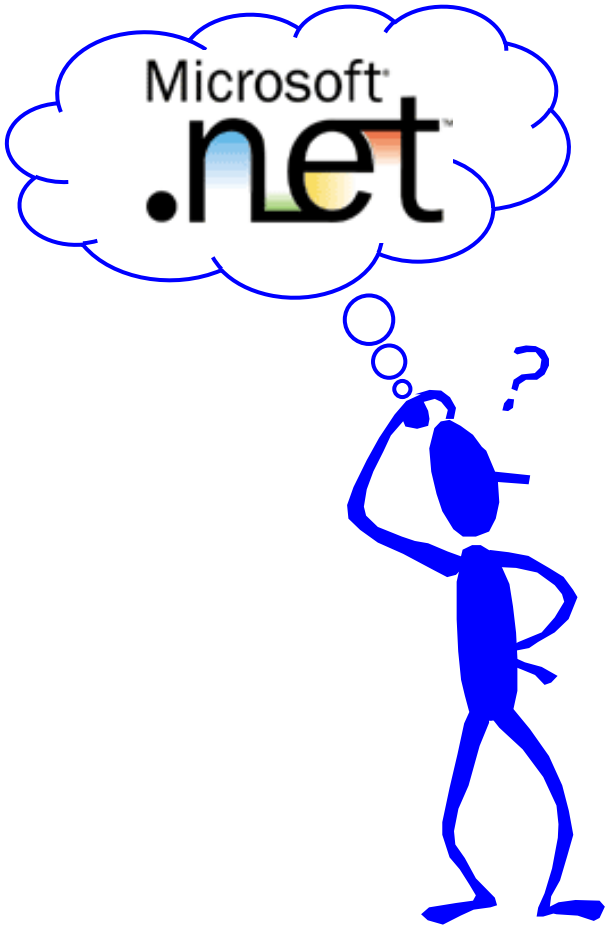
Tjek den nye ASP.NET-platform

- ASP.NET Model View Controller (MVC) Framework

Foredrag for SAM-DATA
marts 2010



Captator
Tlf: 8620 4242
www.captator.dk

Carsten Juel Andersen
Softwarearkitekt
juel@captator.dk
Mobil: 2348 0003

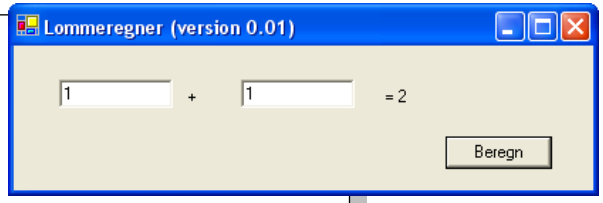
Agenda

- **”Gode gamle” ASP.NET Webforms**
- **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

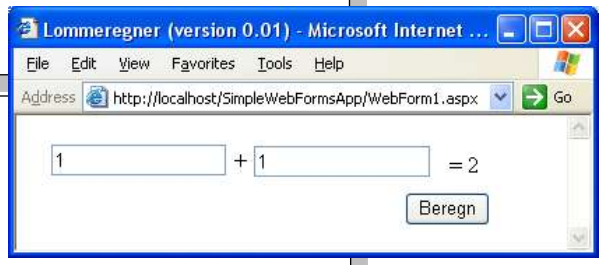
Web Forms - Windows forms déjà vu

- **Designes som windows forms i Visual Studio**
- **Har samme event-model som Windows forms**
- **Fungerer på .NET kode-præmisser - ikke på html-præmisser**

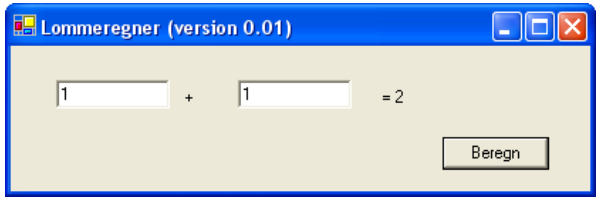
```
public partial class Form1 : System.Windows.Forms.Form
{
  private void btnBeregn_Click(object sender, System.EventArgs e)
  {
    try {
      lblResult.Text = (double.Parse(txtValue1.Text) + double.Parse(txtValue2.Text)).ToString();
    }
    catch {
      lblResult.Text = "Fejl i indtastning ";
    }
} }
```

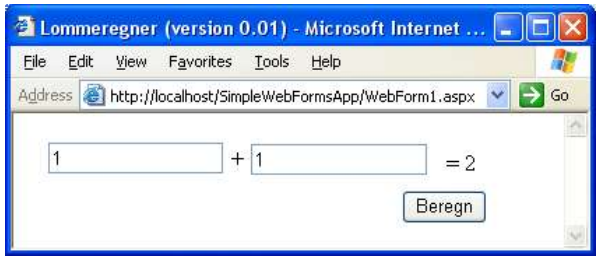


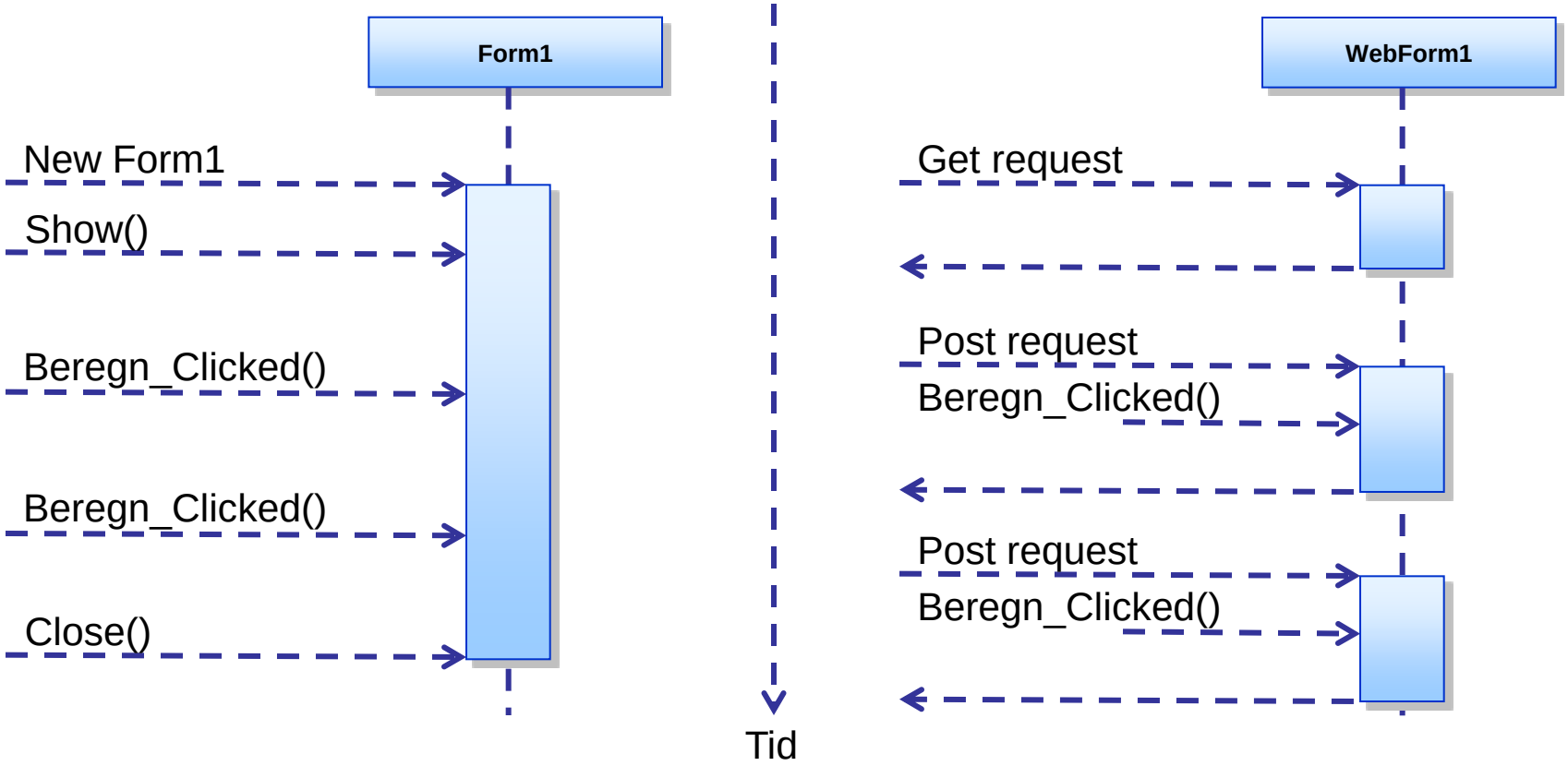

```
public partial class _Default : System.Web.UI.Page
{
private void btnBeregn_Click(object sender, System.EventArgs e)
  {
    try {
      lblResult.Text = (double.Parse(txtValue1.Text) + double.Parse(txtValue2.Text)).ToString();
    }
    catch {
      lblResult.Text = "Fejl i indtastning"
    }
} }
```



