Jokes

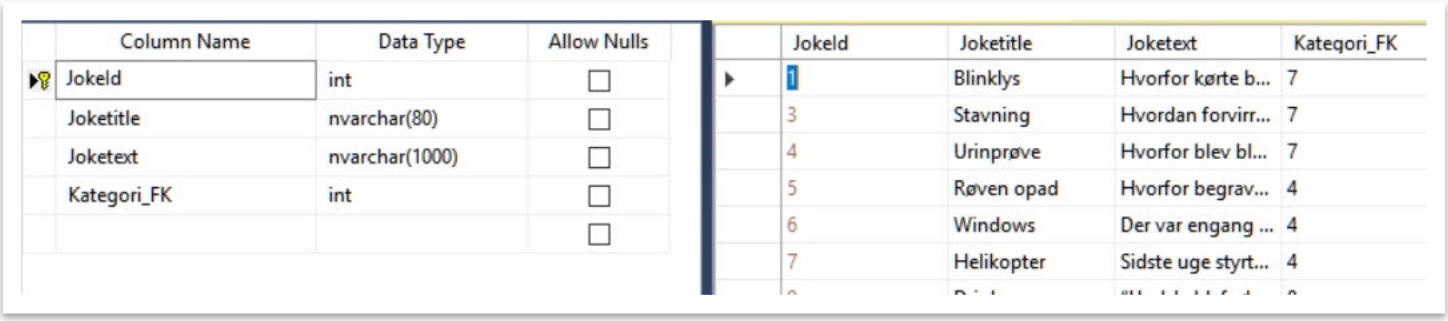
Opret, slet og ret - simpel database med **to tabeller** – jokes med kategori

Projektet her er et mini-projekt, hvor der er et image med. Det giver et simpelt overblik over, hvordan man kan håndtere billeder ved opret, slet og ret data (her er det memes) og et image.

Det særlige ved et billede er, at du udover at skulle **gemme navnet (= en stump tekst) i database-tabellen** (med mulighed for at rette i det og slette det) OGSÅ skal **uploade selve filen** (**jpg, gif, png** osv.) til serveren - til din image-mappe.

Databasen

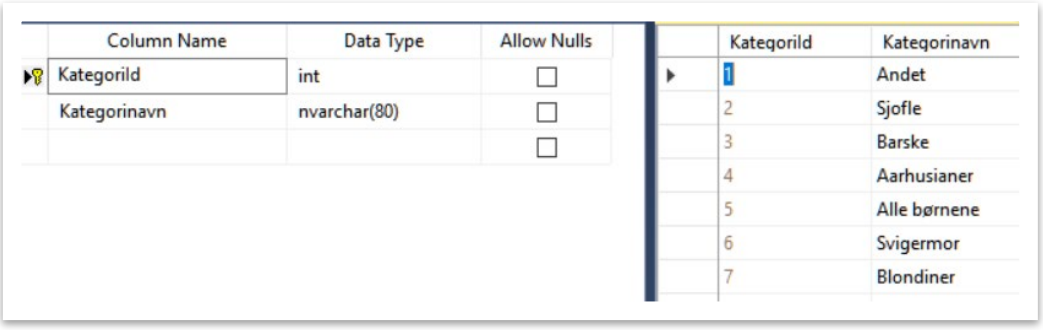
Joketabellen (med fremmednøgle til kategori-tabellen – Kategori_FK):



Column Name	Data Type	Allow Nulls
Jokeld	int	☐
Joketitle	nvarchar(80)	☐
Joketext	nvarchar(1000)	☐
Kategori_FK	int	☐
		☐

Jokeld	Joketitle	Joketext	Kategori_FK
1	Blinklys	Hvorfor kørte b...	7
3	Stavning	Hvordan forvirr...	7
4	Urinprøve	Hvorfor blev bl...	7
5	Røven opad	Hvorfor begrav...	4
6	Windows	Der var engang ...	4
7	Helikopter	Sidste uge styrt...	4

