

CRUD, *Layout Pages* och *Partial Views*

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET MVC vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Look, förutom Linnéuniversitetets logotyp och symbol, är licensierad under:



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.

<http://creativecommons.org/licenses/by-nc-sa/2.5/se/>

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – ASP.NET MVC" och en länk till <https://coursepress.lnu.se/kurs/aspnet-mvc> och till Creative Common-licensen här ovan.

Vad krävs för att...

- ✓ ...ge en applikation CRUD-funktionalitet? Det vill säga vad måste applikationen erbjuda för att användaren ska kunna skapa, läsa, uppdatera och ta bort data?

Nästa födelsedag!

- Ellen Nu kommer att fylla 4 år på en onsdag om 45 dagar.
- Nisse Hult kommer att fylla 47 år på en tisdag om 86 dagar.
- Carl Gustaf kommer att fylla 68 år om 123 dagar.

[Lägg till födelsedag](#)

Skapa födelsedata

Namn:

Robert De Niro

Födelsedatum:

1943-08-17

Skicka

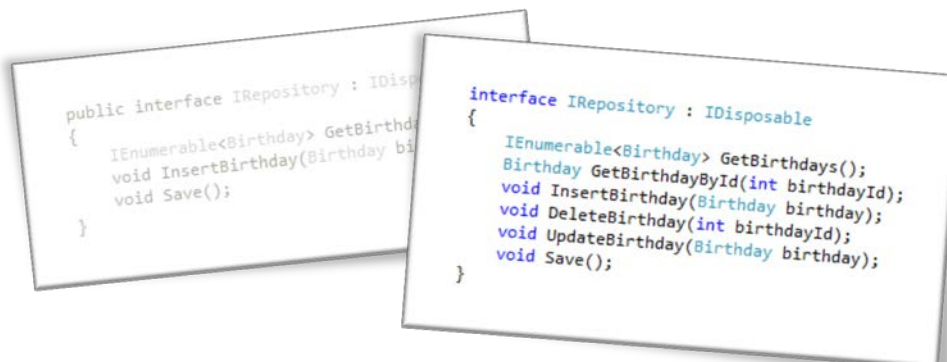
I en CRUD-applikation ska...

- ✓ ...det finnas URL:er för att presentera, skapa, redigera och ta bort data:

URL	Verb	Syfte
/Birthday/	GET	Visar en lista med födelsedata.
/Birthday/Create/	GET	Visar ett tomt HTML-formulär där användaren kan fylla i födelsedata.
	POST	Skapar födelsedata som sparas i databasen.
/Birthday/Edit/id	GET	Visar ett HTML-formulär med födelsedata.
	POST	Sparar ändringarna av födelsedata i databasen.
/Birthday/Delete/id	GET	Visar ett HTML-formulär där användaren måste bekräfta borttagning av specificerat födelsedata.
	POST	Tar bort specificerat födelsedata från databasen.

- ✓ ...centrallagret (*repository*) ha funktionalitet för att:

- Skapa data (*Create*).
- Läsa data (*Read*).
- Uppdatera data (*Update*).
- Ta bort data (*Delete*).



```

public interface IRepository : IDisposable
{
    IEnumerable<Birthday> GetBirthdays();
    void InsertBirthday(Birthday birthday);
    void Save();
}

interface IRepository : IDisposable
{
    IEnumerable<Birthday> GetBirthdays();
    Birthday GetBirthdayById(int birthdayId);
    void InsertBirthday(Birthday birthday);
    void DeleteBirthday(int birthdayId);
    void UpdateBirthday(Birthday birthday);
    void Save();
}

```

View Page with Layout (Razor)

- ✓ I en *View Page with Layout (Razor)* placeras kod som är gemensam för vyer och tjänar på så sätt som en mall för vyerna.
- ✓ En, eller flera, platshållare i en *layout* identifierar de delar av mallen som ska kunna förändras av en vy.
- ✓ Filen som innehåller mallen ska placeras i katalogen `\Views\Shared`.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />
  <title>@ViewBag.Title</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  @RenderBody()