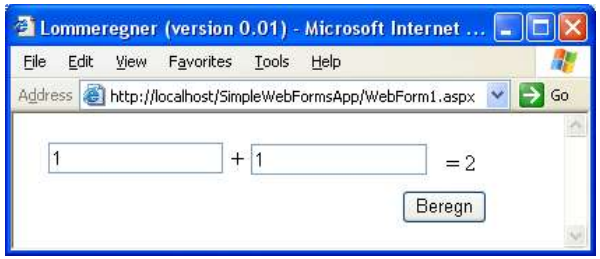
Windows forms kontra Web Forms



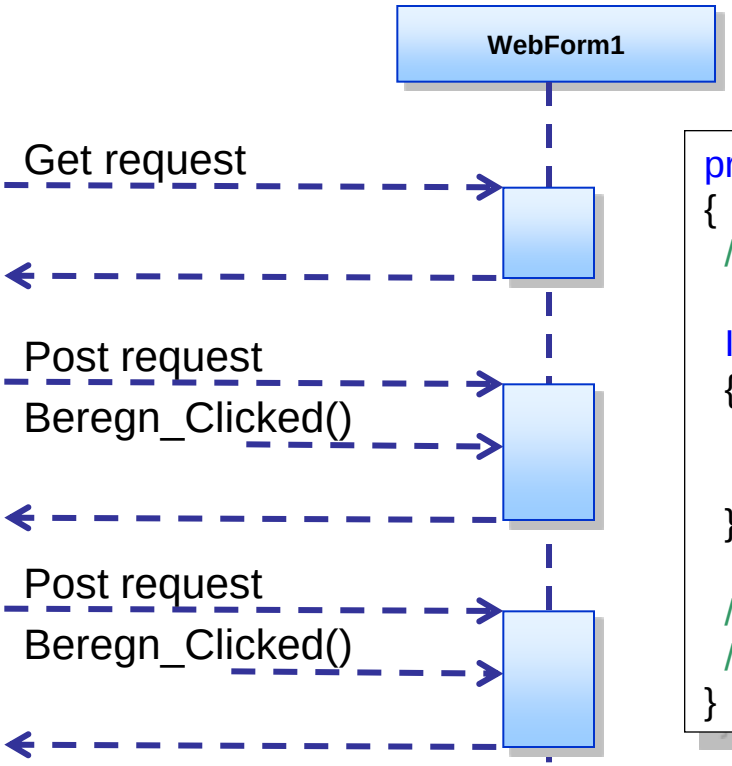




Web Forms - IsPostBack



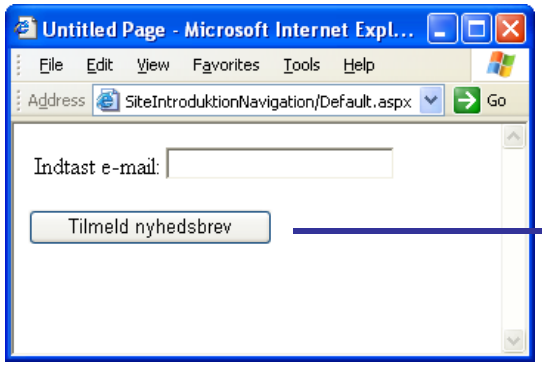
- **Da en web form gendannes forfra, hver gang et event fyres, skal dynamisk indhold, der <u>ikke</u> er UI-bundet, loades hver gang**

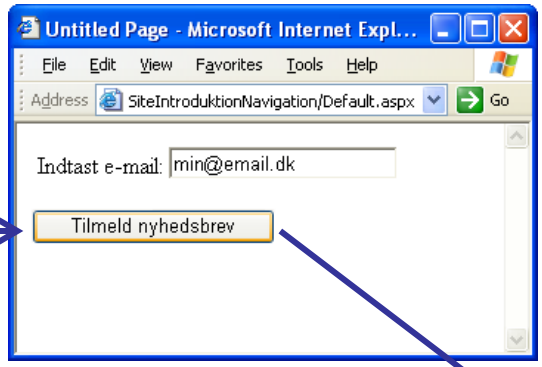


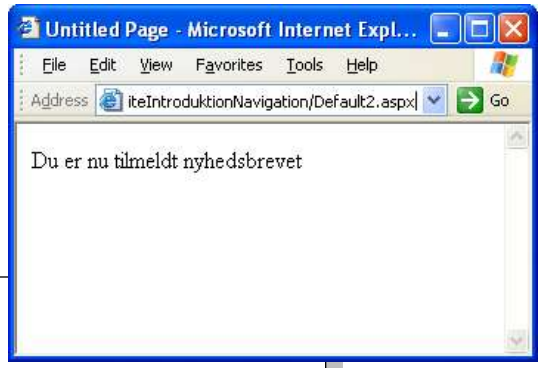

```
private void Page_Load( ... )
{
  // Put user code to initialize the page here

  If (!IsPostBack)
  {
    // Load dynamisk indhold,
    // der bindes til UI kontroller
  }

  // Load dynamisk indhold, der ikke er
  // "bundet" til en UI kontrol hver gang
}
```

Web Forms navigation

- **ASP.NET servermodellen er baseret på, at postback foretages til den samme side**







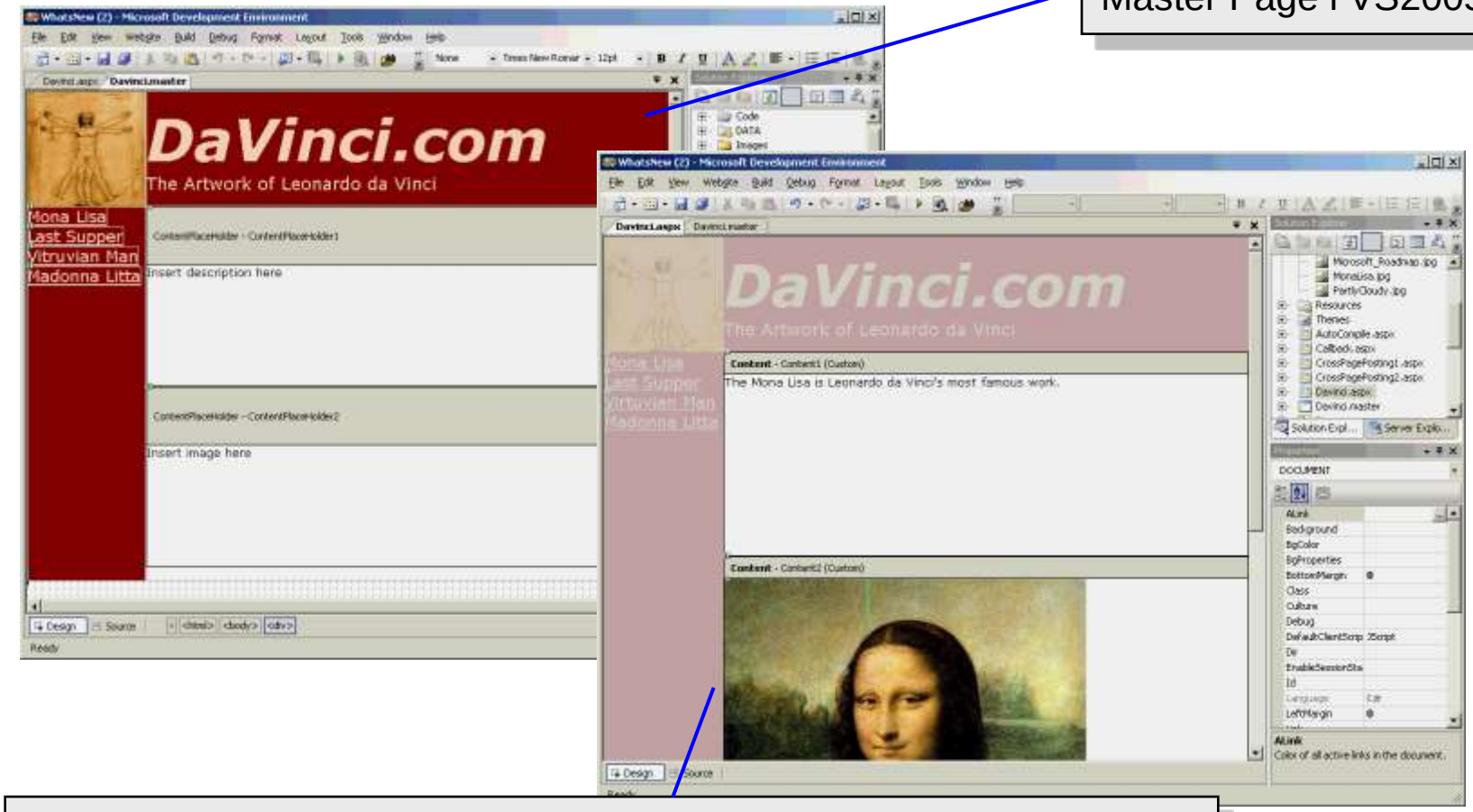
- **Ofte anvendes redirect til at navigere til næste side**

```
public partial class _Default : System.Web.UI.Page
{
  protected void Button1_Click(object sender, System.EventArgs e)
  {
    // gem e-mail adresse ...
    this.Response.Redirect("~/Default2.aspx");
  }
}
```

Master Pages

- **Template baseret side opbygning**

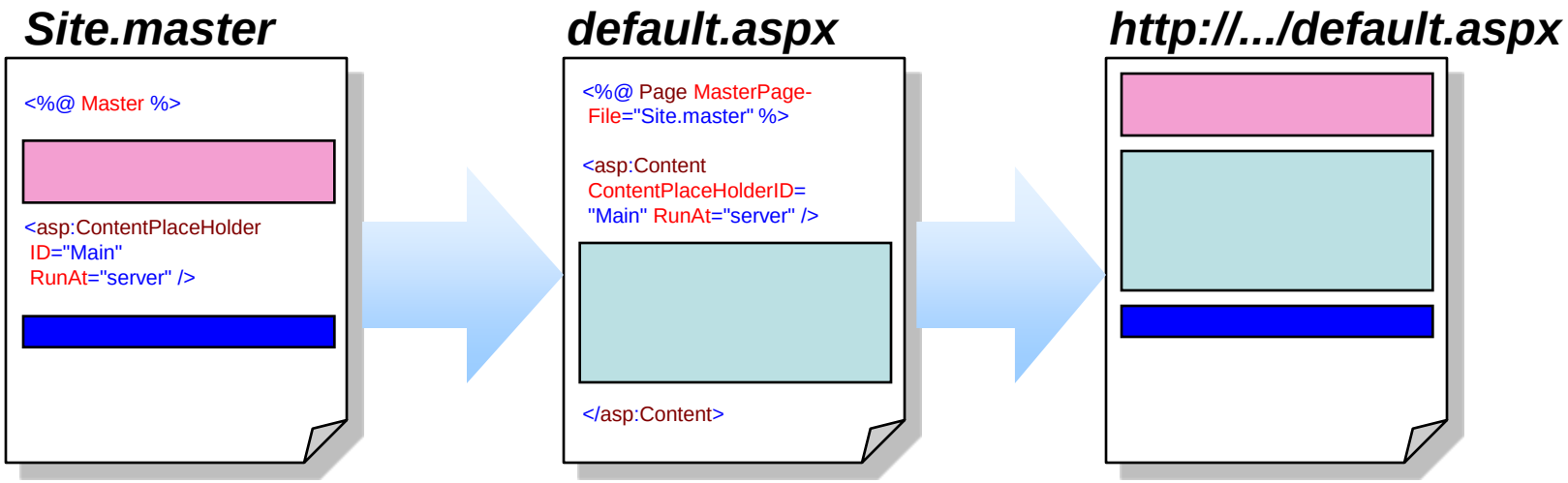
Designview af en Master Page i VS2005



En side, der baserer sig på den givne Master Page. Læg mærke til at Master Page delen er nedtonet (som footer/header visning i Word)

