Översikt

- Skapa Views genom att använda Razor syntax.
- Använda HTML Helpers.
- Återanvända kod i Views.

.eosec

Lektion 1: Skapa Views genom att använda Razor Syntax

- Views.
- Flera views.
- Skapa view.
- Razor view.
- Server Side Code.
- Rendera HTML för alla.
- Alternativa View Engines.
- Skapa egen View Engines.

.eosec

Views

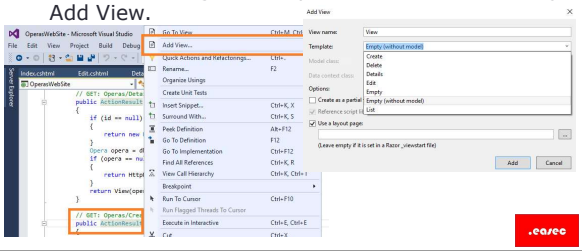
- I MVC finns det vanligtvis en controller för varje klass.
- Undantaget är Home controller, denna är inte knuten till någon klass.
- Controller kan ha en eller flera views.
- Views skapas i katalogen View.
- Om du använder Razor view engine och C#, kommer dessa att få filändelse .cshtml.

Flera views

- Controller kan ha flera views.
- Exempelvis för Produkt:
  - *Details*, visar produkt med namn, artikelnummer och annan information.
  - *Create*, ger möjlighet för användare att skapa produkt.
  - *Edit*, ger möjlighet för användare att editera produkt.
  - *Delete*, ger möjlighet för användare att ta bort produkt.
  - *Index*, kan visa alla produkter.

Skapa view

- I Solution Explorer, under Views, välj katalog som representerar namn för controller, högerklicka och välj Add - View.
- Alternativt: högerklicka på händelse i metod, välj Add View.



Razor view

- Många views som är baserade på Razor, är designade för att visa egenskaper från specifik modelklass.
- Om du knyter dessa views till klass som de skall visa, får du extra hjälp av IntelliSence.
- Andra views kan visa egenskaper från olika modelklasser i olika fall, eller använder ingen klass alls, kallas för dynamic views.

Razor view (forts.)

- Razor view för specifik Modelklass:
  - Strongly-Typed view.
- Från flera klasser eller ingen klass alls:
  - Dynamic view.

Server Side Code

- Razor identifierar server-side code genom att leta efter @ symbolen.
- I syntax för Razor används @ olika:
  - @ för att identifiera C#-baserad server-side code.
  - @@ används för att rendera @ på HTML-sida.
  - @: används för att deklarera rad av text som just text.
  - <text> används för att deklarera rader av text.
- För att rendera text utan HTML-kodning, använd Html.Raw() helper.
- Exempel:

```
<span>Priset inkluderat med moms:
      @(Model.Price * 1.25)
</span>
```

Rendera HTML för alla

- Internet är för alla, även för de som har funktionshinder.
- Tänk på dessa när du skapar dina views.
- Några råd:
  - Använd attributet alt för visuella objekt.
  - Använd inte färger för att förhöja innehåll.
  - Använd tabeller sparsamt, inga nestade tabeller.
  - Undvik att använda bilder som innehåller viktig text.
  - Använd tillsammans med style sheets för att positionera ut innehåll.
  - Lägg all viktig text i HTML-element.

.easec

Alternativa View Engines

- View engine är den komponenten i ramverket MVC, som ansvarar för att lokalisera fil för view, köra server-side kod och rendera HTML-sida så att webbläsare kan visa den för användare.
- Razor är standardmotor i MVC, men det finns alternativ.

.easec

Skapa egen View Engines

- Det finns möjlighet att skapa egen view engine.
- Men denna möjlighet används sällan, då de motorer som finns är kraftfulla, flexibla och lätta att lära.
- Men om du skapar din egen motor, glöm inte att registrera denna i filen Global.asax.

.easec

Lektion 2: Använda HTML-helpers

- Vad är HTML-helpers?
- Användning av HTML-Helpers.
- Html.ActionLink() Helper.
- Html.Action() Helper.
- Använda helpers för att visa egenskaper.
- Begin Form Helper.
- @using.
- Editor Helper.
- Html.LabelFor() och Html.EditorFor().
- Helpers för validering.

Vad är HTML-Helpers?

- Ramverket för MVC inkluderar många funktioner som kallas för HTML-helpers, dessa används i views.
- HTML-helpers är funktioner som du kan kalla på som renderar värden, labels och olika typer av input controls exempelvis textrutor.
- Dessa innehåller logik som gör det lättare att rendera HTML för en Modellklass:s egenskaper, men kan även användas till andra uppgifter.

Användning av HTML-Helpers

- HTML-helpers är enkla metoder som i de flesta fallen returnerar en sträng.
- Denna sträng är en liten sektion av HTML som motor för view kan lägga till i fil för view, för att skapa en komplett HTML-sida.
- Du kan skriva en view som renderar vilket HTML-innehålls som helst utan att använda HTML-helpers, men HTML-helpers hjälper till med arbetsuppgiften för vanliga scenarios.

Html.ActionLink() Helper

- När en användare gör en begäran i en MVC-applikation, kommer MVC-ramverket att skicka vidare begäran till rätt controller och händelse.
- Vanligtvis kommer användare att klicka på länk, Html.ActionLink() helper kan användas för att hjälpa till med att rendera dessa länkar.
- Metoden kommer att returnera ett <a> element med de korrekta värde för href, när användare klickar på en länk på webbsidan.

.eo/ee

Html.ActionLink() Helper -exempel

- Html.ActionLink()

```
@Html.ActionLink("Klicka är för att visa bild 1",
  "Display", new { id = 1 })
```

```
<a href="/photo/display/1">
   Klicka här för att visa bild 1
</a>
```

- Html.ActionLink() helper arbetar med routing engine för att skapa rätt URL för länk.
- Länk ovan kommer att renderas under förutsättning att den kallas från controller med namnet Photo.