  @Scripts.Render("~/bundles/jquery")
  @RenderSection("scripts", required: false)
</body>
</html>
```

Vyer som använder en *Layout Page*

- ✓ Då en vy skapas väljs vilken *Layout Page* som vyn ska använda.
- ✓ I vyn placeras HTML och kod som är specifik för vyn.

The collage illustrates the concept of layout pages in ASP.NET MVC. It features three snippets of Razor code, a diagram of a layout structure, and a screenshot of the 'Add View' dialog.

Code Snippet 1 (Left): Shows the beginning of a view, including the DOCTYPE, charset, viewport, and rendering of the ViewBag title and scripts.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<title>@ViewBag.Title</title>
<styles.Render("~/Content/css")>
<Scripts.Render("~/bundles/modernizr")>
</head>
<body>
    @RenderBody()
    @Scripts.Render("~/bundles/jqueryval")
    @RenderSection("scripts", required: false)
</body>
</html>
```

Code Snippet 2 (Middle): Shows a view model and the rendering of a list of birthdays, including a link to add a new birthday.

```
@model IEnumerable<NextBirthday.Models.Birthday>
@{
    ViewBag.Title = "Nästa födelsedag";
}
<h1>
    Nästa födelsedag!
</h1>
<div>
    @if (Model != null)
    {
        <ul>
            @foreach (var item in Model)
            {
                <li><strong>@Html.DisplayFor(item, "Name")</strong>
                <div>
                    @Html.DisplayFor(item, "Birthdate")
                </div>
            }
        </ul>
    }
</div>
<p>
    @Html.ActionLink("Lägg till födelsedag", "Add", "Birthdays")
</p>
```

Code Snippet 3 (Right): Shows a view model and the rendering of a form for adding a new birthday, including validation and a submit button.

```
@model NextBirthday.Models.Birthday
@{
    ViewBag.Title = "Lägg till födelsedag";
}
<h1>Lägg till födelsedag</h1>
<div>
    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true, "Fel inträffade. Korrigera det som är fel och försök igen.")
        <div class="editor-label">
            @Html.LabelFor(model => model.Name)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Name)
            @Html.ValidationMessageFor(model => model.Name)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.Birthdate)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Birthdate)
            @Html.ValidationMessageFor(model => model.Birthdate)
        </div>
        <p>
            <input type="submit" value="Spara" />
        </p>
    }
</div>
<p>
    @Html.ActionLink("Tillbaka till lista med födelsedagar", "Index")
</p>
<section Scripts>
    @Scripts.Render("~/bundles/jqueryval")
</section>
```

Diagram (Top Center): A diagram showing a layout structure. It consists of a central content area labeled 'Content' and a surrounding area labeled 'Layout'. The 'Layout' area is further divided into 'Header' and 'Footer' sections. The 'Content' area is labeled 'Content' and the 'Layout' area is labeled 'Layout'.

Dialog Box (Right): A screenshot of the 'Add View' dialog box. The 'View name' field is empty. The 'View engine' is set to 'Razor (CSHTML)'. The 'Create a strongly-typed view' checkbox is checked. The 'Model class' is 'Birthday (NextBirthday.Models)'. The 'Scaffold template' is 'Create'. The 'Create as a partial view' checkbox is checked. The 'Use a layout or master page' checkbox is checked and highlighted with a red circle. The 'ContentPlaceHolder ID' is 'MainContent'.

Modifiering av interface och klass för "repository".

