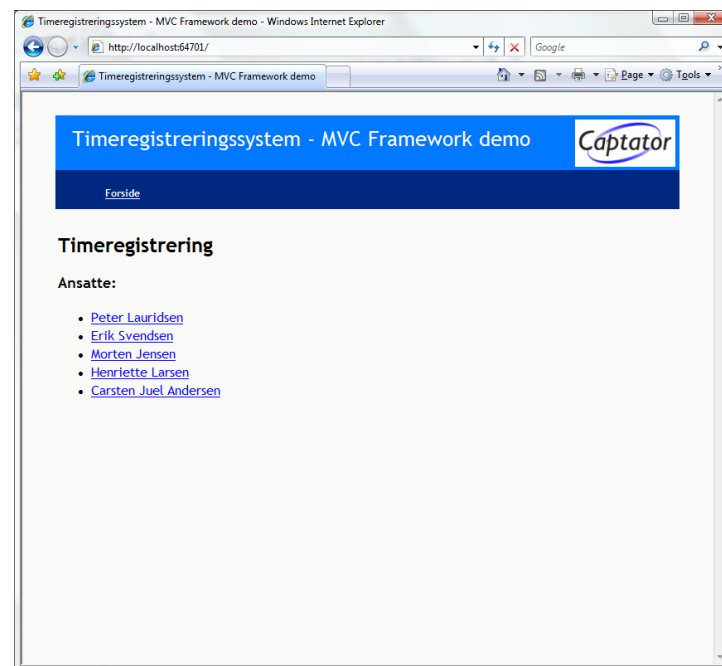
◆ Controller

- Håndterer brugerens input, manipulerer modellen og udvælger et view således at den næste side kan vises i brugergrænsefladen

- ◆ **Klar adskillelse af ansvar (model, view, controller)**
- ◆ **Optimeret for TDD baseret udvikling**
 - Test Driven Development (test-først programmering)
- ◆ **Extensible / pluggable**
- ◆ **URL-mapping komponent, der giver mulighed for pæne og gennemskuelige URL'er**
- ◆ **Eksisterende masterpages, web forms og userkontroller kan benyttes som view komponenter**
 - Postback-delen af disse kan ikke benyttes!
- ◆ **Understøtter derudover alt ikke-UI baseret i ASP.NET**
 - Caching, URL autorisering, session handling etc.

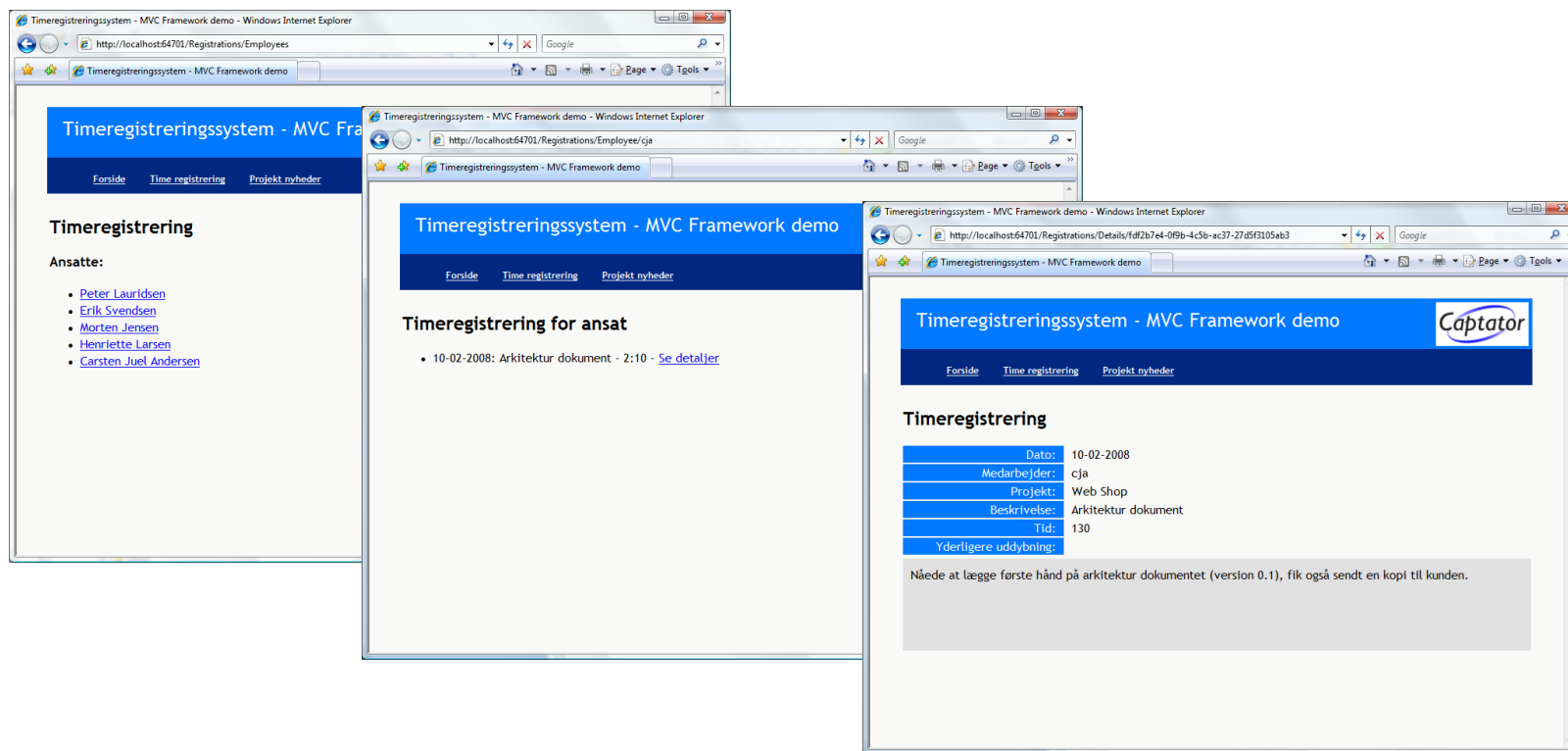
◆ Time- og projektregistereringsapplikation

- Medarbejdere (*Employee*) kan registrere udførte opgaver (*WorkRegistration*) på givne projekter (*Project*)
- Man kan kun registrere på et givet projekt, hvis man har tilladelse (*WorkAllowance*) til at arbejde på projektet
- For hvert projekt er der en diskussionsliste. En medarbejder kan oprette et nyt eller besvare et tidligere indlæg (*ProjectDiscussionEntry*)



◆ I første omgang vil vi se på

- Medarbejderliste
- Liste over den enkelte medarbejders registreringer
- Detaljevisning af registrering



◆ Der er 3 hovedmapper

◆ Controllers

- Som udgangspunkt er der en klasse for hver controller

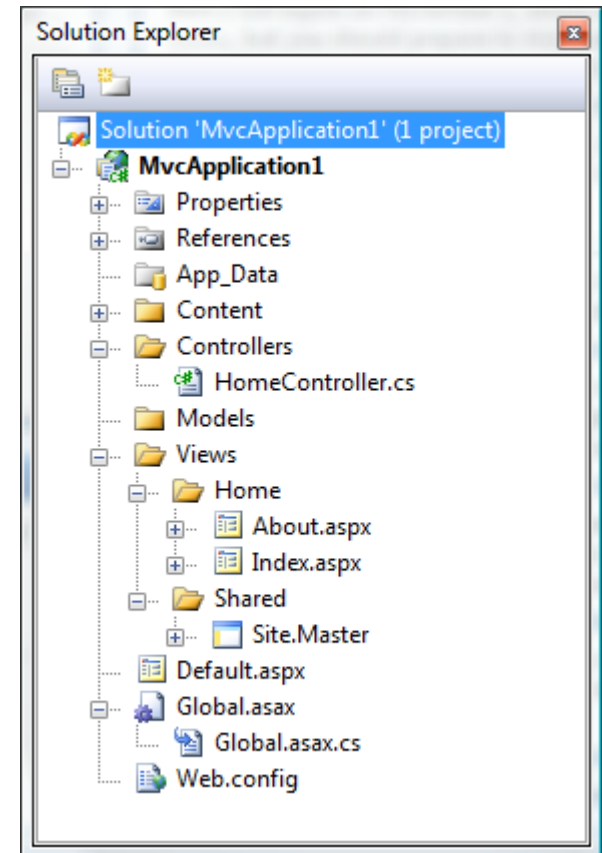
◆ Models

- En eller flere modeller - f.eks. ved brug af LINQ to SQL eller LINQ to Entities