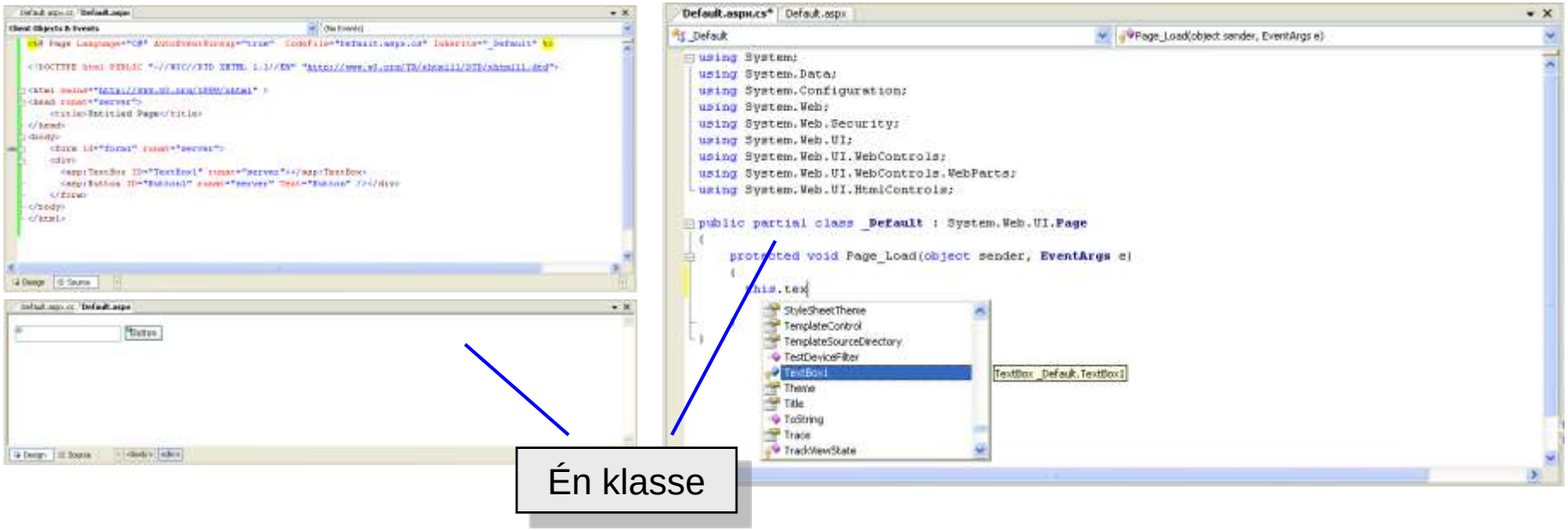
Master Pages

- **Masters definerer en template for en række sider**
  - Fælles indhold defineres på Master Pagen
  - Sidens indhold markeres med <asp:ContentPlaceHolder>
- **Content pages**
  - refererer til en given master
  - fylder de enkelte ContentPlaceHolders ud med indhold ved brug af <asp:Content>



Separation af html og serverside kode

- **Design og kodedel i partielle klasser**



  - Design-kontroller er defineret i designer-delen
  - Event registrering er defineret i designer-delen
  - En række properties oprettes af design-delen
    - Også tilgængelige på edit-time i VS

Sammenhæng – ASPX og code-file

- **ASPX indeholder et page direktiv**
  - Angiver betingelser til brug for generering af designer partiel klasse

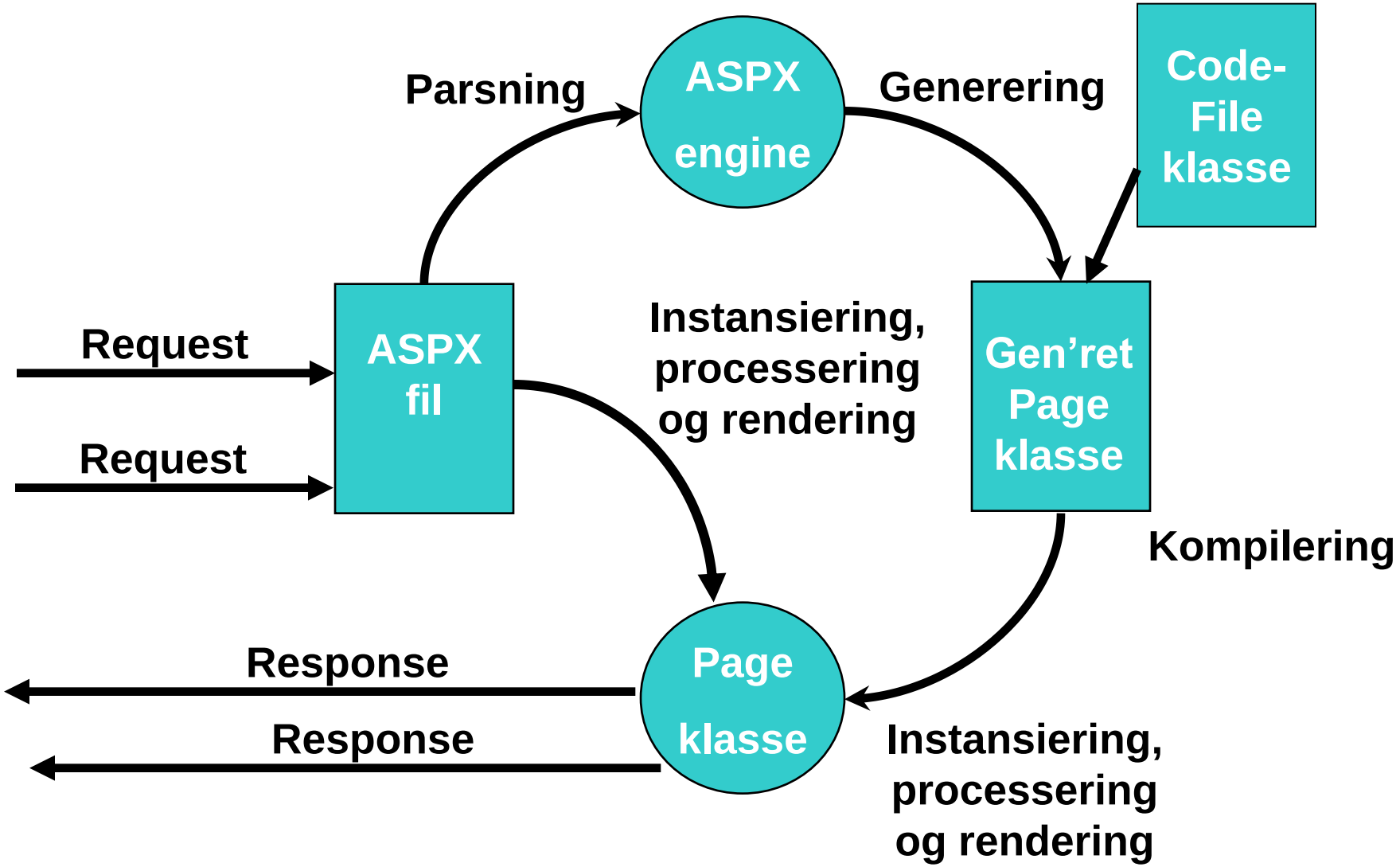
- **Page direktiv angiver (blandt andet) sammenhængen mellem ASPX og CodeFile**

```
<%@ Page Language="C#" AutoEventWireup="true"
   CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" ...
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
...
```

CodeFile – fysisk fil

Inherits – partiel klasse navn

Runtime oversættelse af ASPX



Events og eventregistrering

- **Page events**
  - AutoEventWireup=”true” -> navnesammenfald
    - Page_Load, Page_PreRender etc
- **Øvrige kontroller**
  - Navn på eventmetode i xml-markup

```
<%@ Page Language="C#" AutoEventWireup="true"
        CodeFile="Default.aspx.cs" Inherits="_Default" %>
...
  <asp:Button ID="Button1" runat="server" Text="Button”
            OnClick="Button1_Click" />
...
```

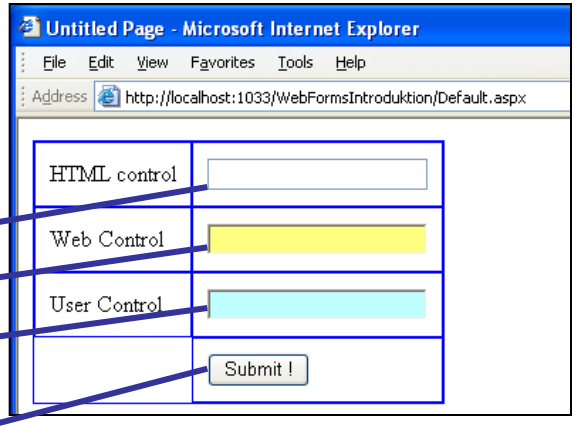
```
public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, System.EventArgs e)
  { ... }

  protected void Button1_Click(object sender, System.EventArgs e)
  { ... }
}
```