Kategoritabellen:



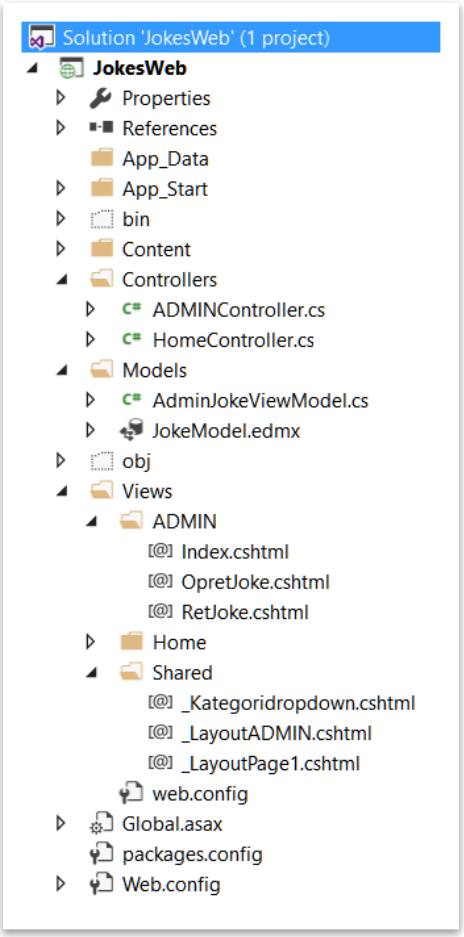
Column Name	Data Type	Allow Nulls
Kategorild	int	☐
Kategorinavn	nvarchar(80)	☐
		☐

Kategorild	Kategorinavn
1	Andet
2	Sjofle
3	Barske
4	Aarhusianer
5	Alle børnene
6	Svigermor
7	Blondiner

Lav evt. også en Bruger-tabel (User-table) med det samme – så har du den, når du senere vil tilføje login-kontrol på admin-delen.

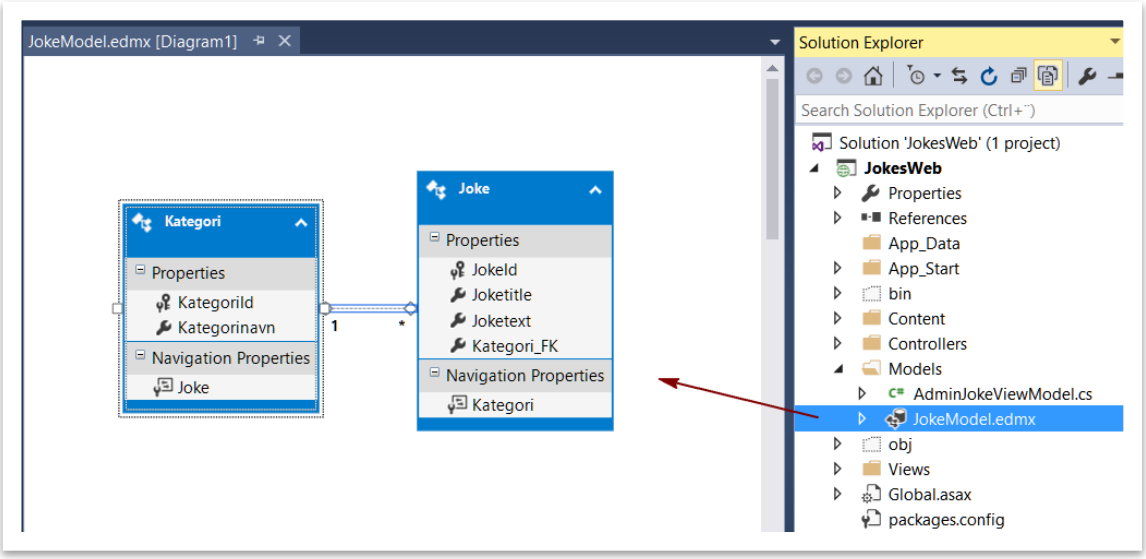
Webprojektet

Mapper og vigtige filer/klasser:



Modellen

Når du installeret Entity Framework til projektet og oprettet **modellen** (”ADO.NET Entity Data Model”) til databasen, skulle din model gerne se ca. sådan her ud:

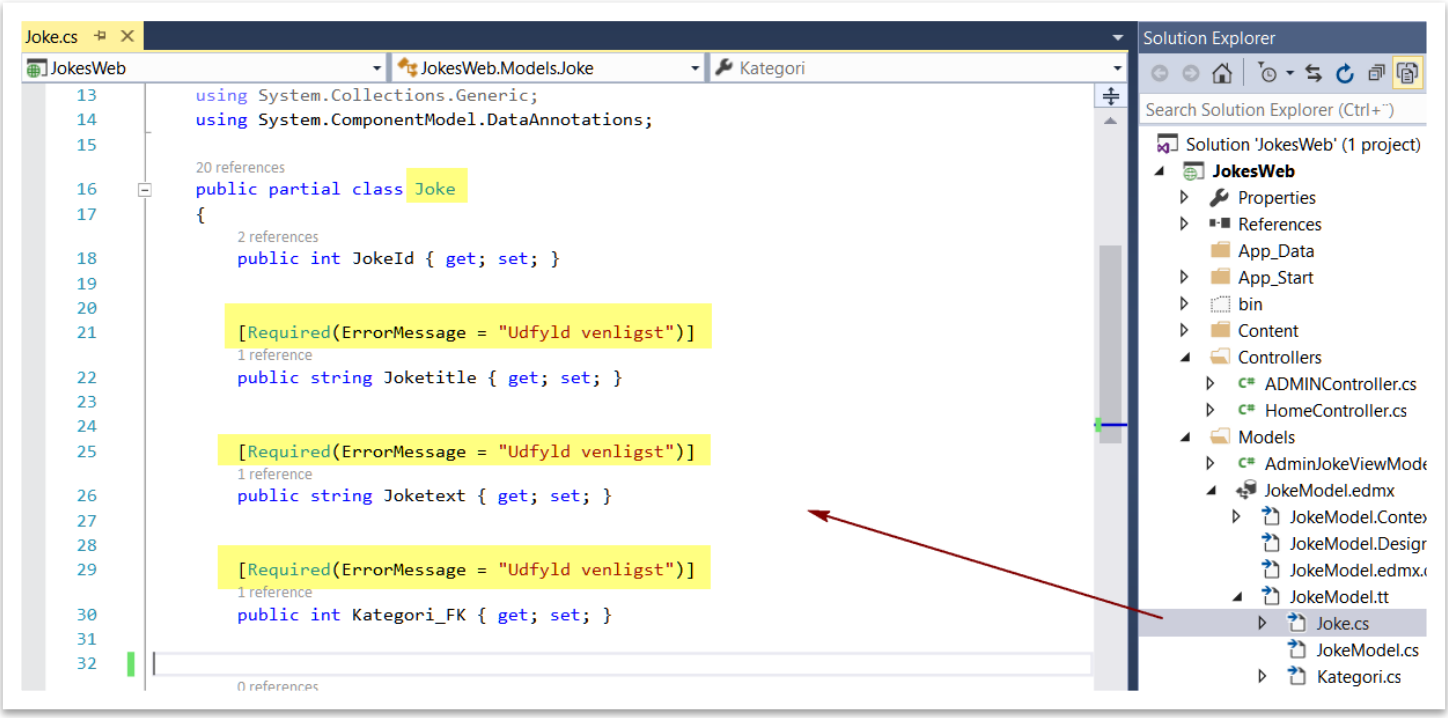


DataAnnotations (til validering af felterne)

Det er vigtigt, at du husker at markere "validerings-regler" ved properties i modellen. Altså hvilke data som SKAL være der (required), hvilken type af data der kræves (string, int, date, nullable osv.) … så det matcher database-tabellens design.