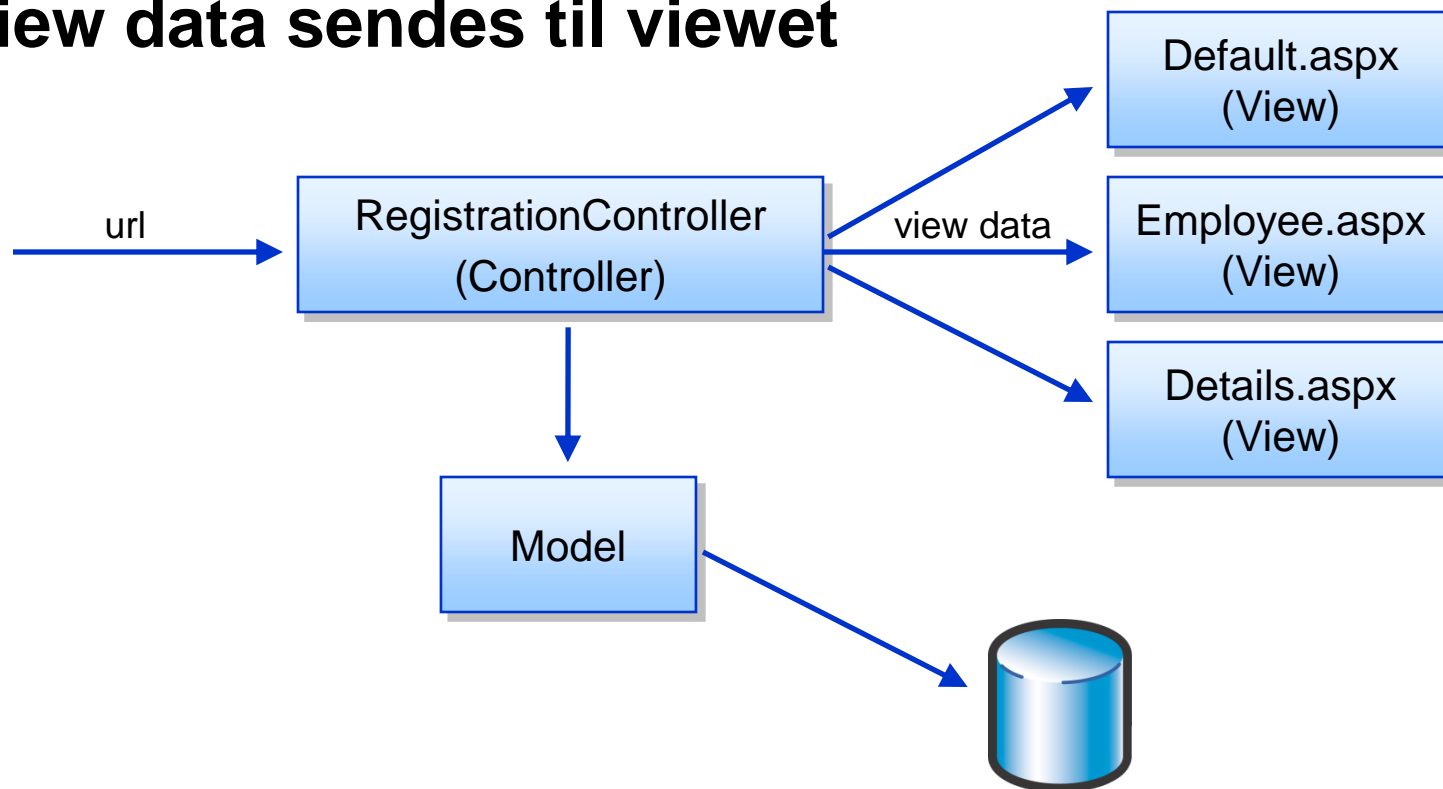
Anbefaling:
Placer model og modelhjælpeklasser
i et selvstændigt projekt

◆ Views

- Undermapper for hver controller
- "Shared" indeholder fælles views



- ◆ Først routes URLen til den matchende controller
- ◆ Controlleren trækker view data ud af modellen
 - Ud fra request parametrene
- ◆ View data sendes til viewet



- ◆ Controller navn = klassenavn, dog uden "Controller"
- ◆ Actions er metoder på controllerklassen

```
namespace MvcApplication.Controllers
```

Controller navn = "Registrations"

```
{  
    public class RegistrationsController : Controller
```

Action navn = "Employees"

```
{  
    [ControllerAction]  
    public void Employees()
```

URL = /Registrations/Employees

```
{  
    [ControllerAction]  
    public void Employee(string initials)
```

URL = /Registrations/Employee/cja

```
{  
    [ControllerAction]  
    public void Details(System.Guid id)
```

URL = /Registrations/Details/fdf2b7e4-0f9b-4c5b-ac37-27d5f3105ab3

```
{  
    }  
}
```

- ◆ En input parameter kan mappes til en variabel i en action metode på forskellig vis

```
[ControllerAction]
public void Details()
{
    System.Guid id = new System.Guid(Request["id"]);
}
```

URL = /Registrations/Details?id=fdf2b7e4-0f9b-4c5b-ac37-27d5f3105ab3

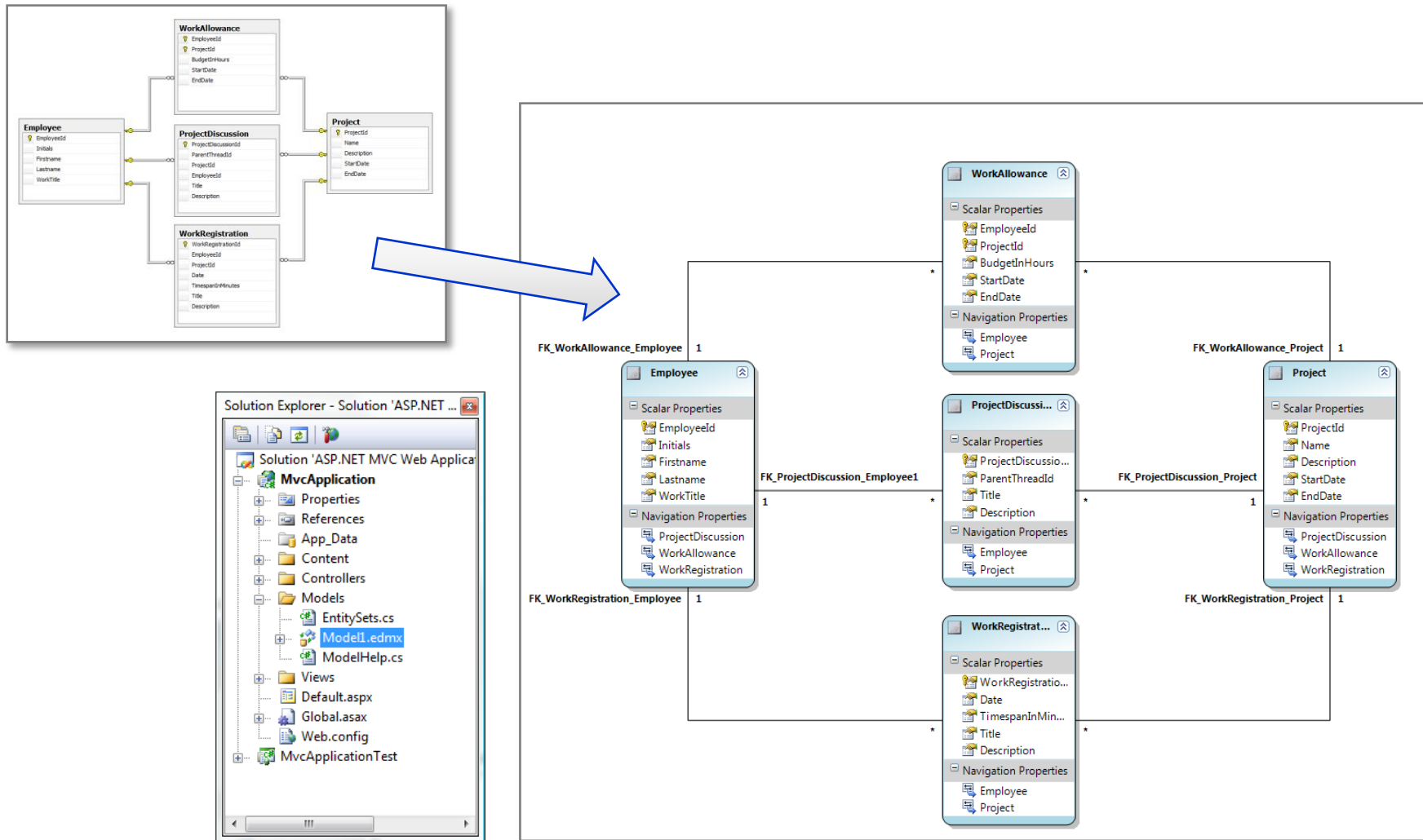
```
[ControllerAction]
public void Details(System.Guid id)
{
}
```