Page og kontrol objekt model

- **Mens siden er under opbygning, findes en objekt model, der afspejler HTML siden**

Control Tree

Control UniqueID	Type
__Page	ASP.default_aspx
ctl02	System.Web.UI.LiteralControl
ctl00	System.Web.UI.HtmlControls.HtmlHead
ctl01	System.Web.UI.HtmlControls.HtmlTitle
ctl03	System.Web.UI.LiteralControl
form1	System.Web.UI.HtmlControls.HtmlForm
ctl04	System.Web.UI.LiteralControl
htmlinput1	System.Web.UI.HtmlControls.HtmlInputText
ctl05	System.Web.UI.LiteralControl
TextBox1	System.Web.UI.WebControls.TextBox
ctl06	System.Web.UI.LiteralControl
WebUserControlWithTextBox1	ASP.webusercontrolwithtextbox_ascx
WebUserControlWithTextBox1$TextBox1	System.Web.UI.WebControls.TextBox
WebUserControlWithTextBox1$ctl00	System.Web.UI.LiteralControl
ctl07	System.Web.UI.LiteralControl
Button1	System.Web.UI.WebControls.Button
ctl08	System.Web.UI.LiteralControl
ctl09	System.Web.UI.LiteralControl



- Runat="server" kontroller får alle en unik id bestående af kontrolnavn samt path
- Den unikke id sikrer, at events samt indtastede data på websiden returneres til den korrekte kontrol serverside

MVC framework overblik

- **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

MVC framework design principper

- **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.

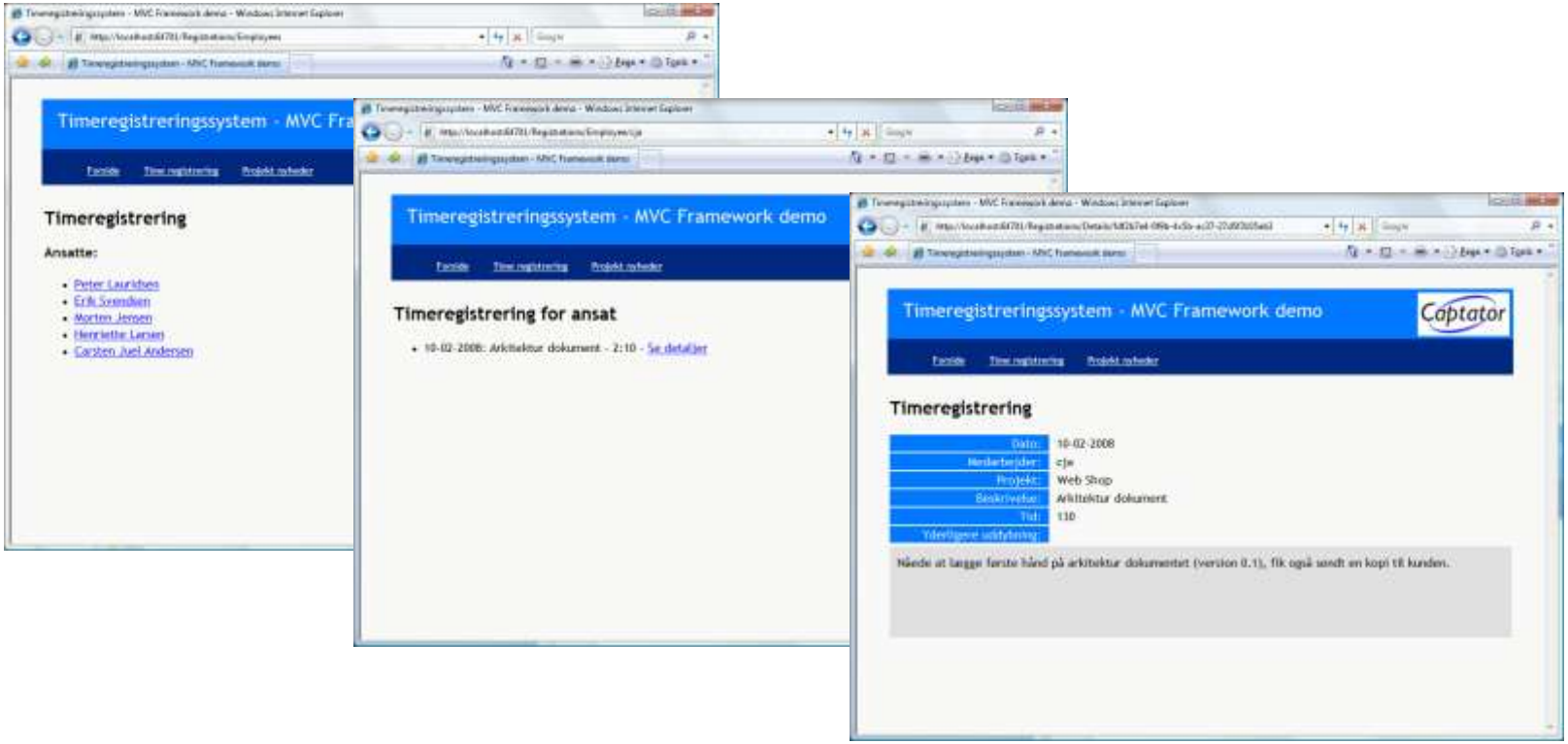
Overblik over MVC demoen

- **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*)

Overblik over den første MVC demo

- **I første omgang vil vi se på**
  - Medarbejderliste
  - Liste over den enkelte medarbejders registreringer
  - Detaljevisning af registrering

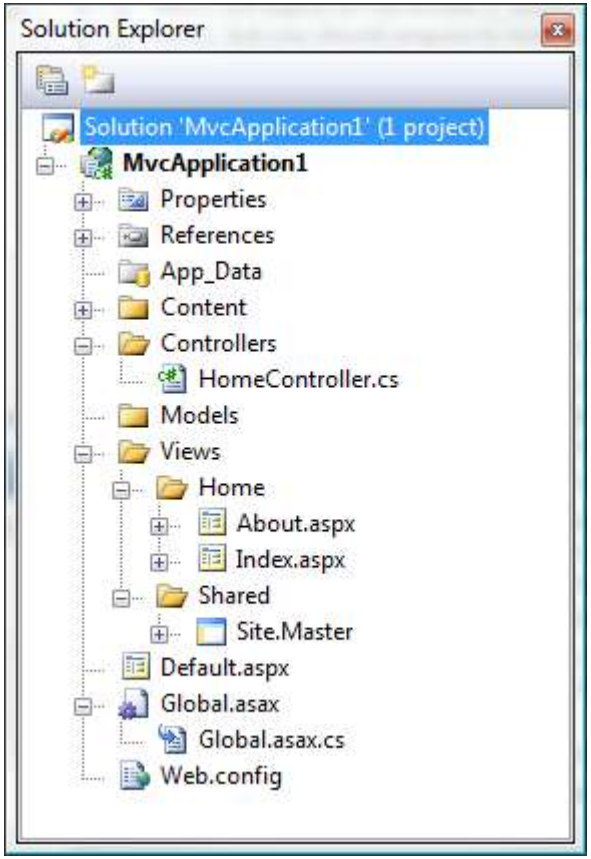


Folderstruktur af ASP.NET MVC Applikation

- **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

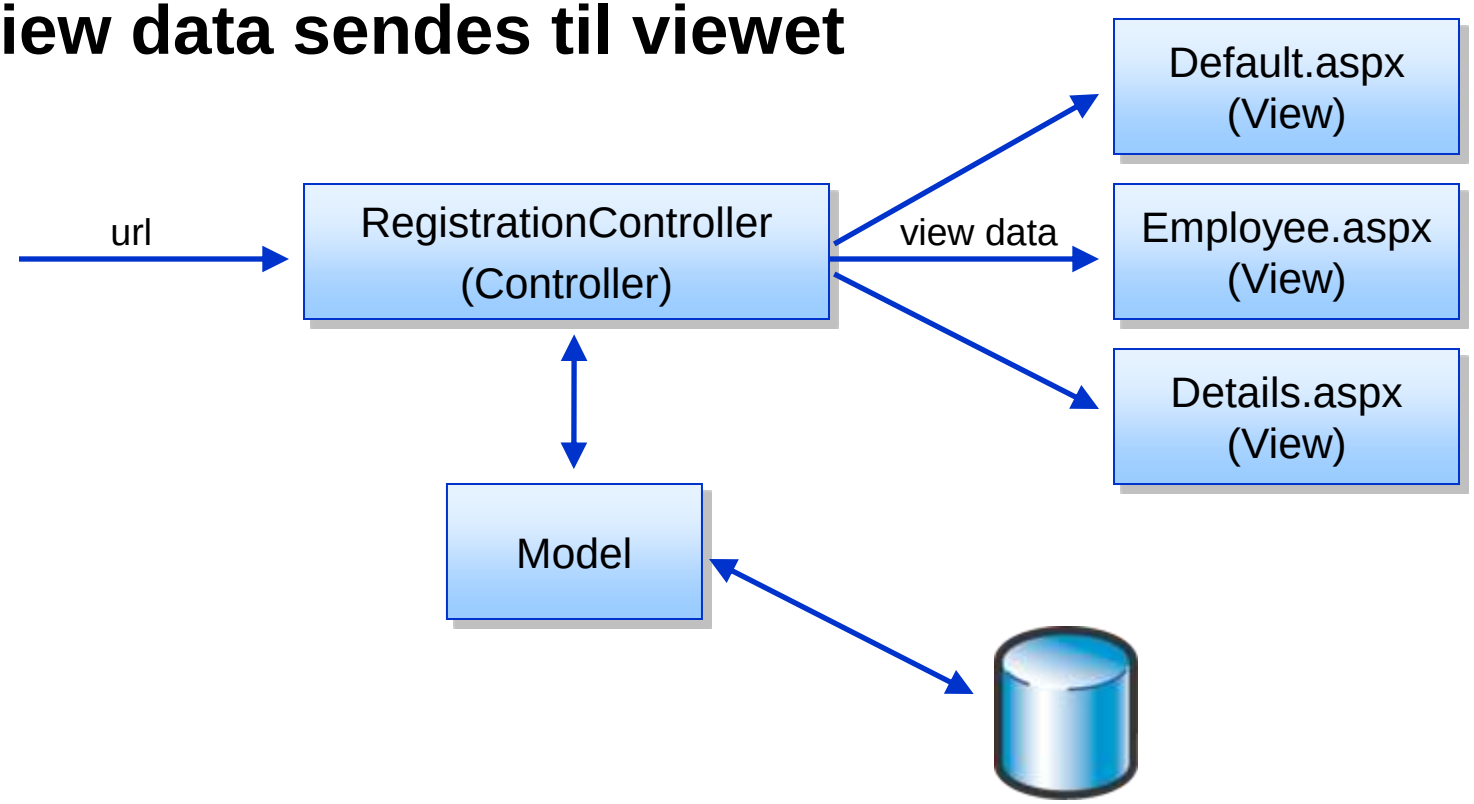
Anbefaling:
Placer model og modelhjælpeklasser i et selvstændigt projekt