.eo/ee

Html.Action() Helper

- Html.Action()

```
<img alt="Denna bild kommer från en händelse"
   src="@Url.Action("GetImage", new { id = 1 })" />
```

```
<img
   alt="Denna bild kommer från en
händelse"
   src="/photo/getimage/1" })"
/>
```

- Html.Action() helper arbetar med routing engine för att skapa rätt URL för länk.
- Länk ovan kommer att renderas under förutsättning att den kallas från controller med namnet Photo.

.eo/ee

Html.ActionLink() och Html.Action()

- Du vill rendera HTML5 <video> tag, för att spela upp en film från en händelse. Vill du använda HtmlActionLink() eller HtmlAction()?

Använda helpers för att visa egenskaper

- MVC innehåller ett antal helpers som kan visa egenskaper från modelklasser.
- Dessa kan användas i view för att visa information om produkter, kommentarer, användareinformation etc.
- HtmlDisplayNameFor() renderar namn för egenskap.
- HtmlDisplayFor() renderar värde för egenskap.

Html.DisplayNameFor() och Html.DisplayFor Helper -exempel

- Html.DisplayNameFor()

```
@Html.DisplayNameFor(model => model.CreatedDate)
```

Created Date

- Html.DisplayFor()

```
@Html.DisplayFor(model => model.CreatedDate)
```

24/12/2016

Begin Form Helper

- För att ta emot indata från användare, måste tillhandahålla ett formulär.
- Typiskt formulär innehåller ett antal labels, som indikerar för användare vilken typ av information som skall ges in.
- Innehåller även input controls och även en knapp för att skicka.
- I HTML så måste formulär omslutas av <form>-tagg.
- I en MVC-view används Html.BeginForm() tillsammans med @using.

.eo/ee

@using

- @using kommer att rendera avslutande </form> tagg tillsammans med avslutande parantes för att försäkra dig om att formuläret skrivs på ett korrekt sätt.

.eo/ee

Html.BeginForm() -exempel

- Html.BeginForm()

```
@using (Html.BeginForm("Create", "Photo",
   FormMethod.Post,
   new { enctype = "multipart/form-data" }))
{
@* Placera input controls här*@
}
```

```
<form action="/Photo/Create" method="post"
   enctype="multipart/form-data">

</form>
```

.eo/ee

Editor Helper

- I ett HTML formulär finns många olika typer av HTML input controls, som kan användas för att ta emot indata från användare.
- I view baserad på Razor används Html.LabelFor() och Html.EditorFor() för att lättare rendera rätt input control för en egenskap i modelklass.

.easec

Html.LabelFor()

- Html.LabelFor()

```
@Html.LabelFor(model => model.ContactMe)
```



```
<label for="ContactMe">
  Contact Me
</label>
```

.easec

Html.EditorFor()

- Html.EditorFor() kommer att rendera rätt input control för egenskap i modelklass.
- Kommer att rendera <input type=”text”> för sträng.
- [DataType(DataType.MultilineText)] kommer att renderas som ett <textarea> element.

```
@Html.EditorFor(model => model.ContactMe
```



```
<input type="checkbox"
   name="Description">
```

.easec

Helpers för validering

- När du begär information från användare, vill du oftast försäkra dig om att det är korrekt information.
- Html.ValidationSummary() och Html.ValidationMessageFor() kan användas för detta.

Html.ValidationSummary() -exempel

- Html.ValidationSummary() kommer att rendera en summering av felaktigheter i ett formulär, i form av en punktlista.

```
@Html.ValidationSummary()
```

```
<ul>
  <li>Ange efternamn</li>
  <li>Ange korrekt e-post adress</li>
</ul>
```

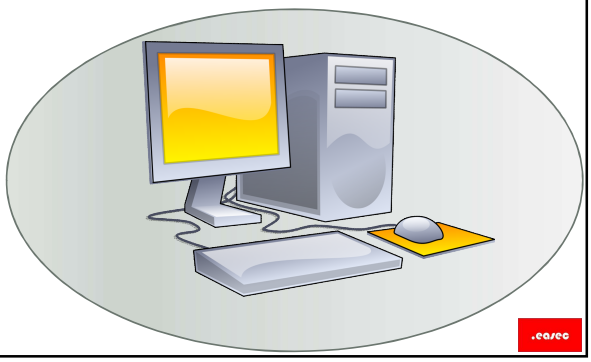
Html.ValidationMessageFor() -exempel

- Html.ValidationMessageFor() kommer att rendera ett felmeddelande bredvid input control.

```
@Html.ValidationMessageFor(model => model.Email)
```

```
Ange korrekt e-post adress
```

Övning Använda HTML Helpers



Lektion 3: Återanvända kod i views

- Skapa Partial Views.
- Använda Partial Views.

Skapa Partial Views

- Partial Views används för att rendera identiska eller liknande HTML på olika platser i din webbapplikation.
- I Visual Studio används samma guide som att skapa view, kryssa i boxruta Create as a Partial View.
- View kommer att skapas utan <head> och <body>, eller någon annan tagg.
- Strongly typed används för samma model, dynamisk om annan model används.

Använda Partial Views

- Genom att använda HTML helpers kan du använda partial views inuti andra views i en webbapplikation.
- För att skicka samma objekt i model från parent view till child view används Html.Partial().
- För att skicka ett objekt i model som är annorlunda från parent view eller annan modelklss används Html.Action().
- Använd samlingarna ViewBag eller ViewData för att dela information mellan händelser i controller, eller parent view eller partial view.

.easec

Repetitionsfrågor

.easec