URL = /Registrations/Details?id=fdf2b7e4-0f9b-4c5b-ac37-27d5f3105ab3

```
[ControllerAction]
public void Details(System.Guid id)
{
}
```

URL = /Registrations/Details/fdf2b7e4-0f9b-4c5b-ac37-27d5f3105ab3

◆ Entitetsmodel genereret ud fra databasen



◆ Nu kan vi i action metoder

- hente data fra controlleren
- sende data videre til et view

```
namespace MvcApplication.Controllers
{
    public class RegistrationsController : Controller
    {
        [ControllerAction]
        public void Employees()
        {
            EmployeeList employees = null;

            using (Model.Context context = new Model.Context())
            {
                employees = new EmployeeList(context.Employee.ToList<Employee>());
            }

            RenderView("Default", employees);
        }
    }
}
```

Anbefaling:
Flyt tilgang til model (inklusive LINQ-queries)
til klasser i "Model"-namespace

- ◆ **Benyt masterpages, .aspx og .ascx**
 - MVC frameworket ligger op til at holde html-koden så simpel som muligt
 - Et view må kun have renderingslogik!

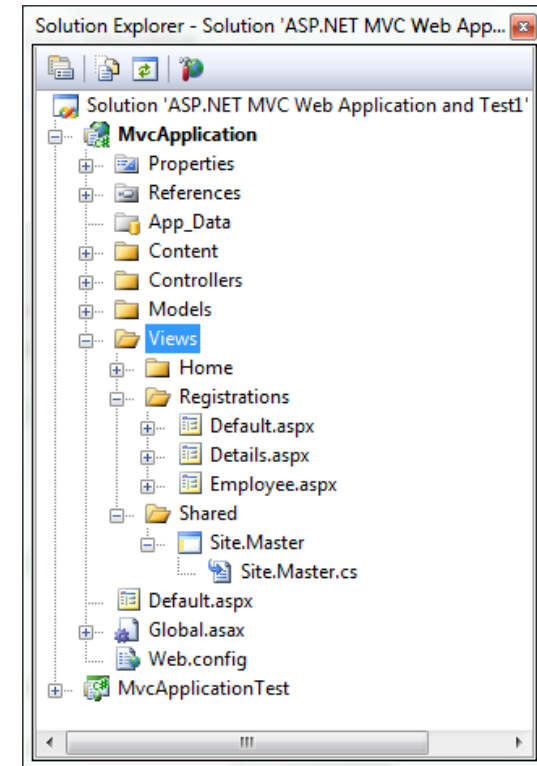
- ◆ **Der benyttes følgende superklasser**
 - Masterpage
 - System.Web.Mvc.ViewMasterPage
 - .aspx
 - System.Web.Mvc.ViewPage
 - .ascx
 - System.Web.Mvc.ViewUserControl

◆ View placeres under "Views" mappen

- Opret undermappe for hver controller (samme navn som controlleren)
- Placer delte filer i "Shared"

◆ Når controller kalder RenderView

- Vælges view i undermappe med samme navn som controller, findes den ikke her vælges view fra Shared



◆ Dette er en "out of the box" sammenkædning mellem controller og view

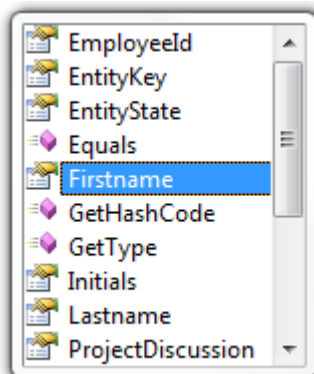
- Det er muligt at plugge sin egen håndtering af dette ind i frameworket

◆ Viewet kaldes fra RegistrationsController

```
[ControllerAction]
public void Employees()
{
    ...
    RenderView("Default", employees);
}
```

◆ Brug den generiske version af ViewPage til at angive typen for view data og få typestærk adgang der til

```
public partial class Default : ViewPage<EmployeeList>
{
    public void Page_Load()
    {
        ViewData[0].
    }
}
```



◆ 2 forskellige principper for rendering

1. via inline code

- Fordel: Typestærk adgang til view data
- Ulempe: Serverside kode og HTML "rodet sammen"

2. via code behind og ASP.NET kontroller

- Fordel: Serverside kode og HTML adskilt
- Ulempe: Databinding er typesvag

◆ Referencer til andre sider indsættes via html-hjælpemetoder, således at der ikke er hardkodede links på siden

Rendering af View – via inline code

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...  
<asp:Content ID="Content2" ContentPlaceHolderID="Main ...  
  
    <ul>  
    <% foreach (Model1.Employee emp in ViewData)  
        { %>  
        <li>  
            <%= Html.ActionLink(emp.Firstname + " " + emp.Lastname,  
                                new { Controller = "Registrations",  
                                    Action = "Employee/" + emp.Initials } )%>  
        </li>  
    <%  
        } %>  
    </ul>  
<asp:Content>
```

```
public partial class Default :  
    ViewPage<EmployeeList>  
{  
}  
}
```

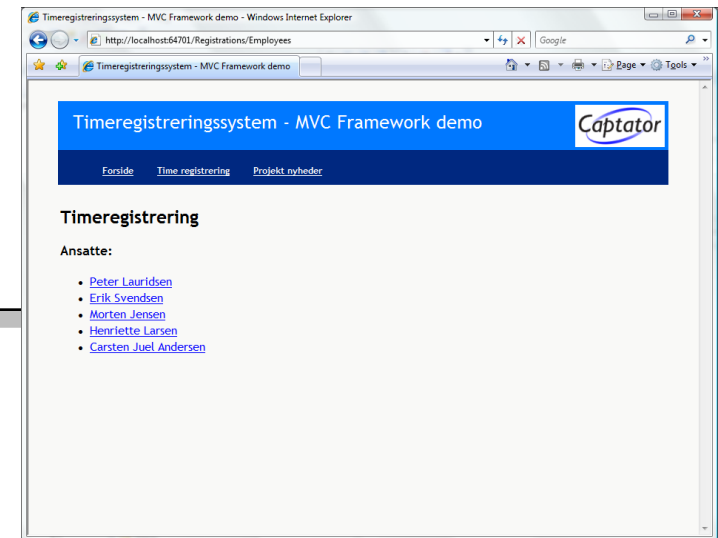


- ◆ Output er simpel html uden <form> tags
- ◆ Ingen "lange WebForms ID'er"