- **Views**
  - Undermapper for hver controller
  - ”Shared” indeholder fælles views



Eksekveringsrækkefølge

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



RegistrationController

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

```
namespace MvcApplication.Controllers
{
  public class RegistrationsController : Controller
  {
    public System.Web.Mvc.ActionResult Employees()
    {
    }

    public System.Web.Mvc.ActionResult Employee(string initials)
    {
    }

    public System.Web.Mvc.ActionResult Details(System.Guid id)
    {
    }
  }
}
```

Controller navn = ”Registrations”

Action navn = ”Employees”

URL = /Registrations/Employees

URL = /Registrations/Employee/cja

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

RegistrationController - parametre

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

```
public System.Web.Mvc.ActionResult Details()
{
  System.Guid id = new System.Guid(Request["id"]);
}
```

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

```
public System.Web.Mvc.ActionResult Details(System.Guid id)
{
}
```

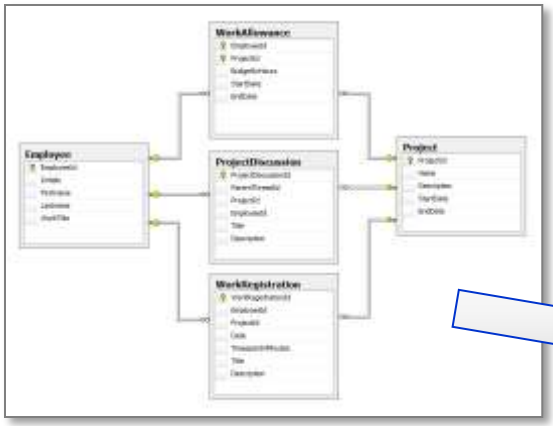
URL = /Registrations/Details?id=fdf2b7e4-0f9b-4c5b-ac37-27d5f3105ab3

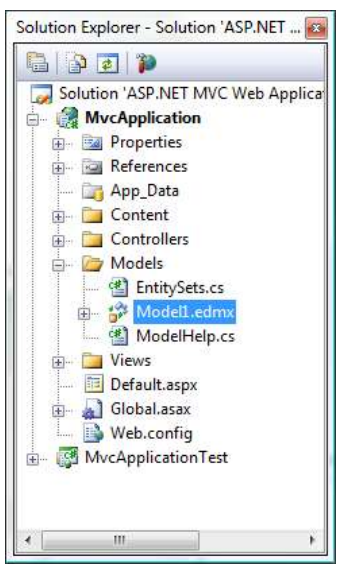
```
public System.Web.Mvc.ActionResult Details(System.Guid id)
{
}
```

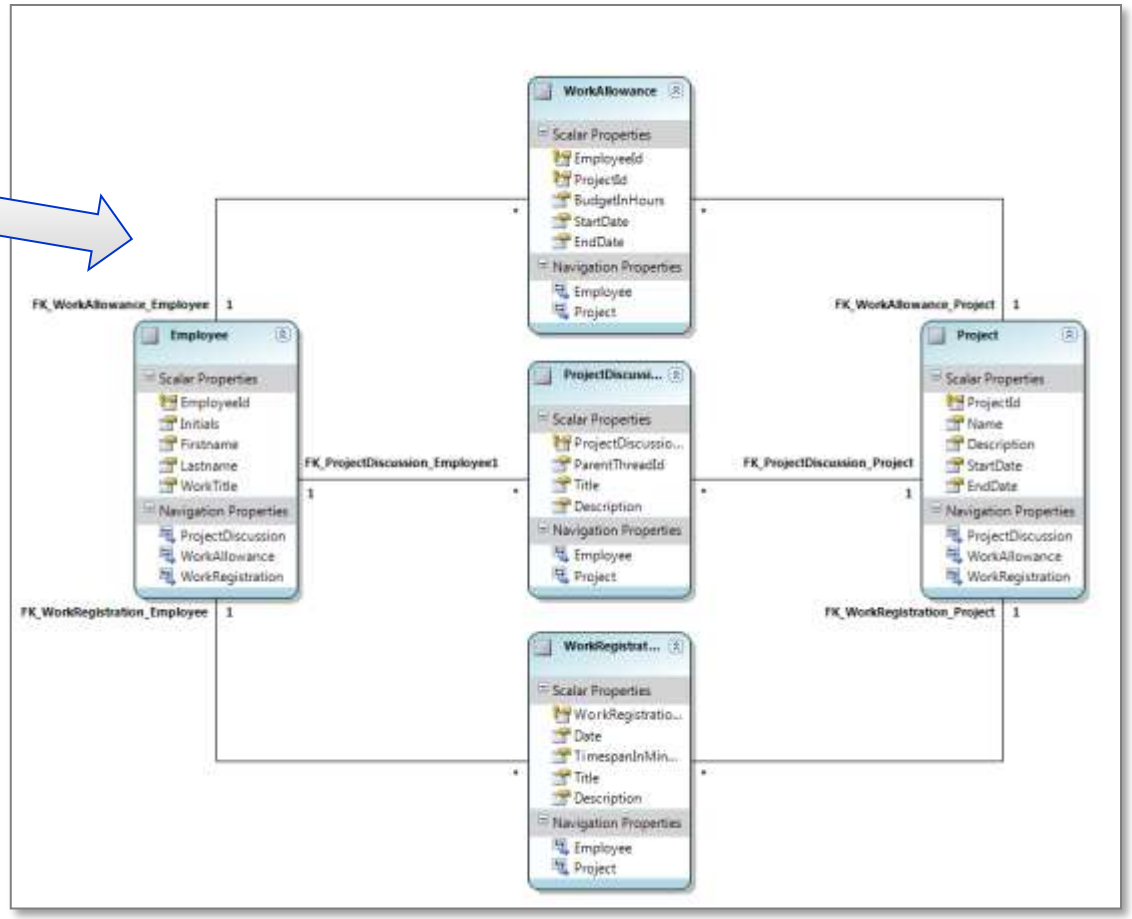
URL = /Registrations/Details/fdf2b7e4-0f9b-4c5b-ac37-27d5f3105ab3

Modellen

- **Entitetsmodel genereret ud fra databasen**







En komplet action metode

- **Nu kan vi i action metoder**
  - hente data fra controlleren
  - sende data videre til et view

```
namespace MvcApplication.Controllers
{
  public class RegistrationsController : Controller
  {
    public System.Web.Mvc.ActionResult Employees()
    {
      EmployeeList employees = null;

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

      return View("Default", employees);
    }
  }
}
```

Anbefaling:
Flyt tilgang til model
(inklusive LINQ-queries)
til klasser i ”Model”-namespacet

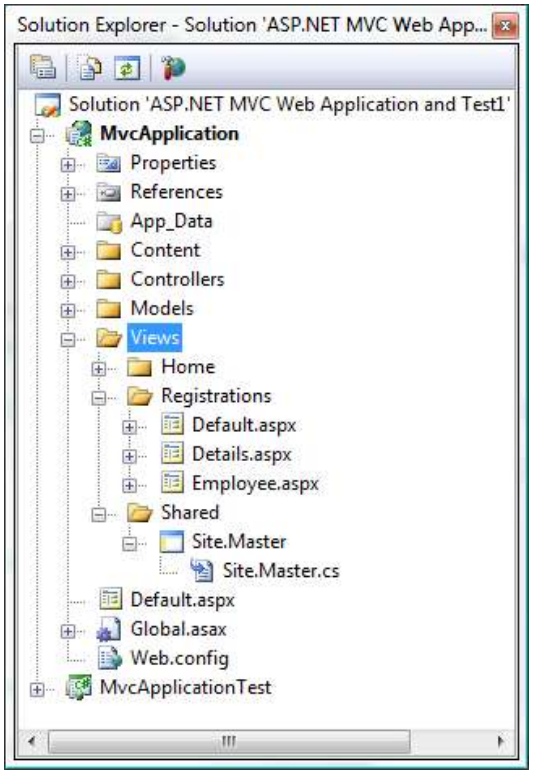
Viewet

- **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

Viewet

- **View placeres under ”Views” mappen**
  - Opret undermappe for hver controller (samme navn som controlleren)
  - Placer delte filer i ”Shared”



- **Når controller returnerer et View**
  - 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

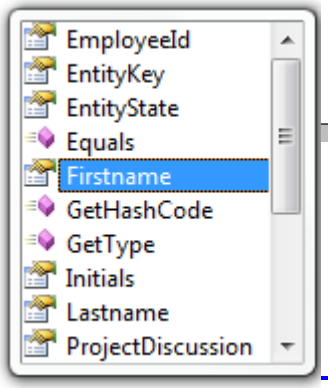
View

- **Viewet kaldes fra RegistrationsController**

```
public System.Web.Mvc.ActionResult Employees()
{
  ...
  return View("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()
  {
    Model[0].
  }
}
```



Rendering af viewet

- **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" ...
    Inherit="System.Web.Mvc.ViewPage<EmployeeList>" %>

<asp:Content ID="Content2" ContentPlaceHolderID="Main ...

