Server side Web Development

Lecture 2

Introduction ASP.NET and MVC

Outline

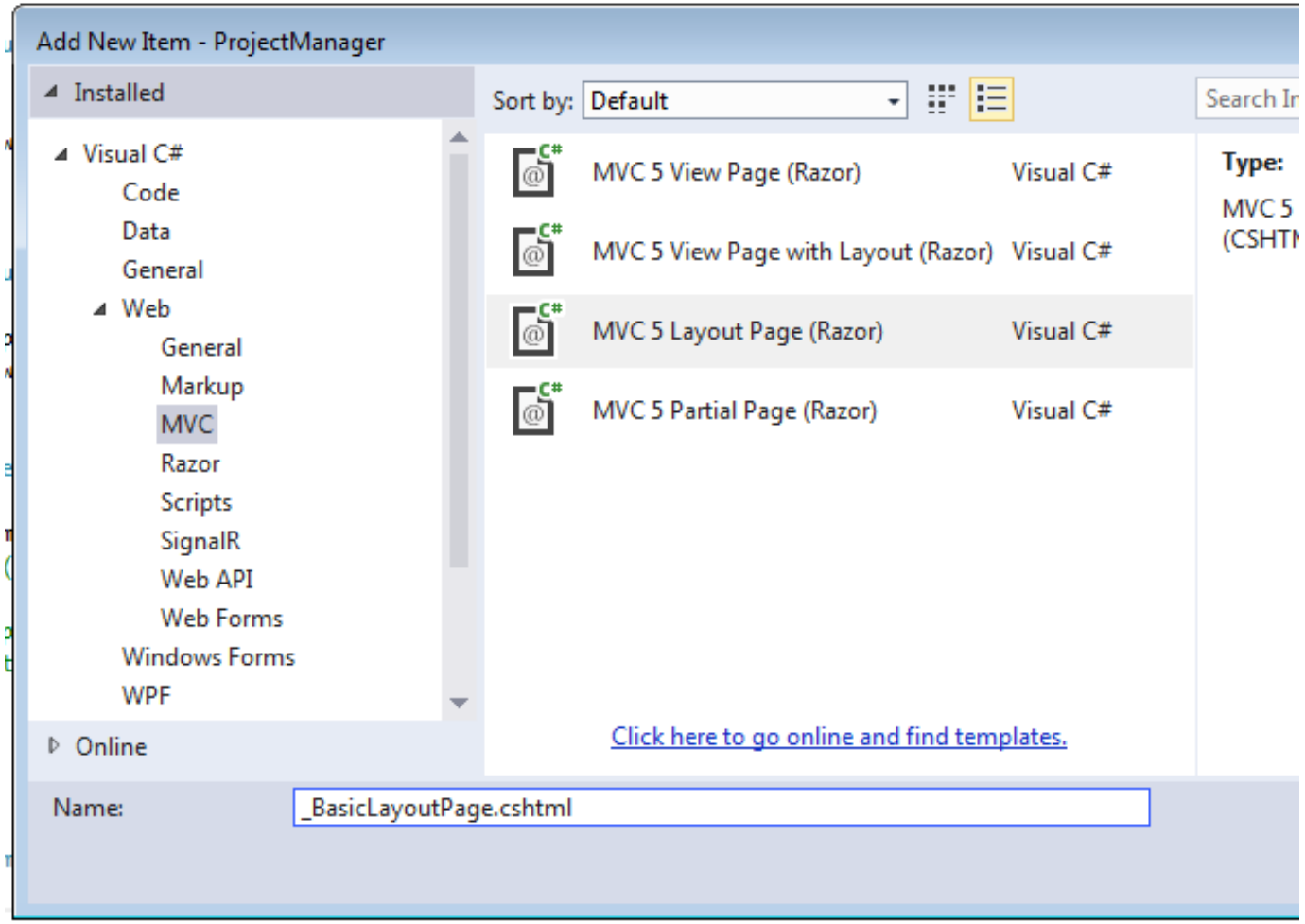
- Layouts
- Example (Exercise1) in the ProjectManager application
- Built in helper methods
  - Partial
  - Action
- Redirecting to other action method
- Miscellaneous
  - Model binding
  - ActionResult return value
  - Lambda expressions
  - Styling with CSS
  - Configuring the default route at application start

Razor code - Layout

- A layout is a template that can be used in several views
- Creates a common look and feel
- Is created in a seperate View
- A View can set it in its razor code block
  Layout = <The specific layout view file>
- The statement in the razor code block
  Layout = null;
  Means that the view is not using any layout (self contained)
- The layout file name has to begin with an _ (underscore character)

Creating a Layout page

- Right-click Views folder select Add ->New item Then…



Name the layout page, starting with an _ character.