```
...  
<div id="maincontent">  
  <h2>Timeregistrering</h2>  
  <h3>Ansatte:</h3>  
  <ul>  
    <li><a href="/Registrations/Employee/pla">Peter Lauridsen</a></li>  
    <li><a href="/Registrations/Employee/esv">Erik Svendsen</a></li>  
    <li><a href="/Registrations/Employee/moj">Morten Jensen</a></li>  
    <li><a href="/Registrations/Employee/hla">Henriette Larsen</a></li>  
    <li><a href="/Registrations/Employee/cja">Carsten Juel Andersen</a></li>  
  </ul>  
</div>  
...
```

◆ aspx med brug af Repeater

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...  
  
<asp:Content ID="Content2" ContentPlaceHolderID="Main ...  
  <asp:Repeater ID="repViewEmployees" runat="server"  
    OnItemDataBound="repViewEmployees_ItemDataBound">  
    <HeaderTemplate>  
      <ul>  
    </HeaderTemplate>  
    <ItemTemplate>  
      <li>  
        <asp:Literal ID="litLink" runat="server" />  
      </li>  
    </ItemTemplate>  
    <FooterTemplate>  
      </ul>  
    </FooterTemplate>  
  </asp:Repeater>  
</asp:Content>
```



◆ ... og den tilsvarende codebehind

```
public partial class Default : ViewPage<EmployeeList>
{
    public void Page_Load()
    {
        repViewEmployees.DataSource = ViewData;
        repViewEmployees.DataBind();
    }

    protected void repViewEmployees_ItemDataBound(
        object sender, System.Web.UI.WebControls.RepeaterItemEventArgs e)
    {
        if ((e.Item.ItemType == ListItemType.Item) ||
            (e.Item.ItemType == ListItemType.Item) )
        {
            System.Web.UI.WebControls.Literal litLink =
                (System.Web.UI.WebControls.Literal)FindControl("litLink");
            Employee emp = (Employee)e.Item.DataItem;
            litLink.Text = Html.ActionLink(
                emp.Firstname + " " + emp.Lastname,
                new { Controller = "Registrations",
                    Action = "Employee/" + emp.Initials });
        }
    }
}
```

- ◆ **HtmlHelper er del af MVCToolkit**
 - HtmlHelper kan tilgås i Views via Html property
- ◆ **ActionLink benytter den opsatte routing af controllere og actions til at skabe et link herudfra**
- ◆ **Eksempler på brug**
 - Kald parameterløs action på den kaldende controller

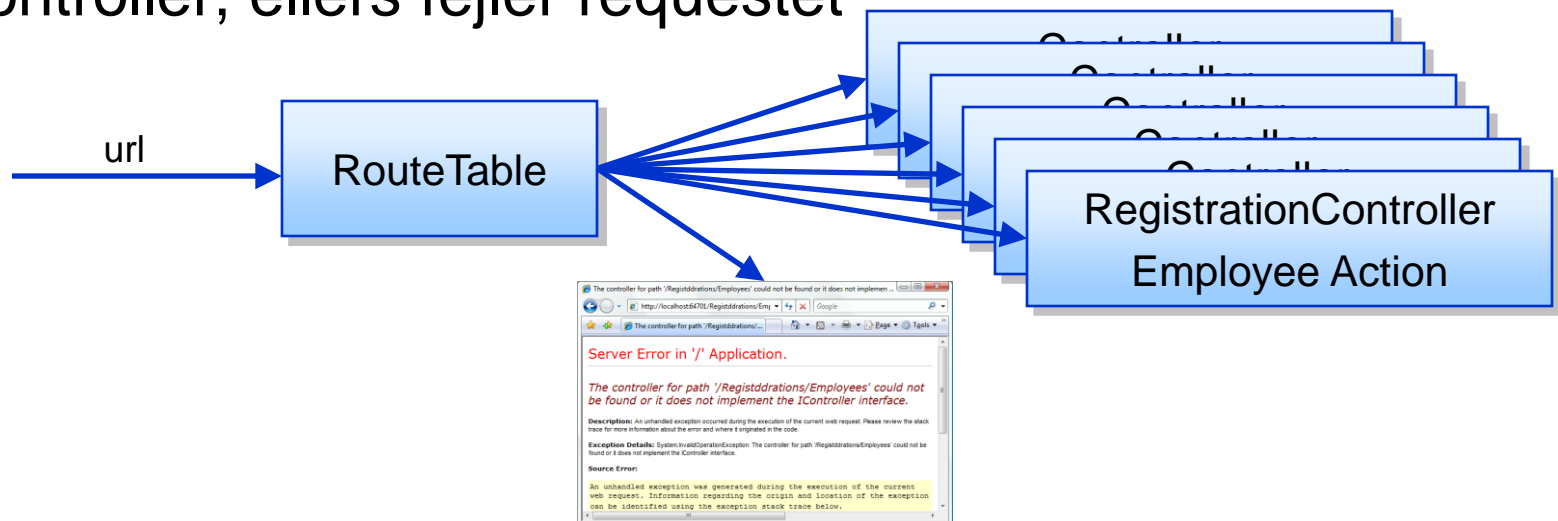
```
Html.ActionLink("linktekst", "actionName");
```

- Kald vilkårlig controller og action evt. med parametre
 - Her benyttes en overload med en anonym klasse

```
Html.ActionLink("linktekst",  
    new { Controller = "Registrations",  
          Action = "Employee",  
          Initials = "cja" } );
```

◆ MVC indeholder en fleksibel URL routing mekanisme

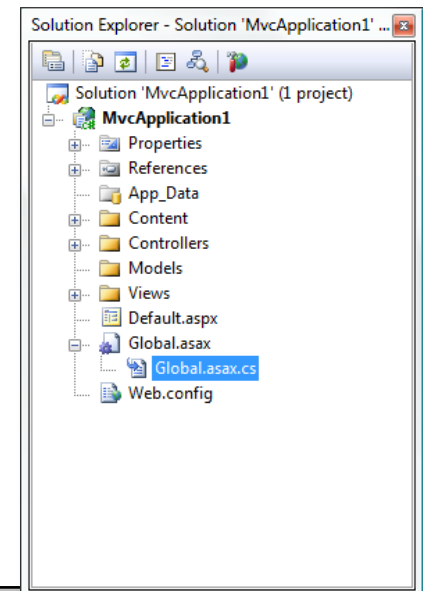
- RouteTable (liste af routing regler) opbygges ved applikationens opstart
- Når et request modtages sendes det gennem routing systemet
- Hvis der findes et match kaldes den matchende controller, ellers fejler requestet



◆ Default placering for opsætning af routing er i Global.asax.cs

```
public class Global : System.Web.HttpApplication
{
    protected void Application_Start(object sender, EventArgs e)
    {
        RouteTable.Routes.Add(new Route
        {
            Url = "[controller]/[action]/[id]",
            Defaults = new { action = "Index", id = (string)null },
            RouteHandler = typeof(MvcRouteHandler)
        });

        RouteTable.Routes.Add(new Route
        {
            Url = "Default.aspx",
            Defaults = new { controller = "Home",
                            action = "Index",
                            id = (string)null },
            RouteHandler = typeof(MvcRouteHandler)
        });
    }
}
```



◆ En Url specifikation kan være

- en eksakt Url

```
Url = "Min/Side"
```

- en templated Url

```
Url = "[controller]/[action]/[id]"
```

- eller en kombination

```
Url = "Min/[action]/[controller]"
```

◆ En Route skal kunne fastlægge controller og action

- ved at [controller] og [action] indgår i URL
- eller ved at controller og action defineres som Defaults

```
RouteTable.Routes.Add(new Route
{
    Url = "Min/Side",
    Defaults = new { controller = "MinController",
                    action = "Side" },
    RouteHandler = typeof(MvcRouteHandler)
});
```

◆ Templated Url

```
url = "[controller]/[action]/[id]"
```

- "[tekst]" angiver en parameter
- Ovenstående vil blandt andet matche følgende

```
/Home/Default    - Hvis der findes en HomeController med en Default action og  
                  en default værdi for "id" der er null
```

```
/Registration/Employee/20  
                  - Hvis der findes en HomeController med en Employee action
```

◆ Med følgende Route og Url

```
new Route
{
    Url = "[controller]/[action]/[id]",
    Defaults = new { action = "Index", id = (string)null },
    RouteHandler = typeof(MvcRouteHandler)
});
```