  <ul>
<% foreach (Model1.Employee emp in Model)
   { %>
    <li>
      <%= Html.ActionLink(emp.Firstname + " " + emp.Lastname,
                          "Employee/" + emp.Initials)%>
    </li>
<%
   } %>
  </ul>
<asp:Content>
```

View output

- 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>
...
```

Rendering af view – via code behind

- **aspx med brug af Repeater**

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

Inherit="TODO System.Web.Mvc.ViewPage<EmployeeList>" %

<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>
```

Rendering af view – via code behind

- ... og den tilsvarende codebehind

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

  protected void repViewEmployees_ItemDataBound(
    object sender, System.Web.UI.WebControls.RepeaterItemEventArgs e)
  {
    if ((e.Item.ItemType == ListItemType.Item) ||
        (e.Item.ItemType == ListItemType.AlternatingItem) )
    {
      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,
                       "Employee/" + emp.Initials);
    }
  }
}
```

Html.ActionLink helper metode

- **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

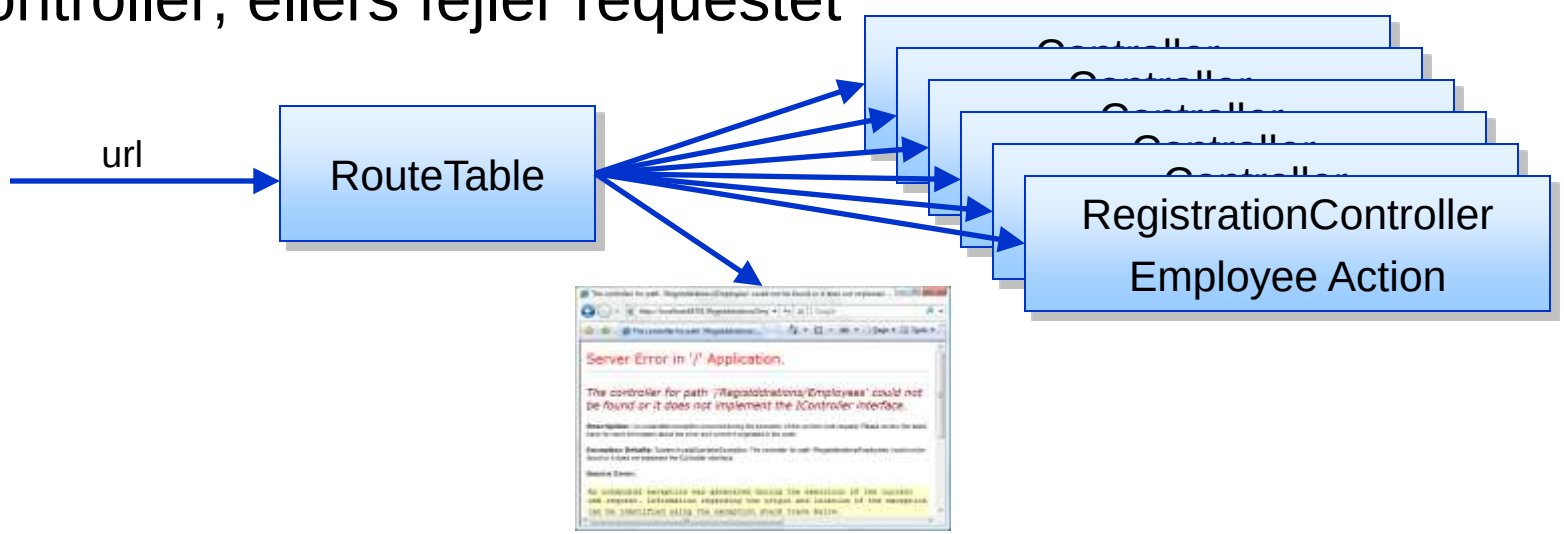
```
Html.ActionLink("linktekst", "actionName");
```

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

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

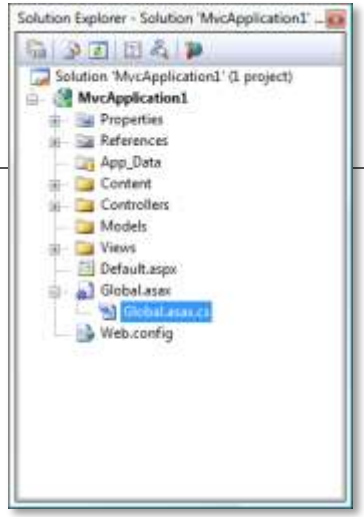
Anden MVC demo – URL routing

- **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



URL routing

- **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)
  {
     routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

     routes.MapRoute(
       "Employee",
       "Registrations/Employee/{initials}/{year}/{month}/{day}",
       new { controller = "Registrations", action = "Employee",
             year = (int?)null, month = (int?)null,
             day = (int?)null }
     );

     routes.MapRoute(
       "Default",
       "{controller}/{action}/{id}",
       new { controller = "Home", action = "Index", id = "" }
     );
}
```

URL routing – Url specifikation

- **En Url specifikation kan være**
  - en eksakt Url
  - en templated Url
  - eller en kombination

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

```
Url = "{controller}/{action}/{id}"
```

```
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

```
routes.MapRoute(
        "EksempelPåMinSide",
        "Min/Side",
        new { controller = "MinController", action = "Side" }
      );
```

URL routing – Url specifikation

- **Templated Url**

```
Url = "{controller}/{action}/{id}"
```

  - `"{tekst}"` angiver en parameter
  - Ovenstående vil blandt andet matche følgende

```
/                   - Hvis der findes en default controller, default action og
                      default ”id”

/Home/Default       - Hvis der findes en HomeController med en Default action og
                      en default værdi for “id”

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

URL routing – fra Route til Controller (1)

- **Med følgende Route og Url**

```
routes.MapRoute(
        "Default",
        "{controller}/{action}/{id}",
        new { controller = "Home", action = "Index", id = "" }
      );
```

```
 /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
```

URL routing – fra Route til Controller (2)

- **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

URL routing – mange parametre

- **Eksempel på brug af adskillige parametre**

```
routes.MapRoute(
       "Employee",
       "Registrations/Employee/{initials}/{year}/{month}/{day}",
       new { controller = "Registrations", action = "Employee",
             year = (int?)null, month = (int?)null,
             day = (int?)null }
     );
```

- **Giver mulighed for at følgende URLer 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**

```
public System.Web.Mvc.ActionResult Employee(string initials,
                                            int? year, int? month, int? day)
{
  ...
}
```

Url routing – validering af parametre

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

```
routes.MapRoute(
        "Employee",
        "Registrations/Employee/{initials}/{year}/{month}/{day}",
        new { controller = "Registrations", action = "Employee",
              year = (int?)null, month = (int?)null,
              day = (int?)null }
        new { initials = @"\p{L} {3,4}" }
      );
```

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

Url routing – udgående url'er

- **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", "Default"
                new { controller = "Home" } );
```

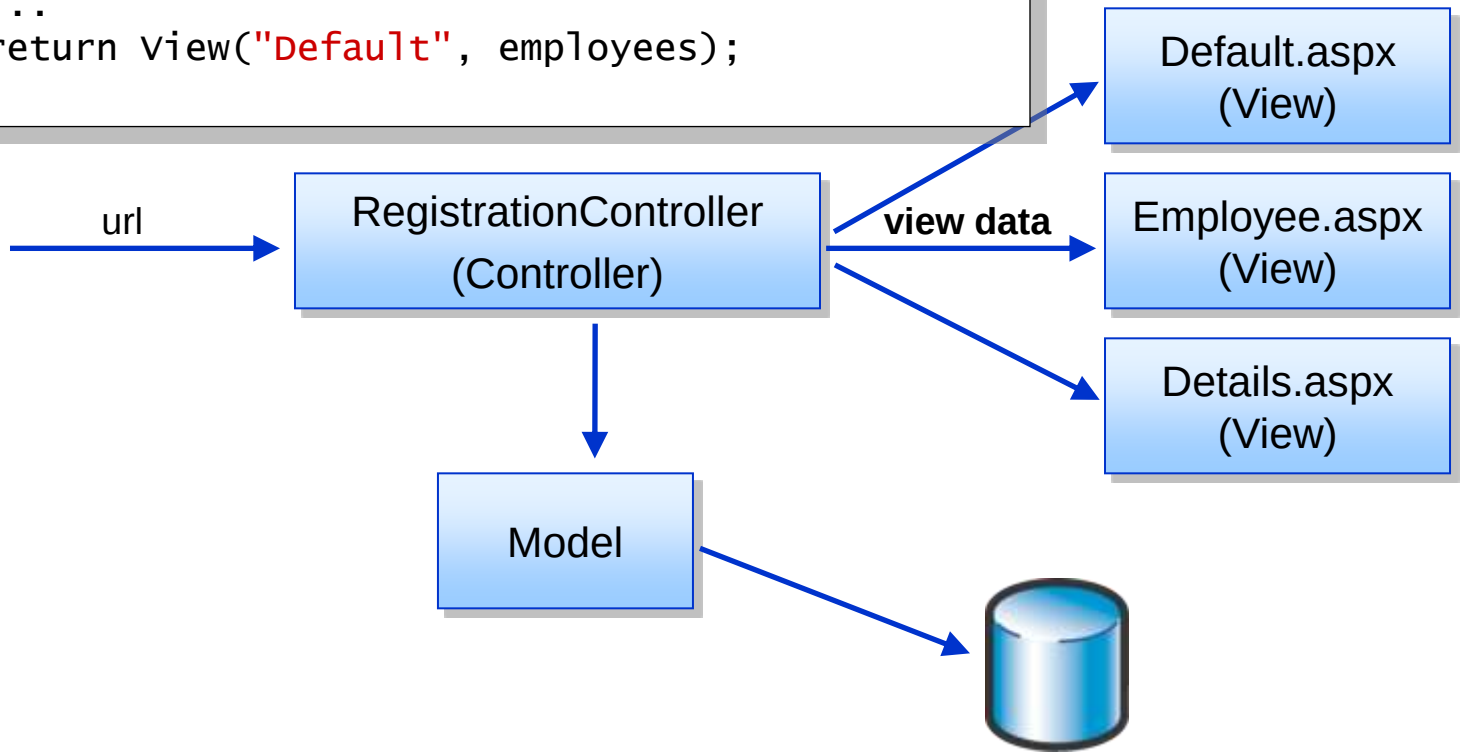
- **Benyt Controller.RedirectToAction til at redirigere til en anden side**
  - RedirectToAction udløser en Response.Redirect