- ✓ Interfacet IRepository kompletteras med definitioner för metoder till för att hämta, ta bort och uppdatera födelsedata.
- ✓ Klassen EFRepository måste implementera metoderna GetBirthdayById, DeleteBirthday och UpdateBirthday.
 - Metoden GetBirthdayById returnerar ett Birthday-objekt med specificerat id. Hittas ingen post i databasen returneras null.
 - Metoden DeleteBirthday anropar Remove, som markerar att Birthday-objektet ska tas bort. Först efter att SaveChanges anropas tas posten bort i databasen.
 - Metoden UpdateBirthday sätter status för entitiesobjektet vilket markerar att Birthday-objektet ska uppdateras. Först efter att SaveChanges anropas uppdateras posten i databasen.

```
interface IRepository : IDisposable
{
    IEnumerable<Birthday> GetBirthdays();
    Birthday GetBirthdayById(int birthdayId);
    void InsertBirthday(Birthday birthday);
    void DeleteBirthday(int birthdayId);
    void UpdateBirthday(Birthday birthday);
}
```

```
public class EFRepository : IRepository
{
    private GeekBirthdayEntities _entities = new GeekBirthdayEntities();

    #region IRepository Members

    public IEnumerable<Birthday> GetBirthdays()
    {
        return _entities.Birthdays.ToList();
    }

    public Birthday GetBirthdayById(int birthdayId)
    {
        return _entities.Birthdays.Find(birthdayId);
    }

    public void InsertBirthday(Birthday birthday)
    {
        _entities.Birthdays.Add(birthday);
    }

    public void DeleteBirthday(int birthdayId)
    {
        var entity = _entities.Birthdays.Find(birthdayId);
        _entities.Birthdays.Remove(entity);
    }

    public void UpdateBirthday(Birthday birthday)
    {
        _entities.Entry(birthday).State = EntityState.Modified;
    }

    public void Save()
    {
        _entities.SaveChanges();
    }
}
```


Redigera födelsedata – från *action method* till formulär

- ✓ Födelsedatat som ska redigeras specificeras med hjälp av primärnyckelns värde, som skickas till controllermetoden så att rätt post med födelsedata kan hämtas från databasen.
- ✓ Födelsedatat, paketerat i ett Birthday-objekt, utgör modellen som skickas till vyn som visar födelsedatat i ett formulär. Hittas inte någon post med specificerat id visas vyn NotFound.

Ett alternativ till View("NotFound") är HttpNotFound(), men då måste vi ta hand om en 404...