Creating a Layout page

- The content looks like this…

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
</head>
<body>
    <div>
        Common code goes here…
        @RenderBody()
        …and here
    </div>
</body>
</html>
```

Using the ViewBag and the Title property you may set the title ☺

The HTML from the file using the layout page is inserted in the position of @RenderBody

Content common for all pages using the layout can now be entered in the div-tag

Using the Layout page

- The view using the layout page may look like this…

```
@model ProjectManager.Models.Department

@{
    ViewBag.Title = "Add department";
    Layout = "~/Views/_BasicLayout.cshtml";
 }

<h2>Add a department</h2>
@using(Html.BeginForm())
{
    @:Department id: @Html.TextBox("depid",Model.depid)
    @:Name: @Html.TextBox("name",Model.name)
    @:Bossid: @Html.TextBox("bossid",Model.bossid)
    <input type="submit" value="Add" />
}
```

Set the Title property
and
Specify which layout page to use

Only the specific code for the page is entered in the view

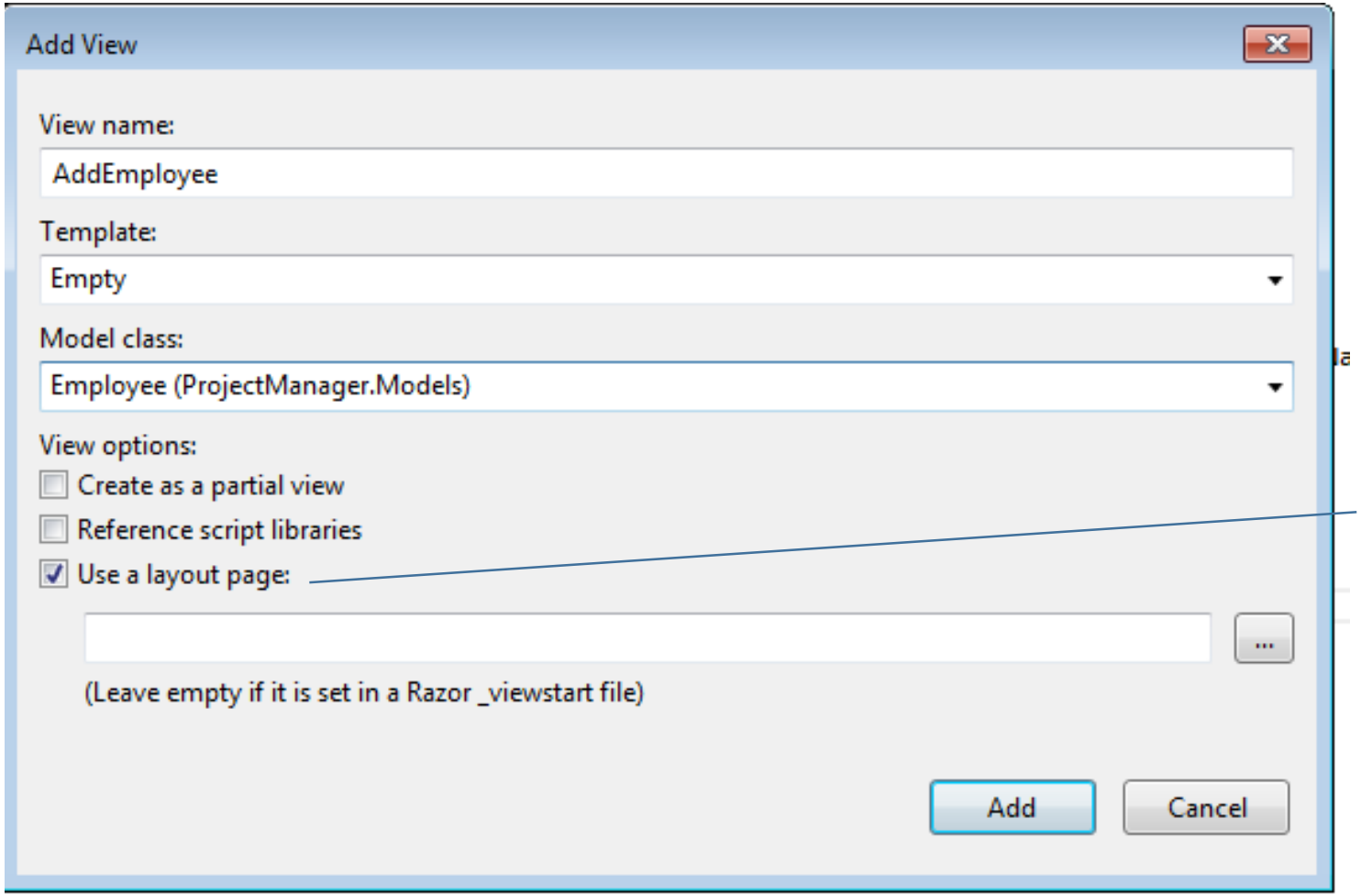
Adding a ViewStart - file

- The ViewStart – file is a layout only containing the line specifying which layout to use
  - I.e.