/Registration/Employee/20

◆ Vil Route skabe et RouteData objekt

```
public class RouteData
{
    public RouteData();
    public Route Route { get; set; }
    public IDictionary<string, object> Values { get; }
}
```

◆ Med en IDictionary indeholdende

```
"Controller" = "Registration"
"Action" = "Employee"
"Id" = 20
```

◆ RequestContext sendes til en RouteHandler

```
public class RequestContext
{
    public RequestContext(IHttpContext httpContext, RouteData routeData);
    public IHttpContext HttpContext { get; internal set; }
    public RouteData RouteData { get; internal set; }
}
```

```
public interface IRouteHandler
{
    IHttpHandler GetHttpHandler(RequestContext requestContext);
}
```

◆ MvcRouteHandler kalder den konkrete action i den givne controller

```
public class MvcRouteHandler : IRouteHandler
{ ... }
```

◆ Routing systemet er fleksibelt og kan benyttes i non-MVC scenarier

- Implementer blot egen RouteHandler

◆ Eksempel på brug af adskillige parametre

```
public new Route
{
    Url = "Registrations/Employee/[initials]/[year]/[month]/[day]",
    Defaults = new { controller = "Registrations", action = "Employee",
                    year = (int?)null, month = (int?)null,
                    day = (int?)null },
    RouteHandler = typeof(MvcRouteHandler)
};
```

◆ Giver mulighed for at følgende URLe routes ...

/Registrations/Employee/cja	- alle registreringer på cja
/Registrations/Employee/cja/2008	- alle reg. på cja i 2008
/Registrations/Employee/cja/2007/12	- alle reg. på cja i december 2007
/Registrations/Employee/cja/2008/02/20	- alle reg. på cja i dag

◆ ... til følgende action

```
[ControllerAction]
public void Employee(string initials, int? year, int? month, int? day)
{
    ...
}
```

- ◆ De enkelte tokens (parametre) i Url'en kan også valideres

```
new Route
{
    Url = "Registrations/Employee/[initials]/[year]/[month]/[day]",
    Defaults = new { controller = "Registrations", action = "Employee",
                    year = (int?)null, month = (int?)null,
                    day = (int?)null },
    Validation = new { initials = @"\p{L} {3,4}" }
    RouteHandler = typeof(MvcRouteHandler)
});
```

- ◆ Dette resulterer i at "initials" skal være 3-4 bogstaver (letters) for, at denne Route kan benyttes

- ◆ For at routing reglerne kan være fuldstændigt afkoblede, er det også nødvendigt at kunne oprette links ud fra de samme regler

- ◆ Benyt `HtmlHelper` instansen til links

- `Html.ActionLink`

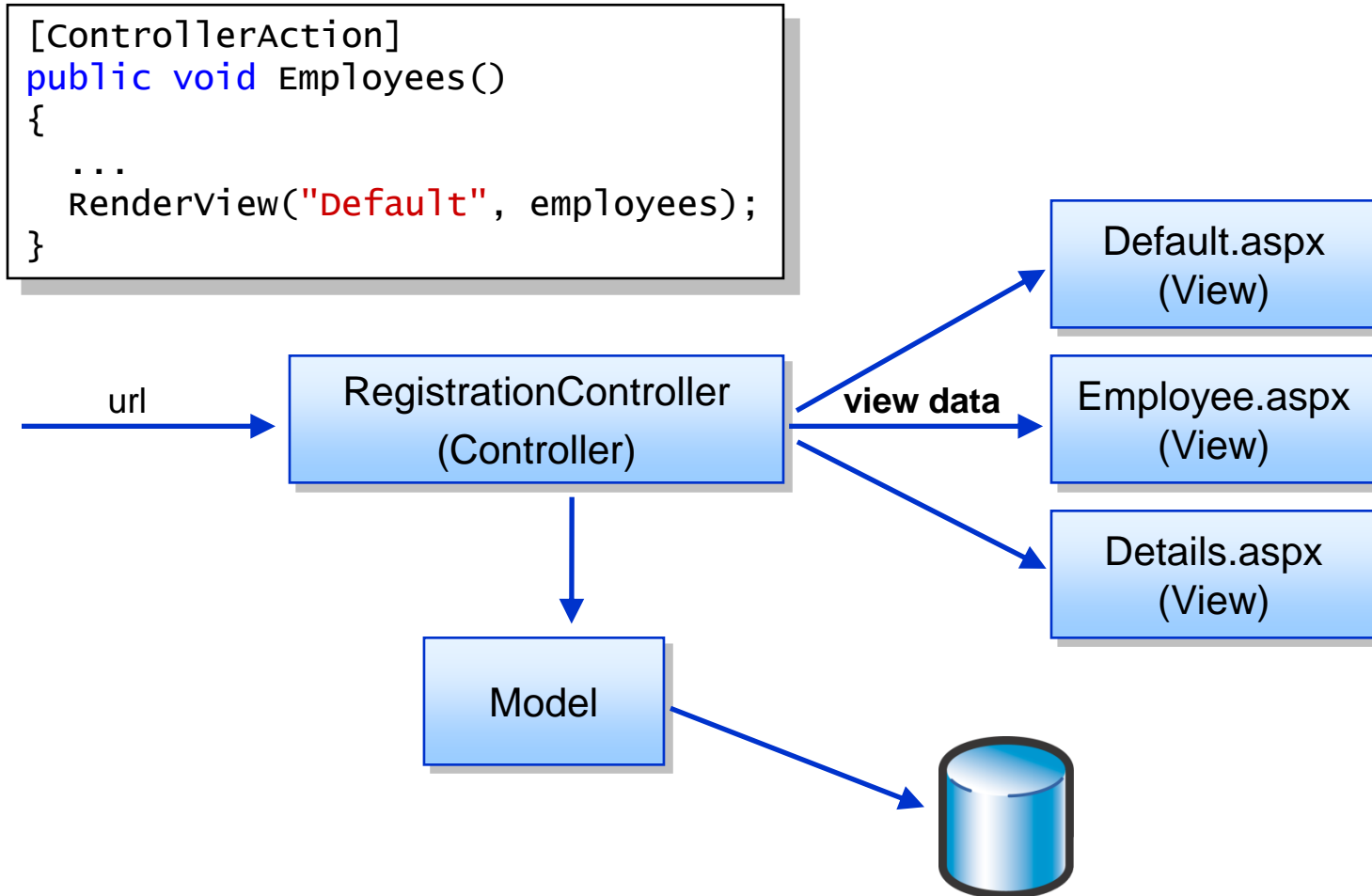
```
Html.ActionLink("linktekst",  
                new { controller = "Home", action = "Default" } );
```