Du kan se en oversigt over mulighederne fx her: http://www.tutorialsteacher.com/mvc/implement-validation-in-asp.net-mvc og https://www.dotnettricks.com/learn/mvc/mvc-data-annotations-for-model-validation

Jeg har blot sat "**Required**" på alle felter, så man får en fejlbesked, hvis man forsøger at oprette en joke uden fx titel eller kategori … men udvid gerne med nogle mere gennemtænkte dataannotations.



PS! Hvis du laver en opdatering af din model ("Update Model from Database"), vil alle dataannotations blive overskrevet. Så husk at gemme en kopi inden opdatering.

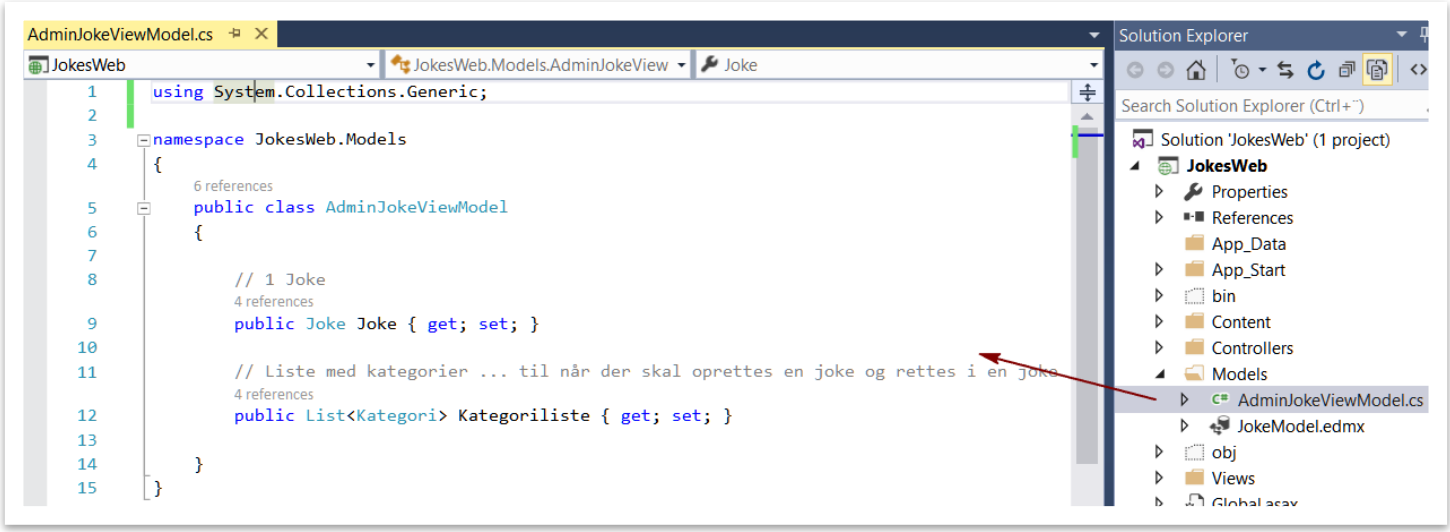
Eller følge rådet her: https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/enhancing-data-validation#add-metadata-classes

ViewModel til opret og ret joke

Fordi det at oprette en ny joke eller at rette i en eksisterende joke kræver, at kategorierne bliver vist (så man kan vælge eller omvælge en kategori til sin nye eller rettede joke) … så har vi brug for **både Joke-modellen og Kategori-modellen** på samme View.