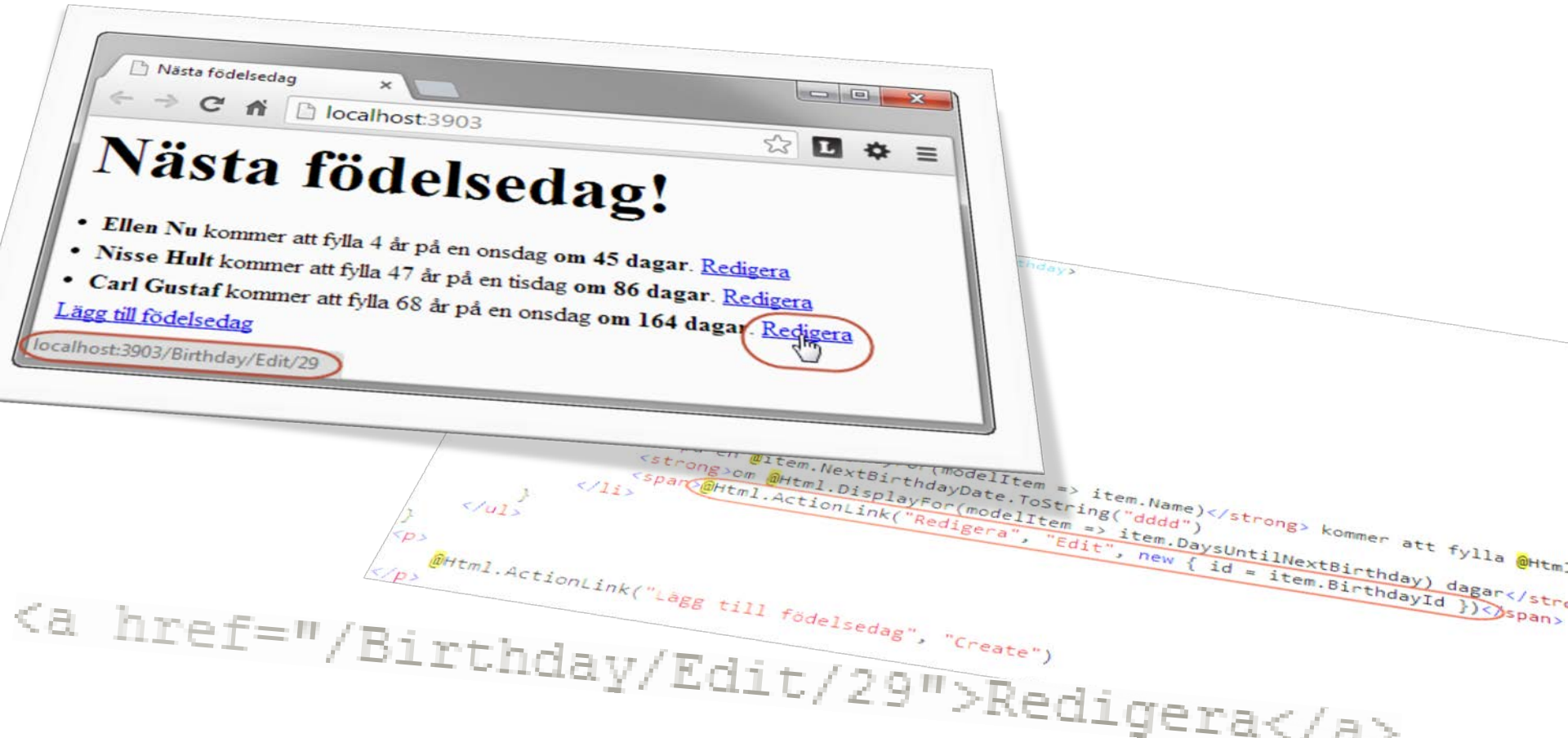
```
public ActionResult Edit(int id = 0)
{
    var birthday = _repository.GetBirthdayById(id);
    if (birthday == null)
    {
        return View("NotFound");
    }
    return View("Edit", birthday);
}
```

```
@{
    ViewBag.Title = "Hittar inte födelsedagen";
}
<h1>
Hittar inte födelsedagen
</h1>
<p>
Födelsedagen du efterfrågar finns inte eller har blivit borttaget.
</p>
<p>
@Html.ActionLink("Tillbaka till lista med födelsedagar", "Index")
</p>
```

```
@model NextBirthday.Models.Birthday
@{
    ViewBag.Title = "Redigera födelsedag";
}
<h1>Redigera födelsedag</h1>
<using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true, "Fel inträffade. Korrigera det som är fel och försök igen.")
    <div class="editor-label">
        @Html.LabelFor(model => model.Name);
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.Name)
        @Html.ValidationMessageFor(model => model.Name)
    </div>
    <div class="editor-label">
        @Html.LabelFor(model => model.Birthdate);
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.Birthdate)
        @Html.ValidationMessageFor(model => model.Birthdate)
    </div>
    <p>
        <input type="submit" value="Spara" />
    </p>
    @Html.ActionLink("Tillbaka till lista med födelsedagar", "Index")
}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```


Men hur kan ett id skickas till en *action method* ?

- ✓ Med `Html.ActionLink`, där controllerklass, metod och parameter anges, kan en länk till en *action method* genereras.



Spara redigerat födelsedata

- ✓ Det redigerade födelsedatat postas till en *action method* med namnet `Edit`, som skapar ett `Birthday`-objekt, utifrån datat i formuläret, uppdaterar objektet, sparar det i databasen och visar ett rättmeddelande. Inträffar fel visas formuläret innehållande felmeddelande(n).

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(Birthday birthday)
{
    if (ModelState.IsValid)
    {
        try
        {
            _repository.UpdateBirthday(birthday);
            _repository.Save();

            return View("Saved", birthday);
        }
        catch (Exception)
        {
            ModelState.AddModelError(String.Empty,
                "Ett fel inträffade då födelsedagen skulle sparas.");
        }
    }

    return View("Edit", birthday);
}
```

```
@model NextBirthday.Models.Birthday

@{
    ViewBag.Title = "Födelsedagen sparad";
}

<h1>Födelsedagen sparad</h1>

<p>
    Födelsedagen för <strong>@Model.Name</strong>,
    <strong>@Model.Birthdate.ToShortDateString()</strong>, har sparats.
</p>
<p>
    @Html.ActionLink("Tillbaka till lista med födelsedagar", "Index")
</p>
```

