

Enkel gästbok 1.0

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 Loock, förutom Linnéuniversitetets logotyp och symbol samt ikoner och fotografier, 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

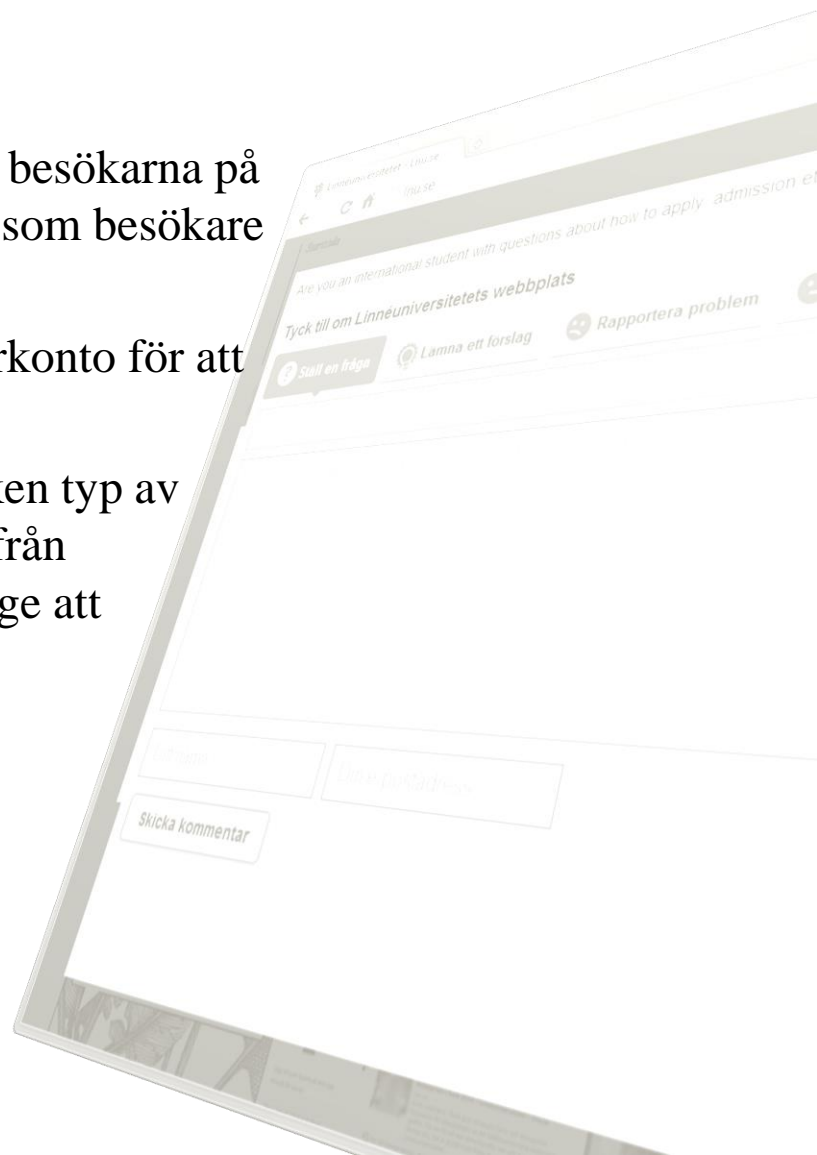
Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol samt ikoner och fotografier i din nya version!

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 är en digital gästbok?