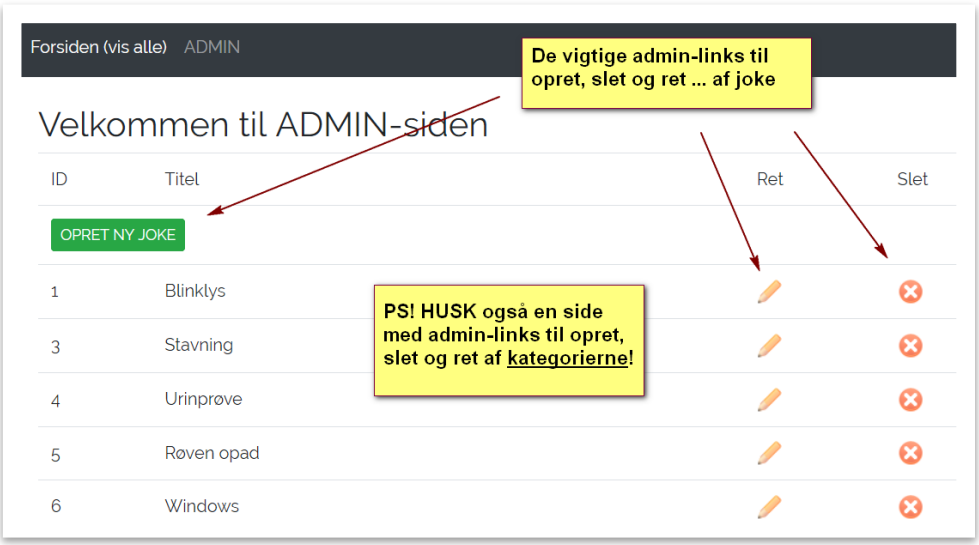
… og da man kun kan linke til 1 model i toppen af et View, så bliver vi nødt til at lave en ViewModel for gennem denne at få adgang til begge modeller.

Så lav en ViewModel i Model-mappen (højreklik Models > Add > Class … kald den fx ADMINJokeViewModel):



Startsiden til opret, slet og ret af jokes

Den sædvanlige – eller hvad du nu finder passende … ikke nødvendigt at bruge ViewModel her … bare brug Joke-modellen (vi skal jo ikke loope kategorierne ind):



Koderne til siden? Jokes-admin-startsiden her er bygget op på fuldstændig samme måde som i de tidligere projekter, **så tjek et af dine tidligere projekter** (eller koderne til Memes – findes som en beskrivelse på VIDOnline) …

Opret ny joke

Opret, ret og slet har alle brug for 2 action-metoder: En til det, der skal vises, når siden loader. Og en til at håndtere det "post" som kommer fra formularen, når der klikkes på submit-button … den med `[HttpPost]`

Action-metoderne til OPRET - i ADMINController

Bemærk, at vi her **bruger vores ViewModel** … som vi klargør med en ny tom joke (som skal udfyldes med den nye joke) + fylder ud med alle kategorier fra databasens kategori-tabel:

```
// ***** OPRET NY ************
// ****************************************************************************
0 references
public ActionResult OpretJoke()
{
    // Ny tom joke (med liste af kategorier) til udfyldelse
    AdminJokeViewModel NyJoke = new AdminJokeViewModel();
    NyJoke.Joke = new Joke();                       // Tom joke-model til at udfylde når der skal oprettes en joke
    NyJoke.Kategoriliste = Db.Kategori.ToList();    // Liste af kategorier fra DB
    return View(NyJoke);
}
```

Og så har vi post-delen … når der er klikket på submit-button = først tjekke om alt er korrekt udfyldt … og hvis det er, så skal den nye joke gemmes i databasen:

```
[HttpPost]
0 references
public ActionResult OpretJoke(AdminJokeViewModel NyJoke)
{
    if (!ModelState.IsValid)
    {
        ViewBag.Besked = "Noget gik galt - prøv igen";
        NyJoke.Kategoriliste = Db.Kategori.ToList(); // Genfyld kategorilisten
        return View(NyJoke); // Retur til formularen - send det allerede-udfyldte med retur
    }

    // Formularen er korrekt udfyldt
    // - joke (fra viewmodel) gemmes i DB
    Db.Joke.Add(NyJoke.Joke);
    Db.SaveChanges();

    TempData["Besked"] = "Joke er oprettet!!!";

    return RedirectToAction("Index", "ADMIN");
}
```