```
@{
        Layout = "~/Views/Layoutname.cshtml"
}
```

- If a view does not contain a line which specify the layout, the view-engine will look in the ViewStart file for this line
- The purpose is to avoid changing the layout specification on each view if a layout page with a different name is going to be used

Creating views with Layout page



New views can be set to use an existing layout page

Example – Project Manager application

- Exercise 1 has the following steps
- Create a class for Employee in Models folder
- In the view “TheListOfDepartments” add ActionLinks that show the department name and calls action method “ShowDepartmentAndBoss”
  - Pass the department id (depid) and the employee id for the boss (bossid)
- Create a class (DepartmentBoss.cs) for the combined data in Models folder
- Create action method “ShowDepartmentAndBoss”
  - Takes two parameters, depid and bossid
  - Find the department object and the employee object
  - Create a DepartmentBoss object with values from the Department and Employee objects
  - Pass the DepartmentBoss object to the View call
- Create a view “ShowDepartmentAndBoss”
  - Strongly typed with DepartmentBoss
  - Let the view display the data from the Model object

There will be a somewhat different solution later using Partial Views and Action calls

Lambda expressions

- A powerful way of writing anonymous functions in a condensed way
  - E.g.
    theSum = (x,y,z) => (x+y+z);
  - Is a function taking three values, stored in the parameters x,y and z, and returning the sum
  - The lambda operator => is interpreted as “goes to”
- Can be used in C# to shortening code writing e.g. when working with Lists

Lambda expression - example

The code

```
Employee empObj = null;
foreach (Employee emp in employeeList)
{
    if (emp.empid == Convert.ToInt32(bossid))
        empObj = emp;
}
```

Can be shortened to

```
Employee empObj = employeeList.Find(x => x.empid == Convert.ToInt32(bossid));
```

More HTML helper functions

ASP.NET MVC comes with some predefined extensions, that makes things a bit easier for us as developers:

- Partial() - Renders a specified partial view
- Action() - Invokes a child action in any Controller

Partial views

- A partial view is a way of reusing HTML and Razor syntax, recurring in several views, by placing this code in a separate partial view
- Will only contain fragments of HTML code and razor syntax
- Does not use layouts
- Create a partial view by adding a view the ordinary way, but check the “create as partial” view option
- Should be placed in the Views/Shared folder
- A partial view that is not strongly typed will by default use the calling views Model

HTML extensions – Partial

@Html.Partial("_ViewName", [model], [ViewDataDictionary])

Renders a partial view. The partial view can access the same model (ViewBag) as it´s parent.

By using the model or/and ViewDataDictionary we can pass such variables created on the fly in the calling view

Partial views - example

Assune there is a listing which occur in a similar way in several views. The code for doing the list may be placed in a partial view

```
<ul>
    @foreach (var obj in Model)
    {
        <li>
            @obj.name
        </li>
    }
</ul>
```

Two views calling the partial view:

```
@model List<ProjectManager.Models.Department>
@{
    ViewBag.Title = "List Departments";
}
<h2>List Departments</h2>

@Html.Partial("PartialListing")
```

```
@model List<ProjectManager.Models.Employee>
@{
    ViewBag.Title = "List Employee";
}
<h2>List Employee</h2>

@Html.Partial("PartialListing")
```

Child Action methods

- Child actions are action-methods called from a within a view
- Controller logic that would be repeated in several views can be placed in a child action method
- Any action method can be used as a child action method
- Usually a child action method would render a PartialView

HTML extensions – Action

@Html.Action("ActionName", "Controller", RouteValues)

Invokes and renders an action within the containing View.

Example:

@Html.Action("_MinDetails", "Book", new {id=2})

Will render the View _MinDetails in controller BookController and send the param id the value 2

Child Action - example

Three action methods, the first coordinating the final view and two child action method called from the final coordinating view.
The two child action methods, shown on next slide, are for finding the department and employee.

```
public ActionResult ShowDepartmentAndBossSecond(string bossid, string depid)
{
   ViewBag.Departmentid = depid;
   ViewBag.Employeeid = bossid;
   return View();
}
```

Child Action – example cont.