- ◆ Benyt `Controller.RedirectToAction` til at redirigere til en anden side

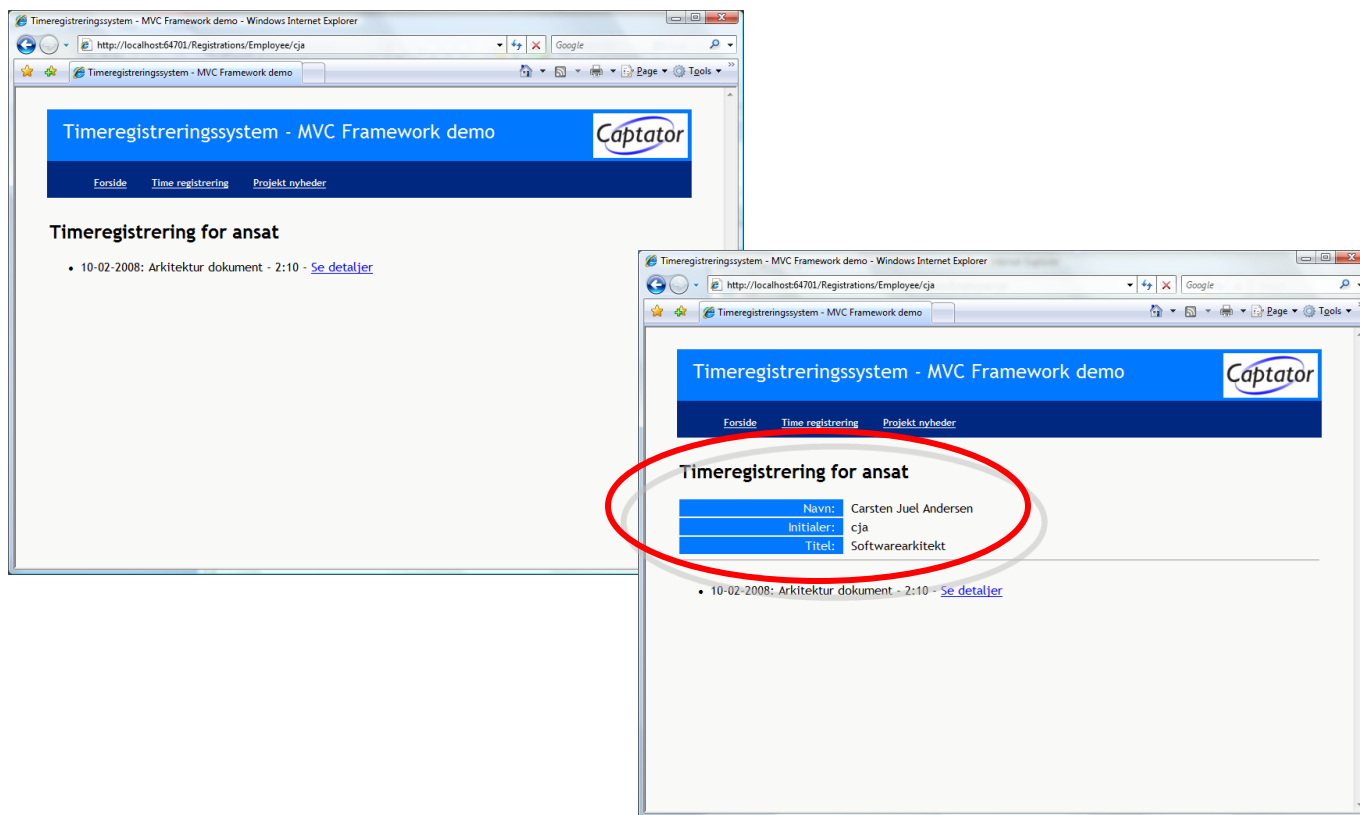
- `RedirectToAction` udløser en `Response.Redirect`

```
RedirectToAction( new { controller="Home" action="Default" } );
```


- ◆ Views må kun bero på de data Controller leverer
- ◆ ViewData overføres fra Controller til View



- ◆ Siden med timerregistreringer for ansat skal udvides til også at vise stamoplysninger om ansat



◆ I controlleren kan ViewData tilføjes typesvagt ...

```
[ControllerAction]
public void Employee(string initials, int? year, int? month, int? day)
{
    // Her hentes data fra modellen

    ViewData["Employee"] = employee;
    ViewData["Registrations"] = registrations;

    RenderView("Employee");
}
```

◆ ... til følgende view

```
public partial class Employee : ViewPage
{ }
```

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...
...
Navn: <%= ((Employee)ViewData["Employee"]).Firstname %>
...
<% foreach (Model1.WorkRegistration wr in
                ((WorkRegistrationList)ViewData["Registrations"]))
{ %> ...
```

◆ Ved hjælp af et specifikt ViewData-objekt ...

```
public class EmployeeViewData
{
    public Model.Employee Employee { get; set; }
    public Model.WorkRegistrationList RegList { get; set; }
}
```

◆ ... kan ViewData tilføjes typestærkt i controlleren ...

```
[ControllerAction]
public void Employee(string initials, int? year, int? month, int? day)
{
    // Her hentes data fra modellen

    RenderView("Employee",
        new Views.Registrations.EmployeeViewData
            { Employee = employee, RegList = registrations } );
}
```

- ◆ og ved at tilføje ViewData-klassen som typeparameter på viewet

```
public partial class Employee : ViewPage<EmployeeViewData>
{ }
```

- ◆ ... er der typestærk adgang til ViewData:

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...
...
    Navn: <%= ViewData.Employee.Firstname %>
    ...
    <ul>
    <% foreach (Model1.WorkRegistration wr in ViewData.RegList)
    { %>
        <li><%= wr.Date.ToShortDateString() %> : <%= wr.Title %> -
            <%= ConvertToReadableTime(wr.TimespanInMinutes) %> -
            <%= Html.ActionLink("Se detaljer",
                new { Action="Details", Id=wr.WorkRegistrationId }) %>
        </li>
    } %>
    </ul>
    ...
```

- ◆ **UserControls kan benyttes til at minimere dubleret kode, når dele af layout/funktionalitet går igen fra side til side** (det er der jo ikke noget nyt i)
- ◆ **ViewUserController indsættes på et view nøjagtigt som UserControls indsættes i web forms**

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...  
<%@ Register src="EmployeeDetailsControl.ascx"  
            tagname="EmployeeDetailsControl" tagprefix="uc1" %>  
...  
<uc1:EmployeeDetailsControl ID="EmployeeDetailsControl1"  
                            ViewDataKey="Employee" runat="server" />  
...
```

- ◆ **ViewData kan være identisk med sidens ...**
- ◆ **... eller være en udsnit af de oprindelige ViewData**
 - Hvis ViewDataKey property er sat

Anbefaling:
Gør ViewUserController typestærk
og benyt ViewDataKey

◆ Gør ViewData på ViewUserController typestærk ...

```
public partial class EmployeeDetailsControl : ViewUserController<Employee>
{ }
```

◆ ... således at der er typestærk tilgang til data:

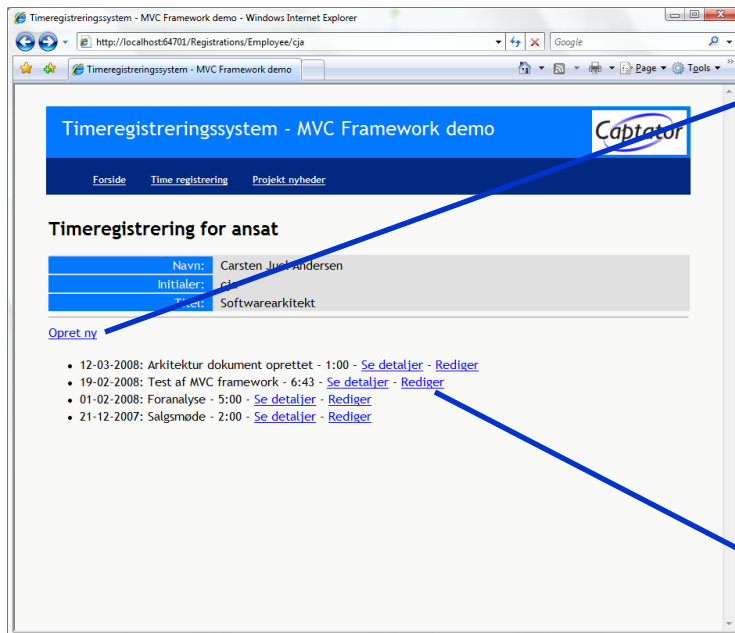
```
<%@ Control Language="C#" CodeBehind="EmployeeDetailsControl.aspx.cs" ...
Navn: <%= ViewData.Firstname %> <%= ViewData.Lastname%>
...
```

◆ På viewet sikres, at Employee sendes til ViewUserController ved at sætte ViewDataKey