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

Tredje MVC demo – fra controller til view

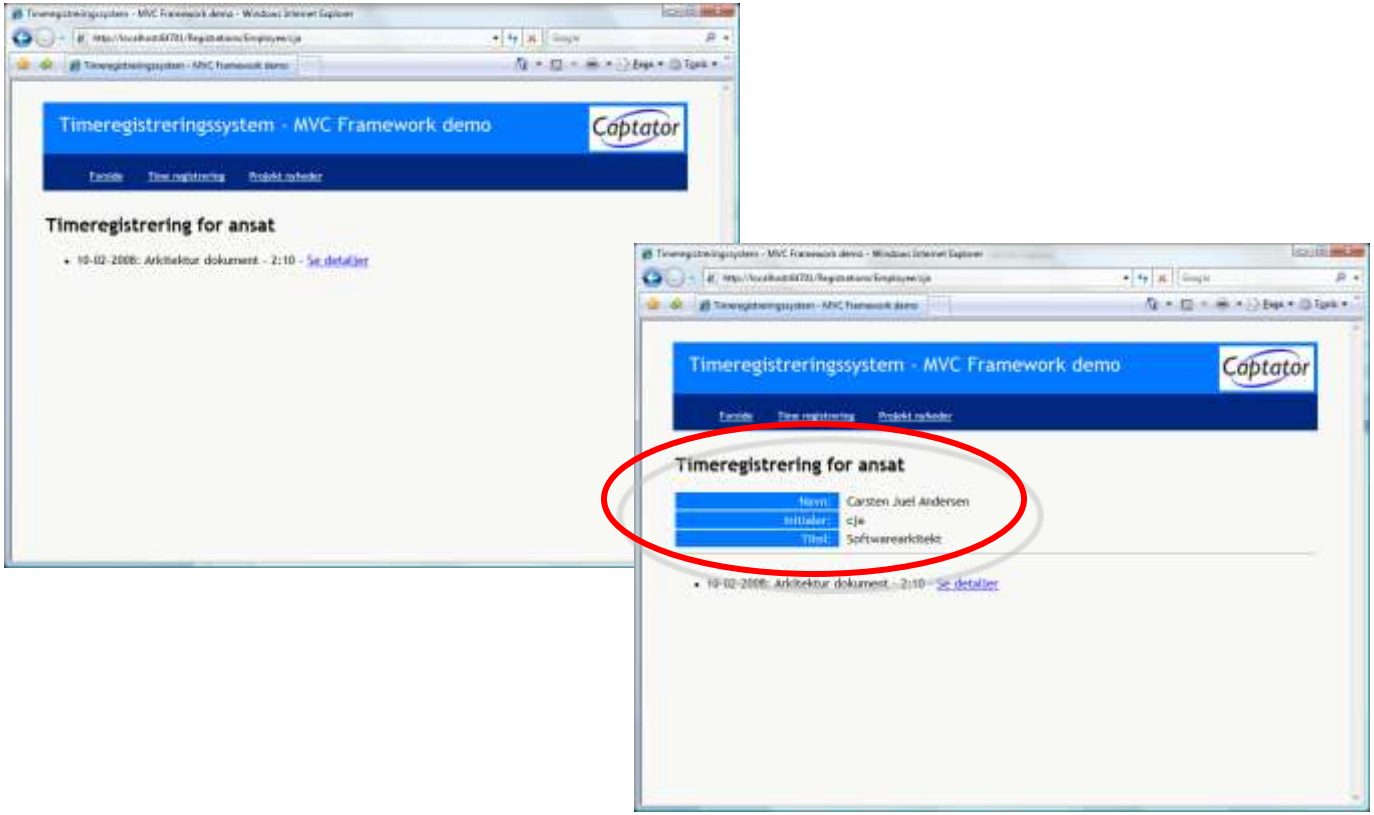
- Views må kun bero på de data Controller leverer
- ViewData overføres fra Controller til View

```
public System.Web.Mvc.ActionResult Employees()
{
  ...
  return View("Default", employees);
}
```



Overførsel af ViewData til viewet

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



Typesvag overførsel af ViewData

- I controlleren kan ViewData tilføjes typesvagt …

```
public System.Web.Mvc.ActionResult Employee(string initials,
                                            int? year, int? month, int? day)
{
  // Her hentes data fra modellen

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

  return View("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"]))
{ %> ...
```

Typestærk overførsel af ViewData (1)

- **Ved hjælp af et specifikt ViewData-objekt …**

```
public class EmployeeViewData
{
  public Model1.Employee Employee { get; set; }
  public Model1.WorkRegistrationList RegList { get; set; }
}
```

- **… kan ViewData tilføjes typestærkt i controlleren …**

```
public System.Web.Mvc.ActionResult Employee(string initials,
                                            int? year, int? month, int? day)
{
  // Her hentes data fra modellen

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

Typestærk overførsel af ViewData (2)

- og ved at tilføje ViewData-klassen som type-parameter på viewet

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

- … er der typestærk adgang til Model (ViewData):

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

Typestærk viewstate, property hedder **Model**

Brug af ViewUserControl (1)

- **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)
- **ViewUserControl 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 ViewUserControl typestærk og benyt ViewDataKey

Brug af ViewUserControl (2)

Captator

- **Gør ViewData på ViewUserControl typestærk …**

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

- **… således at der er typestærk tilgang til data:**

```
<%@ Control Language="C#" CodeBehind="EmployeeDetailsControl.ascx.cs" ...