```
[ChildActionOnly]
public ActionResult ShowDepartmentPartial(string depid)
{
    Department depObj = departmentList.Find(x => x.depid == Convert.ToInt32(depid));
    if (depObj != null)
       return PartialView(depObj);
    return null;
}
[ChildActionOnly]
        public ActionResult ShowEmployeePartial(string empid)
        {
            Employee empObj = employeeList.Find(x => x.empid == Convert.ToInt32(empid));
            if (empObj != null)
                return PartialView(empObj);
            return null;
        }
```

Child Action – example cont.

The final view for coordinating the result by calling the two child action methods

```
@{
    ViewBag.Title = "ShowDepartmentAndBossSecond";
}

<h2>ShowDepartmentAndBossSecond</h2>
<h3>
    @Html.Action("ShowDepartmentPartial", new { depid = ViewBag.Departmentid})
    @Html.Action("ShowEmployeePartial", new { empid = ViewBag.Employeeid})
</h3>
```

HTML extensions

Partial vs Action

Use partial when:

- Rendering static content in the partial view
- The partial view can use data in the parent views model

Use Action when you need data that is not accessible in the partial views model

Redirection of action method

- POST-requests are used to change the state of an application
  - Issued from a view with a form input, there is a risk that the user will reload the view, thereby resubmitting the form
  - To avoid this problem use the Post-Redirect-Get pattern
  - In the action-method, receiving the posted data, redirect to a GET action-method
- Use *RedirectToAction("action")* in the action-method to redirect internally to another action-method, after the post
- In order to preserve the posted data store it in the TempData variable
  - Similar to how Session variable works, but preserves data only between two consecutive requests

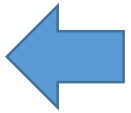
Redirection of action method - example

```
public ViewResult AddDepartment()
{
    Department depObj = new Department();
    return View(depObj);
}

[HttpPost]
public RedirectToRouteResult AddDepartment(Department depObj)
{
    TempData["DepartmentObject"] = depObj;
    return RedirectToAction("ShowDepartment");
}

public ActionResult ShowDepartment()
{
    Department depObj = (Department) TempData["DepartmentObject"];
    return View(depObj);
}
```

Calls the view AddDepartment
(See next slide)



The posted result from the view

Redirected to a GET action-method

Redirection of action method - example

```
@model ProjectManager.Models.Department

@{
    ViewBag.Title = "Add department";
 }

<h2>Add a department</h2>
@using(Html.BeginForm())
{
    @:Department id: @Html.TextBox("depid",Model.depid)
    @:Name: @Html.TextBox("name",Model.name)
    @:Bossid: @Html.TextBox("bossid",Model.bossid)
    <input type="submit" value="Add" />
}
```

Model Binding

- Is the process of creating .NET objects using the data sent by the browser in a request
- Data can be delivered from several sources
  - Request.Form
  - RouteData.Values
  - Request.QueryString
  - Request.Files
- Uses name matching between the source and parameter name (in action method)
  - When a request is made the model binder searches the sources and stops when a matching name is found
- Reading see Freeman: chapter 24 and chapter 9 pp. 227 – 230

ActionResult return value

- Is a class which provide the default return value from an action-method
- Has several subclasses such as e.g. ViewResult, which is what a view will return
- What should we use as return type for an action-method?
  - If there is only one possible return value use the specific subclass
  - If the method has several possible return values of different types use the main class ActionResult
    - In some cases we might end up returning a null-value

Styling with CSS

- Add a folder Content to the project (if it does not exist already)
  - This is where css-files should be placed
- For views you want to style add link-tag to the css-file in the header (as usual)
- In most applications you will probably use a layout page
  - Then the link-tags need to be placed in the layout-page header

- Reading see Freeman: chapter 7 pp. 190 - 192

Configuring the default route at application start

- In the file Global.asax directly in the project folder
  - Find the method Application_Start which is run when application starts
  - Calls the method RegisterRoutes
  - E.g.

In Global.Asax

```
public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
```

Configuring the default route at application start

- In the file RouteConfig.cs in the App_Start folder
  - Find the method RegisterRoutes
  - Here the default setting for the URL the application will use is stored
  - E.g.

In App_Start/RouteConfig.cs

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
       name: "Default",
       url: "{controller}/{action}/{id}",
       defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
    );
}
```

Reading

- Pro ASP.NET MVC 5
  - Chapter 3 – The MVC pattern
  - Chapter 5 – Working with Razor
  - Chapter 20 – Views (espec. About Partial Views and Action)
- Links:
  - http://stackoverflow.com/questions/4743741/difference-between-viewresult-and-actionresult
  -