```
...
<uc1:EmployeeDetailsControl ID="EmployeeDetailsControl1"
                             ViewDataKey="TheEmployee" runat="server" />
...
```

- "TheEmployee" skal være en property på viewets ViewData af typen Employee

◆ Opret ny timeregistrering



Timeregistreringssystem - MVC Framework demo

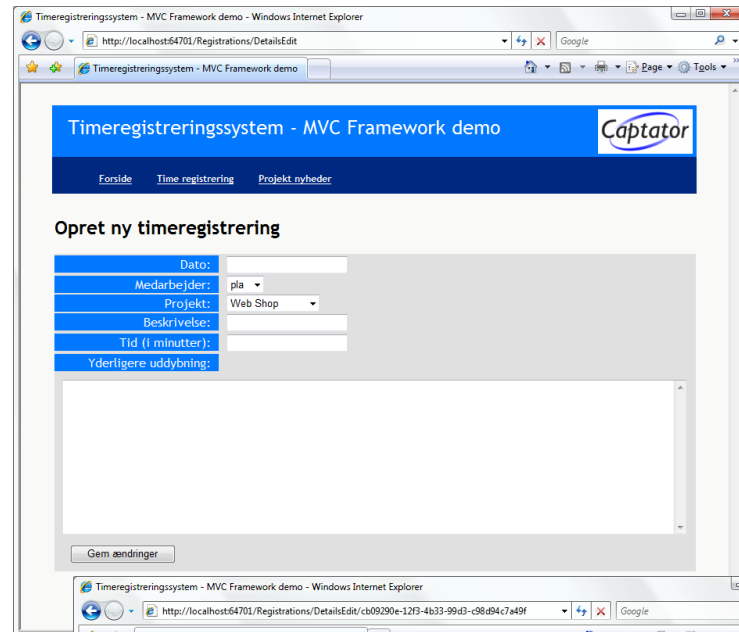
Forside Time registrering Projekt nyheder

Timeregistrering for ansat

Navn: Carsten Juul Andersen
Initialer: cja
Projekt: Softwarearkitekt

[Opret ny](#)

- 12-03-2008: Arkitektur dokument oprettet - 1:00 - [Se detaljer](#) - [Rediger](#)
- 19-02-2008: Test af MVC framework - 6:43 - [Se detaljer](#) - [Rediger](#)
- 01-02-2008: Foranalyse - 5:00 - [Se detaljer](#) - [Rediger](#)
- 21-12-2007: Salgsmøde - 2:00 - [Se detaljer](#) - [Rediger](#)



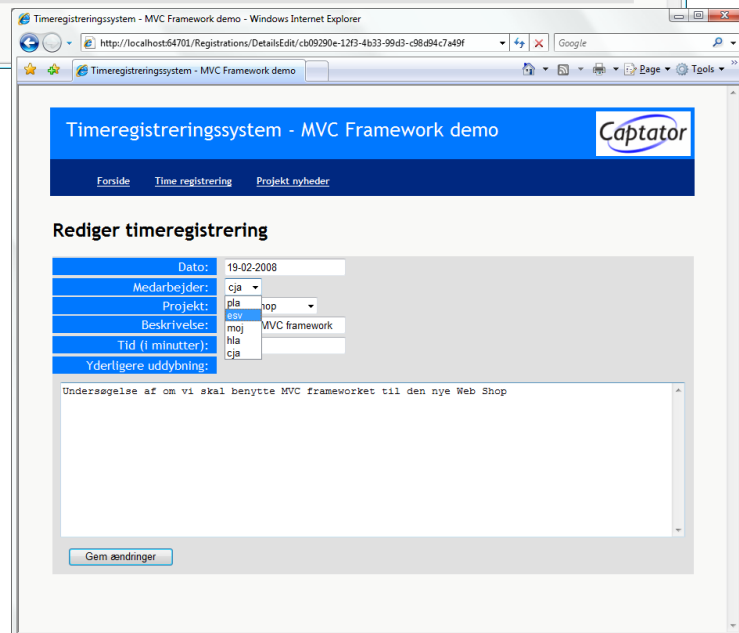
Timeregistreringssystem - MVC Framework demo

Forside Time registrering Projekt nyheder

Opret ny timeregistrering

Dato:
Medarbejder:
Projekt:
Beskrivelse:
Tid (i minutter):
Yderligere uddybning:

◆ Redigering af eksisterende timeregistreringer



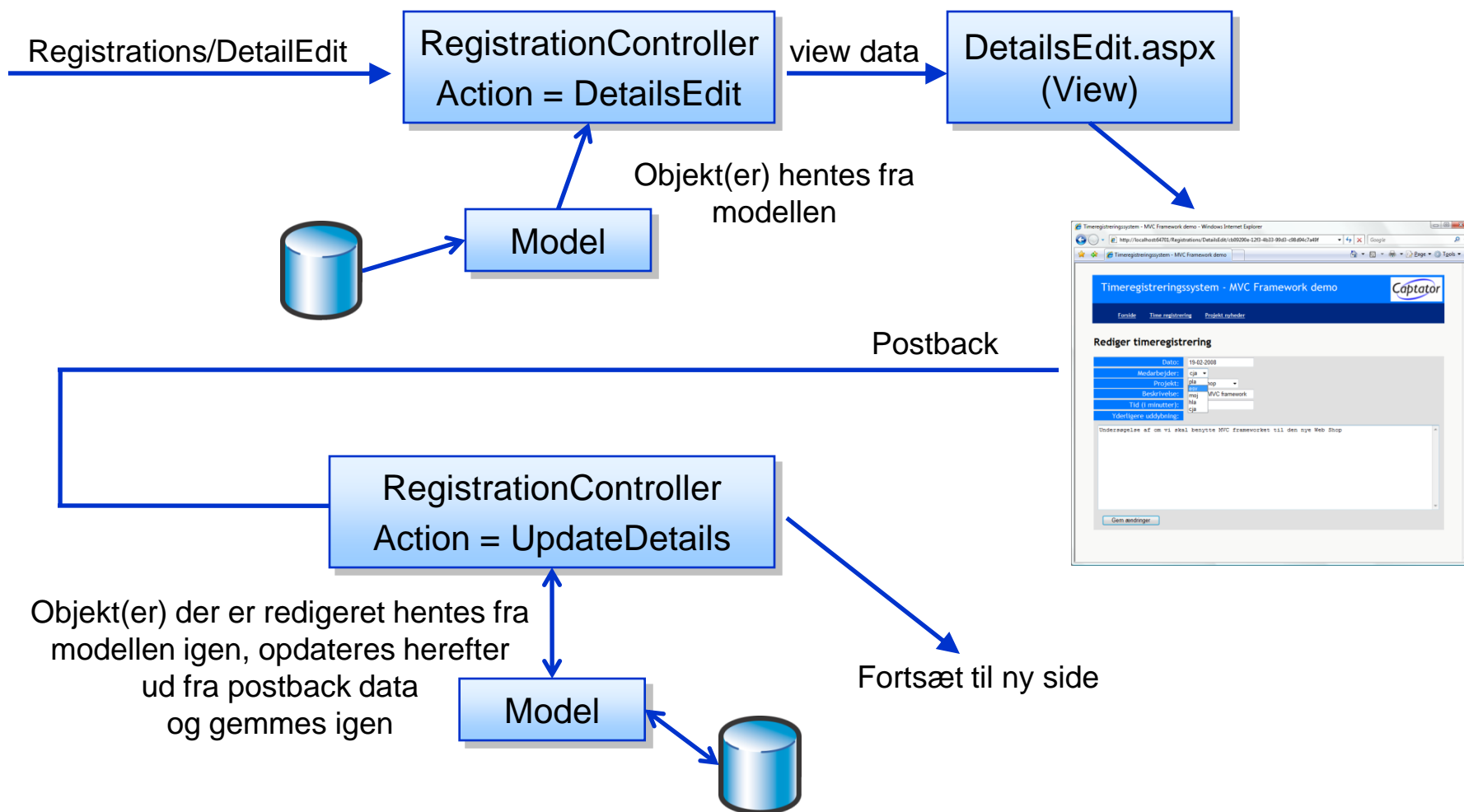
Timeregistreringssystem - MVC Framework demo

Forside Time registrering Projekt nyheder

Rediger timeregistrering

Dato: 19-02-2008
Medarbejder:
Projekt:
Beskrivelse:
Tid (i minutter):
Yderligere uddybning:

◆ En action skaber data - en anden opdaterer data



◆ Først oprettes links til opret og rediger kontrollere:

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...  
...  
<%= Html.ActionLink("Opret ny", new { Action="DetailsEdit" }) %>  
...  
<% foreach (Model1.WorkRegistration wr in ViewState.RegList)  
    { %> ...  
<%= Html.ActionLink("Rediger", new { Action="DetailsEdit",  
                                         Id=wr.WorkRegistrationId }) %>  
<% } %> ...
```