Bemærk at ActionResult'et modtager ViewModel'en … det der (`AdminJokeViewModel JokeRettes`)

Viewet (OpretJoke.cshtml i ADMIN-mappen)

Her laver vi en select-dropdown med options … hvor kategorierne bliver loopet ind i … så man kan vælge en kategori til den nye joke … https://www.w3schools.com/tags/tag_select.asp

```
@model JokesWeb.Models.AdminJokeViewModel

@{
    ViewBag.Title = "OpretJoke";
    Layout = "~/Views/Shared/_LayoutADMIN.cshtml";
}

<h1>Opret ny joke</h1>

<form action="~/ADMIN/OpretJoke" method="post">

    @* Samlet valideringsbesked - tilføj evt. individuel validering ved hvert input - se tidligere projekter *@
    <div class="text-danger">@Html.ValidationSummary()</div>

    @* *** Joke-titel *** *@
    <div class="form-group">
        <label for="Joketitle">Overskrift:</label>
        <input type="text" name="Joke.Joketitle" id="Joketitle" value="@Model.Joke.Joketitle" placeholder="Skriv en titel" class="f
    </div>

    @* *** Joke-tekst *** *@
    <div class="form-group">
        <label for="Joketext">Joken:</label>
        <textarea name="Joke.Joketext" id="Joketext" class="form-control">@Model.Joke.Joketext</textarea>
    </div>

    @* *** Joke-kategori *** *@
    <div class="form-control">
        <label for="Kategori">Vælg nyhedskategori:</label>
        <select id="Kategori" name="Joke.Kategori_FK" class="form-control">

            @* Loop alle kategorier ud som options i select-dropdown'en *@
            @foreach (var item in Model.Kategoriliste)
            {
                <option value="@item.KategoriId">
                    @item.Kategorinavn
                </option>
            }

        </select>
    </div>

    @* *** SUBMIT ****@
    <button type="submit" class="btn btn-primary">OPRET NY</button>

</form>
```

Tjek, at det virker … at du kan oprette en joke uden fejl. Ret fejl om nødvendigt inden du går videre.

Slet joke

Når du skal slette en joke, er kategorierne ikke vigtige – du skal jo ikke vise brugeren alle kategorierne på siden for at han kan slette en joke. Så du har ikke brug for ViewModel til slet-funktionen … du kan nøjes med at bruge Joke-modellen. Og derfor ligner denne del så meget det tidligere, du har lavet.

Så kig i og genbrug koderne fra et tidligere projekt fx Citat-projektet eller Skuespiller-projektet, hvor du allerede har lavet de to slet-action-metoder + viewet. Det skal laves stort set på samme måde her.

Ret (rediger/update) joke

Opret, ret og slet har alle brug for 2 action-metoder: En til det, der skal vises, når siden loader. Og en til at håndtere det "post" som kommer fra formularen, når der klikkes på submit-button … den med `[HttpPost]`

Action-metoderne til RET – i ADMINController

Igen skal vi sende alle kategoerine med ud til viewet … og så selvfølgelig den joke, som skal rettes …

```
// ***** RET *****************
// ****************************************************************************
0 references
public ActionResult RetJoke(int? Id) // Id = den joke som skal rettes
{

    // Hvis der ikke er en Id med, så spark tilbage til index-siden
    if (Id == null)
        return RedirectToAction("Index");

    // Lav tom viewmodel - klar til at blive udfyldt med joken (som skal rettes) + kategorilisten
    AdminJokeViewModel JokeRettes = new AdminJokeViewModel();

    // Find Joke som måske skal rettes og put joken i viewmodel
    JokeRettes.Joke = Db.Joke.Find(Id);

    // Hvis der ikke blev fundet en Joke som matcher id
    if (JokeRettes == null)
        return RedirectToAction("Index");

    JokeRettes.Kategoriliste = Db.Kategori.ToList(); // Fyld kategorier i viewmodels kategoriliste

    return View(JokeRettes);
}
```