Borttagning av födelsedata

- ✓ Borttagning av födelsedag följer i stora drag samma principer som redigering av födelsedata. MEN...
- ✓ ...det är viktigt att födelsedata inte kan tas bort från databasen med en GET-förfrågan. Det måste åtminstone göras med en POST-förfrågan varför någon form av bekräftelseformulär är lämpligt att använda.
- ✓ Det är även lämpligt att användaren får ett rättmeddelande då födelsedatat tagits bort.

```
@model IEnumerable<NextBirthday.Models.Birthday>
@{
    ViewBag.Title = "Nästa födelsedag";
}
<h1>
    Nästa födelsedag!
</h1>
@if (Model != null)
{
    <ul>
        @foreach (var item in Model)
        {
            <li><strong>@Html.DisplayFor(modelItem => item.Name)</strong> kommer att fylla
            år på en @item.NextBirthdayDate.ToString("ddd")
            <strong>om @Html.DisplayFor(modelItem => item.DaysUntilNextBirthday) dagar
            <span>
                @Html.ActionLink("Redigera", "Edit", new { id = item.BirthdayId })
                @Html.ActionLink("Ta bort", "Delete", new { id = item.BirthdayId })
            </span>
            </li>
        }
    </ul>
    <p>@Html.ActionLink("Lägg till födelsedag", "Create")</p>
}
```

```
public ActionResult Delete(int id = 0)
{
    var birthday = _repository.GetBirthdayById(id);
    if (birthday == null)
    {
        return View("NotFound");
    }
    return View("Delete", birthday);
}

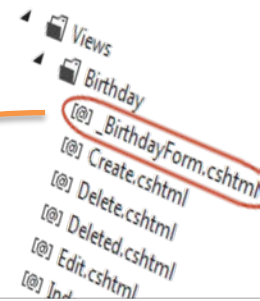
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    try
    {
        _repository.DeleteBirthday(id);
        _repository.Save();
        return RedirectToAction("Deleted");
    }
    catch (Exception)
    {
        ModelState.AddModelError(String.Empty,
            "Ett fel inträffade då födelsedagen skulle tas bort.");
    }
    return View("Delete", _repository.GetBirthdayById(id));
}
```

```
@model NextBirthday.Models.Birthday
@{
    ViewBag.Title = "Bekräfta borttagning av födelsedag";
}
<h1>Bekräfta borttagning av födelsedag</h1>
<p>
    Bekräfta att du vill ta bort <strong>@Model.Name</strong>, <strong>@Model.Birthdate.ToShortDateString()</strong>
</p>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary()
    <p>
        <input type="submit" value="Ta bort" />
    </p>
}
```

```
@{
    ViewBag.Title = "Födelsedagen borttagen";
}
<h1>Födelsedagen borttagen</h1>
<p>
    Födelsedagen har tagits bort.
</p>
<p>@Html.ActionLink("Tillbaka till lista med födelsedagar", "Index")</p>
```

Formulären för Create och Edit är lika!

- ✓ Då vyer, som Create och Edit, till stora delar har samma innehåll kan det gemensamma innehållet brytas ut och placeras i en partiell vy, *partial view*.



```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include="Name, Birthdate")] Birthday birthday)
{
    if (ModelState.IsValid)
    {
        try
        {
            // ...
        }
    }
}
```

Krävs eftersom vi i den partiella vyn har ett gömt fält för BirthdayId.

```
@model NextBirthday.Models.Birthday
@{
    ViewBag.Title = "Lägg till födelsedag";
}
<h1>Lägg till födelsedag</h1>
@Html.Partial("_BirthdayForm")
<p>
    @Html.ActionLink("Tillbaka till lista", "Index")
</p>
```

```
@model NextBirthday.Models.Birthday
@{
    ViewBag.Title = "Redigera födelsedag";
}
<h1>Redigera födelsedag</h1>
@Html.Partial("_BirthdayForm")
<p>
    @Html.ActionLink("Tillbaka till lista", "Index")
</p>
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

```
@model NextBirthday.Models.Birthday
using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true, "Fel inträffade. Korrigera det som är fel och försök igen.")
    <div class="editor-label">
        @Html.LabelFor(model => model.Name)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.Name)
        @Html.ValidationMessageFor(model => model.Name)
    </div>
    <div class="editor-label">
        @Html.LabelFor(model => model.Birthdate)
    </div>
    <div class="editor-field">
        @Html.EditorFor(model => model.Birthdate)
        @Html.ValidationMessageFor(model => model.Birthdate)
    </div>
    <p>
```