Navn: <%= Model.Firstname %> <%= Model.Lastname%>
...
```

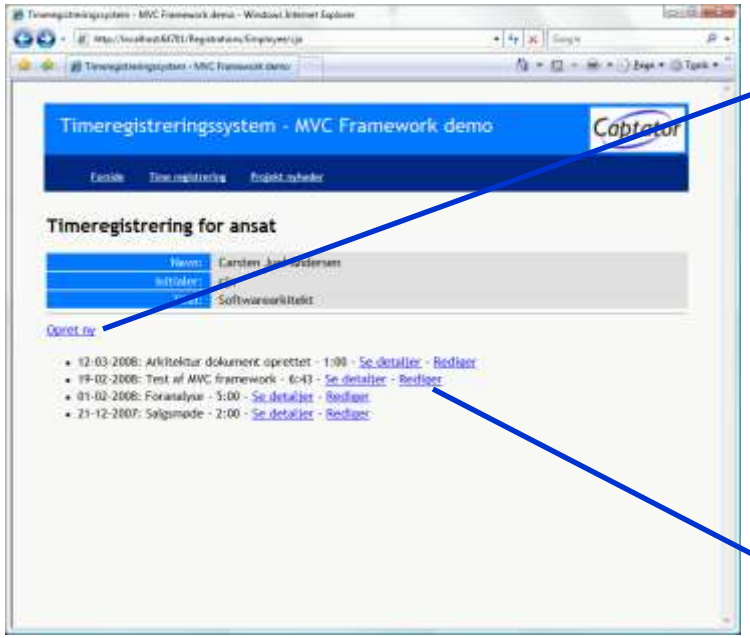
- **På viewet sikres, at Employee sendes til ViewUserControl ved at sætte ViewDataKey**

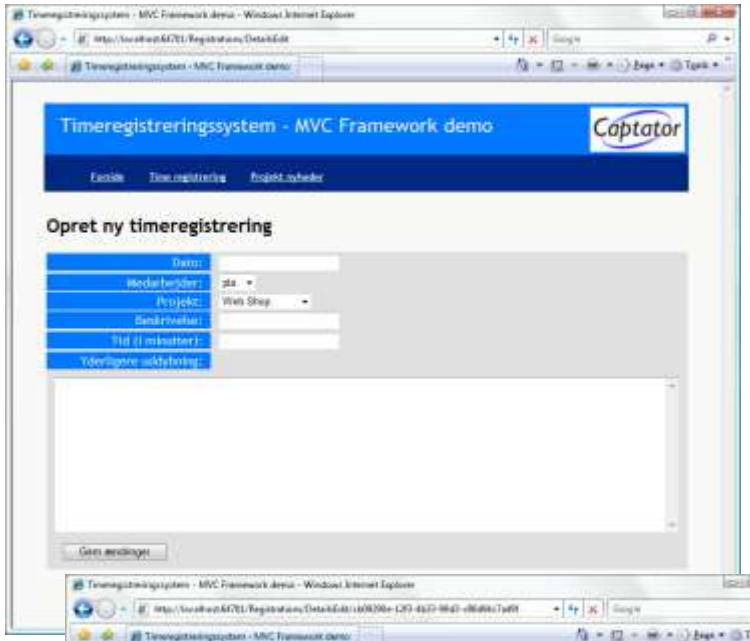
```
...
<uc1:EmployeeDetailsControl ID="EmployeeDetailsControl1"
                            ViewDataKey="Employee" runat="server" />
...
```

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

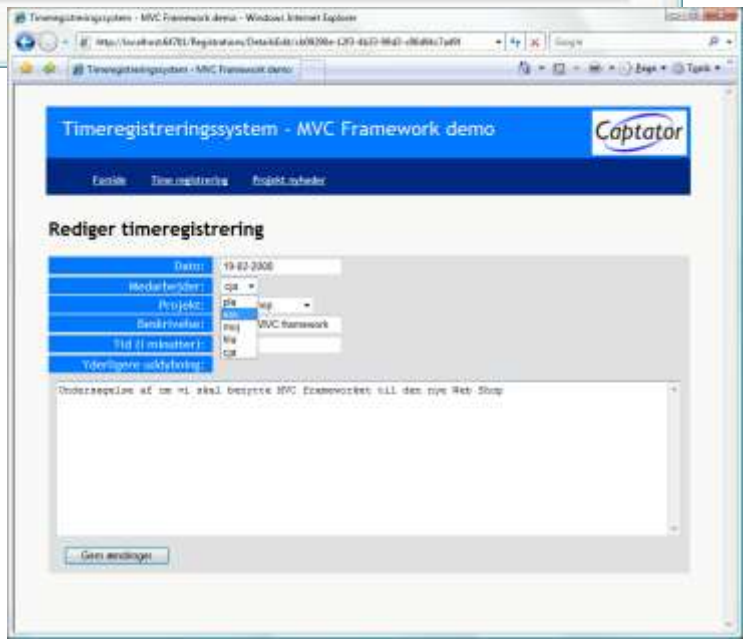
Fjerde MVC demo – forms og post data

- **Opret ny timeregistrering**



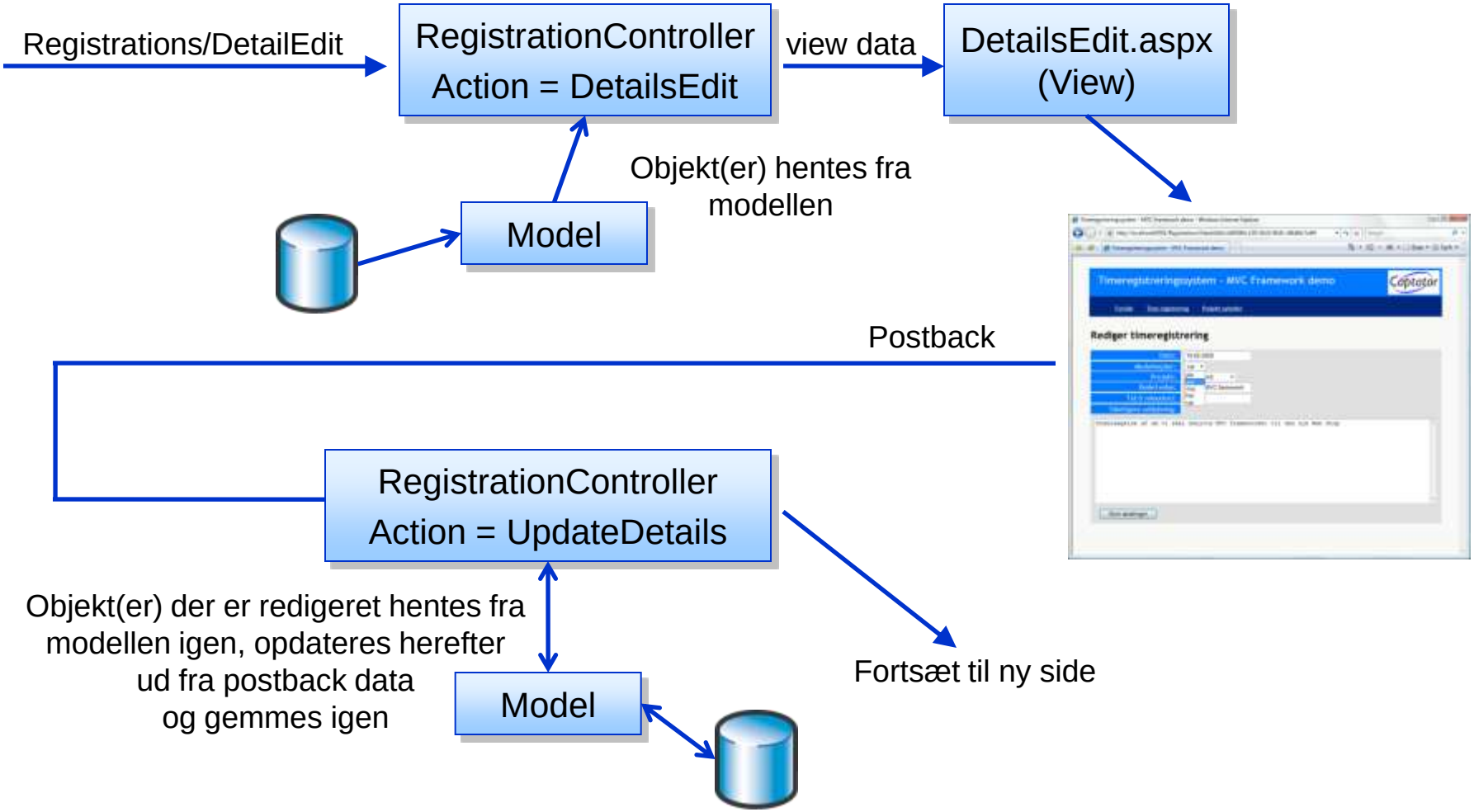


- **Redigering af eksisterende timeregistreringer**



Forms og post data - eksekveringsfølge

- **En action skaber data - en anden opdaterer data**



Redigering af eksisterende objekt (1)

- **Først oprettes links til opret og rediger controllere:**

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

- **DetailsEdit action**

```
public System.Web.Mvc.ActionResult DetailsEdit(System.Guid id)
{
  WorkRegistration wr = // Hentes fra modellen

  return View("DetailsEdit", wr);
}
```

Redigering af eksisterende objekt (2)

- **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/<%= Model.WorkRegistrationId %>"
method="post">

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

Redigering af eksisterende objekt (3)

- **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>

...
```

Redigering af eksisterende objekt (4)

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

```
public System.Web.Mvc.ActionResult 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

  return RedirectToAction("Employees");
}
```

Populering af dropdowns (1)

- 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

```
public System.Web.Mvc.ActionResult 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

  return View("DetailsEdit", wrEdit);
}
```

Populering af dropdowns (2)

- **Brug af HtmlHelper-metoden Select**

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

 Medarbejder: <%= Html.Select("EmployeeId", Model.Employees,
                 (object)Model.Wr.Employee.EmployeeId.ToString()))%><br />
  ...
  <input type="submit" value="Gem ændringer" />
</form>
...
```

- **I den anvendte overload tager Html.Select imod**
  - Id
  - Listen, der skal displayes
  - Det valgte elements key

Tilføj opret funktionalitet (1)

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

```
public System.Web.Mvc.ActionResult 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

  return View("DetailsEdit", wrEdit);
}
```

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

Tilføj opret funktionalitet (2)

- **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 = (Model.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) ?
            "" : Model.Wr.WorkRegistrationId.ToString() %>" method="post">
  Dato: <%= Html.TextBox("Date", (IsNew) ?
                          "" : Model.Wr.Date.ToShortDateString())%><br />
...
```

Tilføj opret funktionalitet (3)

- **Update action tilrettes, så den kan håndtere opret**

```
public System.Web.Mvc.ActionResult 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

  return RedirectToAction("Employees");
}
```

Simplificer update action

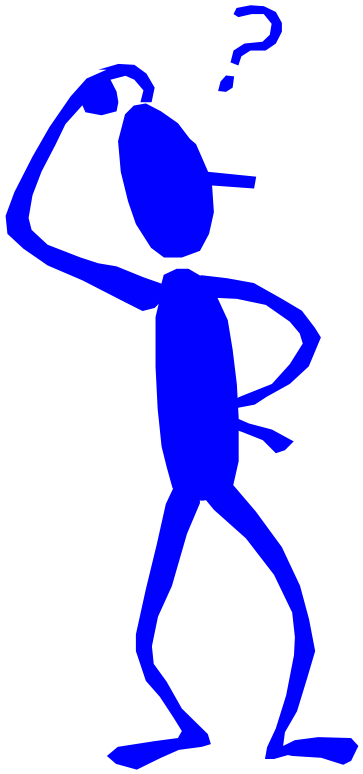
- **Anvend UpdateModel**
  - Mapper form-parametre direkte ind på model objekt

```
public System.Web.Mvc.ActionResult 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();
  }

  UpdateModel<Models.WorkRegistration>(wr);

  // Gem objekt i modellen og foretag en redirect

  ...
}
```

ASP.NET MVC Framework

- **MVC Framework er et alternativ til web forms**
  - Ikke nødvendigvis en afløser!
- **MVC har klare styrker især når det drejer sig om visning af data**
  - JavaScript, CSS m.v. bliver lettere tilgængelig, når sideopbygning er simplere (simplere ID'er m.v.)
  - Klar opdeling mellem forretningslogik og UI-del
- **Der findes stadig mere "udtryksfulde" kontroller til Web Forms frem for MVC**
  - Men mon ikke det kommer ...

Spørgsmål



www.captator.dk

nyheder, artikler, information, ...