Og når der er klikket på submit-knappen = post:

```
[HttpPost]
0 references
public ActionResult RetJoke(AdminJokeViewModel JokeRettes)
{
    // Hvis det ikke er udfyldt korrekt
    if (!ModelState.IsValid)
    {
        ViewBag.Besked = "Noget gik galt - prøv igen";
        JokeRettes.Kategoriliste = Db.Kategori.ToList(); // Genfyld kategorilisten
        return View(JokeRettes); // Retur til formularen - send det allerede-udfyldte med retur
    }

    // Gem i DB
    Db.Joke.AddOrUpdate(JokeRettes.Joke);
    Db.SaveChanges();

    TempData["Besked"] = "Joke er rettet";
    return RedirectToAction("Index", "ADMIN");

}
```

Viewet (RetJoke.cshtml i ADMIN-mappen)

```
@model JokesWeb.Models.AdminJokeViewModel

@{
    ViewBag.Title = "Ret joke";
    Layout = "~/Views/Shared/_LayoutADMIN.cshtml";
}

<h1>Ret joke</h1>

<form action="~/ADMIN/RetJoke" method="post">

    @* Samlet valideringsbesked - tilføj evt. individuel validering ved hvert input - se tidligere projekter *@
    <div class="text-danger">@Html.ValidationSummary()</div>

    @* *** Joke-ID *** *@
    <div class="form-group">
        <label for="JokeId">ID:</label>
        <input type="text" readonly name="Joke.JokeId" id="JokeId" value="@Model.Joke.JokeId" placeholder="Husk titel" class="form-control" />
    </div>

    @* *** Joke-titel *** *@
    <div class="form-group">
        <label for="Joketitle">Overskrift:</label>
        <input type="text" name="Joke.Joketitle" id="Joketitle" value="@Model.Joke.Joketitle" placeholder="Husk joke" class="form-control" />
    </div>

    @* *** Joke-tekst *** *@
    <div class="form-group">
        <label for="Joketext">Joken:</label>
        <textarea name="Joke.Joketext" id="Joketext" class="form-control">@Model.Joke.Joketext</textarea>
    </div>

    @* *** Joke-kategori *** *@
    <label for="Kategori">Vælg jokekategori:</label>
    <select id="Kategori" name="Joke.Kategori_FK" class="form-control">
        @foreach (var item in Model.Kategoriliste)
        {
            <option value="@item.KategoriId" @(item.KategoriId == Model.Joke.Kategori_FK ? "selected='selected'" : "")>
                @item.Kategorinavn
            </option>
        }
    </select>

    @* *** SUBMIT ***@
    <button type="submit" class="btn btn-primary">RET JOKE</button>

</form>
```

Det her er en if-sætning, hvor vi sætter "selected" på den af kategori (ID) som svarer til jokens kategori (kategori_FK) ...

Så man kan aflæse hvilken kategori der er valgt/selected til joken nu ... med mulighed for at vælge en anden kategori.

... og den findes selvfølgelig ikke på OpretJoke ... da vi ikke på forhånd kan vælge/selecte en kategori til en ny joke.

Tilføj selv de manglende dele – fx:

- **Slet joke** … brug koderne fra et tidligere projekt som inspiration (og læs afsnittet ”Slet joke” herover et sted)
- **Lav login-side** og sørg for, at man kun har adgang til ADMIN-delene, hvis man er logget ud – husk også en **logud-knap**