◆ DetailsEdit action

```
[ControllerAction]  
public void DetailsEdit(System.Guid id)  
{  
    WorkRegistration wr = // Hentes fra modellen  
  
    RenderView("DetailsEdit", wr);  
}
```

◆ DetailsEdit oprettes med typestærk ViewData

```
public partial class DetailsEdit : ViewPage<WorkRegistration>
{ }
```

◆ DetailsEdit.aspx

```
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" ...
...
<h2>Rediger timeregistrering</h2>

<form action="/Registrations/UpdateDetails/<%= ViewData.WorkRegistrationId
%>" method="post">

    Dato: <%= Html.TextBox("Date", ViewData.Date.ToShortDateString()) %><br />
    Beskrivelse: <%= Html.TextBox("Title", ViewData.Title) %><br />

    <input type="submit" value="Gem ændringer" />
</form>
...
```

form action med URL, der
peger på UpdateDetails action

Id'en på textboxen skal være have
samme navn som propertyen, der redigeres

◆ Html uden ViewState og "lange id'er"

```
...  
  
<h2>Rediger timeregistrering</h2>  
  
<form action="/Registrations/UpdateDetails/cb09290e-12f3-4b33-99d3-  
c98d94c7a49f" method="post">  
  
    Dato: <input id="Date" name="Date" size="20" value="19-02-2008"><br />  
    Beskrivelse:<input id="Title" name="Title" size="20" value="Test af MVC  
framework"><br />  
    <input type="submit" value="Gem ændringer" />  
</form>  
  
...
```

◆ Update action

- Form og get-parametre mappes automatisk til input parametre på action

```
[ControllerAction]
public void UpdateDetails(System.Guid id, System.DateTime date,
                          int timespanInMinutes, string title, string description)
{
    WorkRegistration wr;

    wr = // Hent eksisterende objekt fra modellen
    wr.Date = date.Date;
    wr.TimespanInMinutes = timespanInMinutes;
    wr.Title = title;
    wr.Description = description;

    // Gem objektet i modellen

    RedirectToAction("Employees");
}
```

- ◆ Vi mangler at håndtere lister ...
- ◆ ... så der oprettes en dedikeret ViewData klasse ...

```
public class WorkRegistrationEditViewData
{
    public WorkRegistration Wr { get; set; }
    public EmployeeList Employees { get; set; }
    public ProjectList Projects { get; set; }
}
```

- ◆ ... og i controller action'en DetailsEdit tilføjes lister

```
[ControllerAction]
public void DetailsEdit (System.Guid id)
{
    WorkRegistrationEditViewData wrEdit = new WorkRegistrationEditViewData();

    wrEdit.Wr =          // Hent WorkRegistration objekt fra modellen
    wrEdit.Employees =   // Hent alle Employees
    wrEdit.Projects =    // Hent alle Projects

    RenderView("DetailsEdit", wrEdit);
}
```

◆ Brug af HtmlHelper-metoden Select

```
...  
<form action="/Registrations/UpdateDetails/<%=ViewData.Wr.WorkRegistrationId.  
ToString() %>" method="post">  
...  
Dato: <%= Html.TextBox("Date", ViewData.Wr.Date.ToShortDateString())%><br />  
Medarbejder: <%= Html.Select("EmployeeId", ViewData.Employees,  
    (object)ViewData.Wr.Employee.EmployeeId.ToString())%><br />  
...  
    <input type="submit" value="Gem ændringer" />  
</form>  
...
```

◆ I den anvendte overload tager Html.Select imod

- Id
- Listen, der skal displays
- Det valgte elements key

◆ Controller action'en DetailsEdit tilrettes, så input parameteren id bliver nullable

```
[ControllerAction]
public void DetailsEdit (System.Guid? id)
{
    WorkRegistrationEditViewData wrEdit = new WorkRegistrationEditViewData();

    if (id.HasValue)
    {
        wrEdit.Wr =          // Hent WorkRegistration objekt fra modellen
    }
    wrEdit.Employees = // Hent alle Employees
    wrEdit.Projects =  // Hent alle Projects

    RenderView("DetailsEdit", wrEdit);
}
```

- Wr-property på ViewData sættes til null, hvis det drejer sig om oprettelse af ny WorkRegistration

- ◆ For at lette koden i viewets ascx-template oprettes en IsNew property i code behind

```
public partial class DetailsEdit : ViewPage<WorkRegistrationEditViewData>
{
    protected bool IsNew;
    public void Page_Load()
    {
        IsNew = (ViewData.Wr == null);
    }
}
```

- ◆ Udsnit af viewets ascx-template

- Der sættes tomme strenge ind i input, hvis det er opret:

```
<% if (IsNew) { %><h2>Opret ny timeregistrering</h2>
<% } else      { %><h2>Rediger timeregistrering</h2><% } %>

<form action="/Registrations/UpdateDetails/<%= (IsNew) ?
    "" : ViewData.Wr.WorkRegistrationId.ToString() %>" method="post">
    Dato: <%= Html.TextBox("Date", (IsNew) ?
        "" : ViewData.Wr.Date.ToShortDateString())%><br />
    ...
```

◆ Update action tilrettes, så den kan håndtere opret

```
[ControllerAction]
public void UpdateDetails(System.Guid? id, System.DateTime date,
                          int timespanInMinutes, string title, string description)
{
    WorkRegistration wr;

    if (id.HasValue)
        wr = // Hent eksisterende objekt fra modellen
    else
    {
        wr = new Model1.WorkRegistration();
        wr.WorkRegistrationId = System.Guid.NewGuid();
    }
    wr.Date = date.Date;
    wr.TimespanInMinutes = timespanInMinutes;
    wr.Employee = // Hent employee fra modellen ud fra id
    wr.Project = // Hent project fra modellen ud fra id
    wr.Title = title;
    wr.Description = description;

    // Gem objekt i modellen

    RedirectToAction("Employees");
}
```

◆ anvend UpdateFrom(Request.Form)

- Mapper form-parametre direkte ind på model objekt

```
[ControllerAction]
public void UpdateDetails(System.Guid? id)
{
    WorkRegistration wr;

    if (id.HasValue)
        wr = // hent eksisterende objekt fra model
    else
    {
        wr = new Model1.WorkRegistration();
        wr.WorkRegistrationId = System.Guid.NewGuid();
    }

    wr.UpdateFrom(Request.Form);

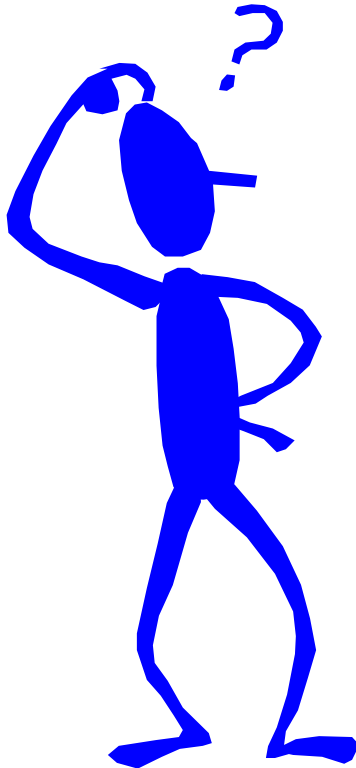
    // Hack - det ser ikke ud til at referencer håndteres af UpdateFrom
    //          det må være en fejl i den nuværende version
    wr.Employee = // Hent employee ud fra key fra
                  //   new System.Guid(Request.Form["employeeId"]));
    wr.Project = // Hent på tilsvarende vis project

    // Gem objekt i modellen og foretag en redirect
    ...
}
```

- ◆ **MVC Framework er et alternativ til web forms**
 - Ikke en afløser!

- ◆ **MVC har klare styrker så længe, det drejer sig om visning af data**
 - JavaScript, CSS m.v. bliver lettere tilgængelig, når sideopbygning er simplere (simplere ID'er m.v.)
 - Den klare opdeling mellem forretningslogik og UI-del

- ◆ **Web forms har klart fordele (endnu i hvert tilfælde) når det kommer til redigering og kompleks brugerinteraktion**



www.captator.dk

nyheder, artikler, information, ...