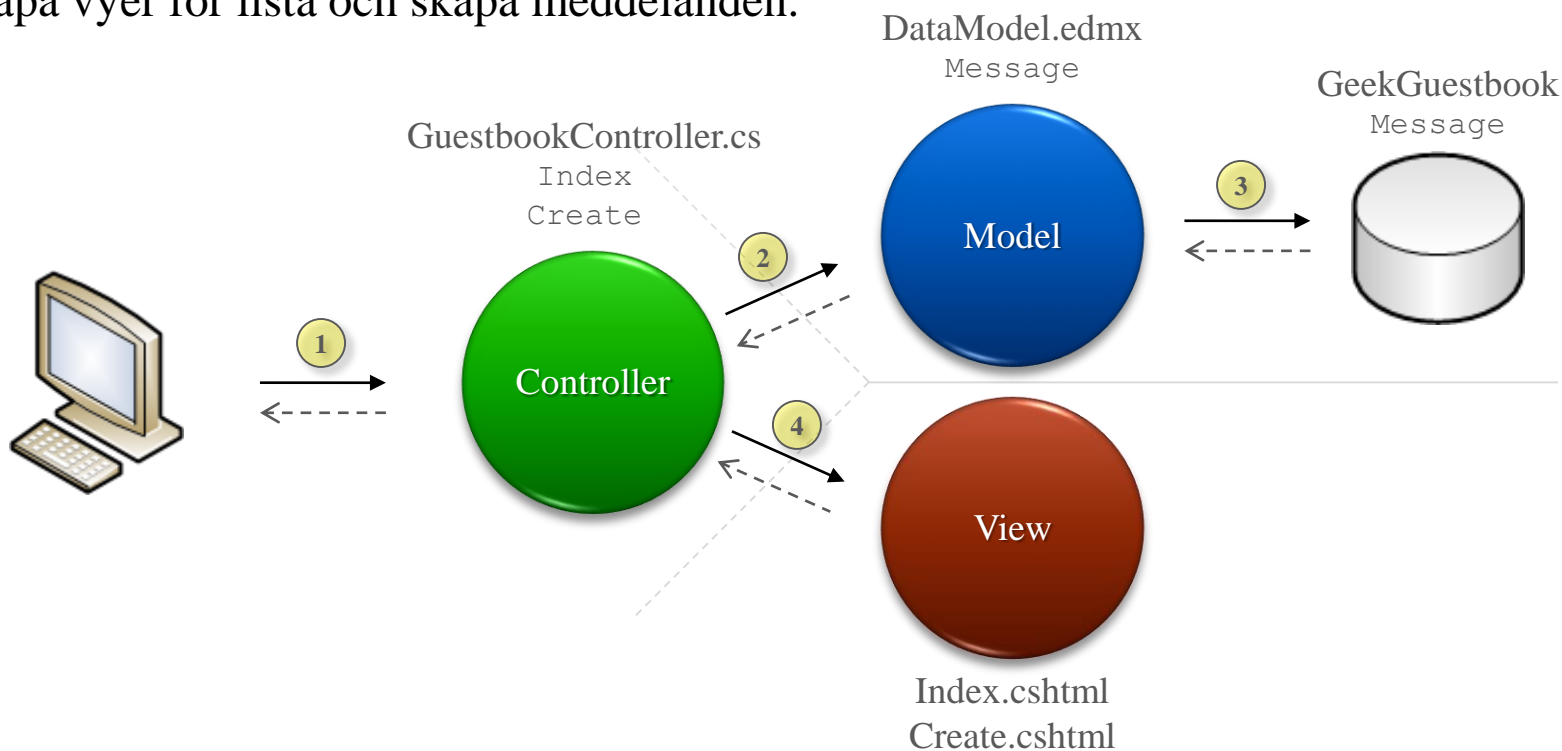
- ✓ På webben är en gästbok ett system som tillåter besökarna på en webbplats att lämna ett publikt meddelande, som besökare och webbansvarig kan läsa.
- ✓ I allmänhet behöver besökaren inte ett användarkonto för att lämna ett meddelande.
- ✓ Syftet med en webbplats gästbok är att visa vilken typ av besökare webbplatsen har, och få återkoppling från dem. Detta gör det möjligt för den webbansvarige att förbättra webbplatsen.

Meddelanden ska kunna visas och skickas in. Vi behöver kunna lagra en kort rubrik, meddelandets innehåll, namn, mejladress och datum med tidpunkt i en databas.



Gästbokapplikationen i stora drag

- ✓ Skapa databas, GeekGuestbook, med tabellen Message innehållande meddelanden.
- ✓ Skapa datamodell innehållande klassen Message.
- ✓ Skapa controller med metoder för lista (Index) och skapa (Create) meddelanden.
- ✓ Skapa vyer för lista och skapa meddelanden.



Skapa databasen

- ✓ Skapa databasen GeekGuestbook.
- ✓ Lägg till användaren appUser och se till att användaren har rollen db_owner. All kommunikation med databasen kommer att gå genom appUser.
- ✓ Lägg till tabellen Message med fälten:
 - MessageId, för primärnyckelns värde.
 - Header, för rubriken.
 - Body, för meddelandet.
 - Name, för eventuellt namn.
 - Email, för eventuell mejladress.
 - Created, datum då meddelandet skapades.

The screenshot shows the SQL Server Enterprise Manager interface. The 'GeekGuestbook' database is selected, and the 'dbo.Message' table is highlighted. The 'Users' folder shows the 'appUser' user, which is also highlighted. The 'Database role membership' window is open, showing the 'db_owner' role assigned to the 'appUser'.

Message Table Structure:

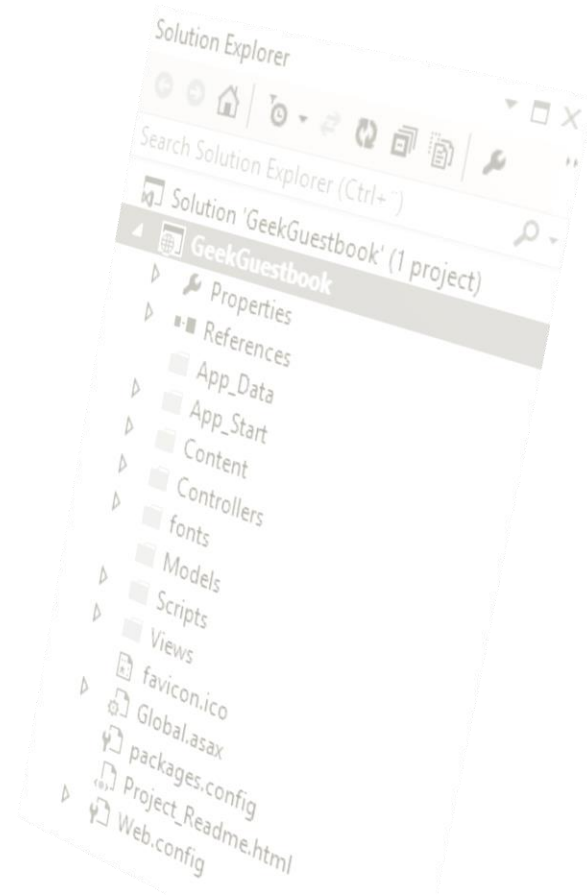
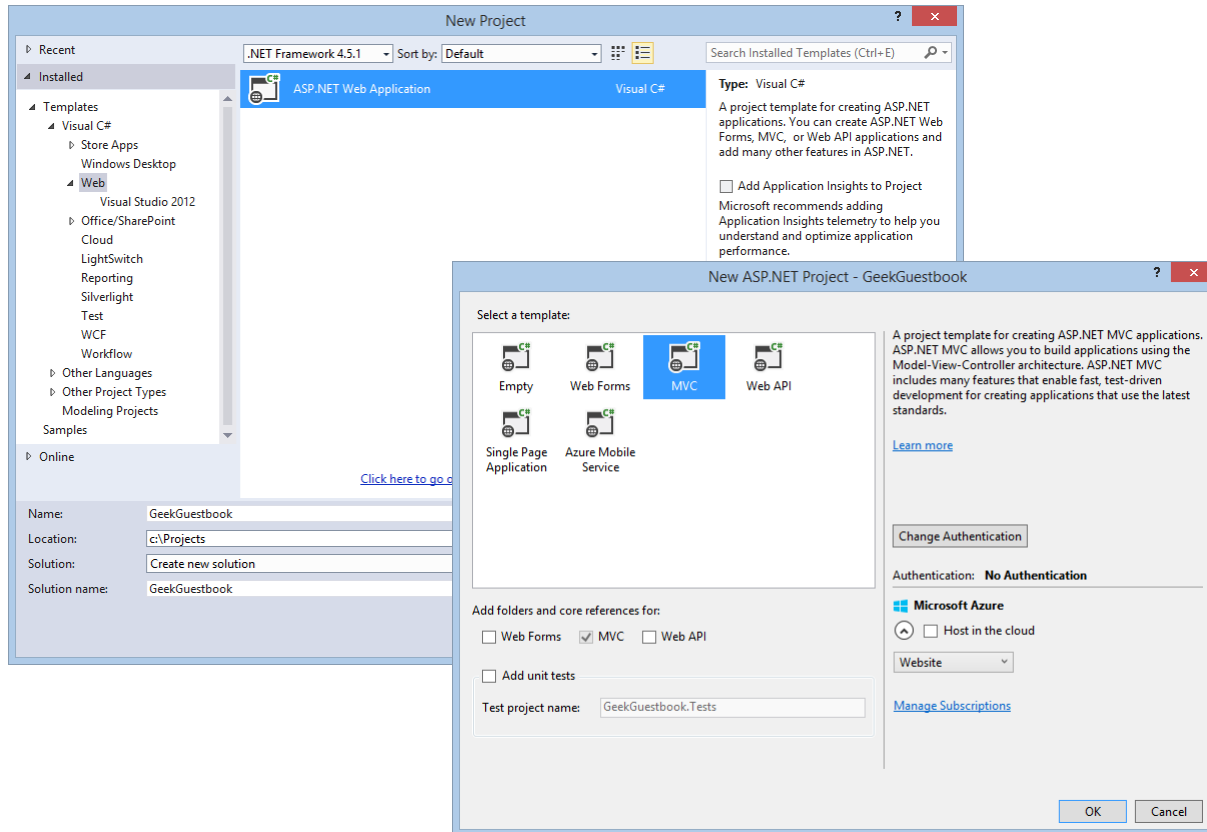
Column Name	Condensed Type	Nullable
MessageId	int	No
Header	varchar(50)	No
Body	varchar(500)	No
Name	varchar(50)	Yes
Email	varchar(50)	Yes
Created	datetime2(0)	No

Database role membership:

Role	Members
db_datawriter	<input type="checkbox"/>
db_ddladmin	<input type="checkbox"/>
db_denydatareader	<input type="checkbox"/>
db_denydatawriter	<input type="checkbox"/>
db_owner	<input checked="" type="checkbox"/>
db_securityadmin	<input type="checkbox"/>

Skapa ASP.NET MVC-projektet

- ✓ Skapa ett nytt projekt av typen **ASP.NET Web Application**, ge det namnet **GeekGuestbook** och välj projektmallen **MVC**.



Skapa modell

- ✓ Med hjälp av *Entity Data Model Wizard* genereras modellen med utgångspunkt från databasen.

1. Add New Item - GeekGuestbook

2. Entity Data Model Wizard - What should the model contain?

3. Entity Data Model Wizard - Choose Your Data Connection

4. Connection Properties

5. Entity Data Model Wizard - Choose Your Database Objects and Settings

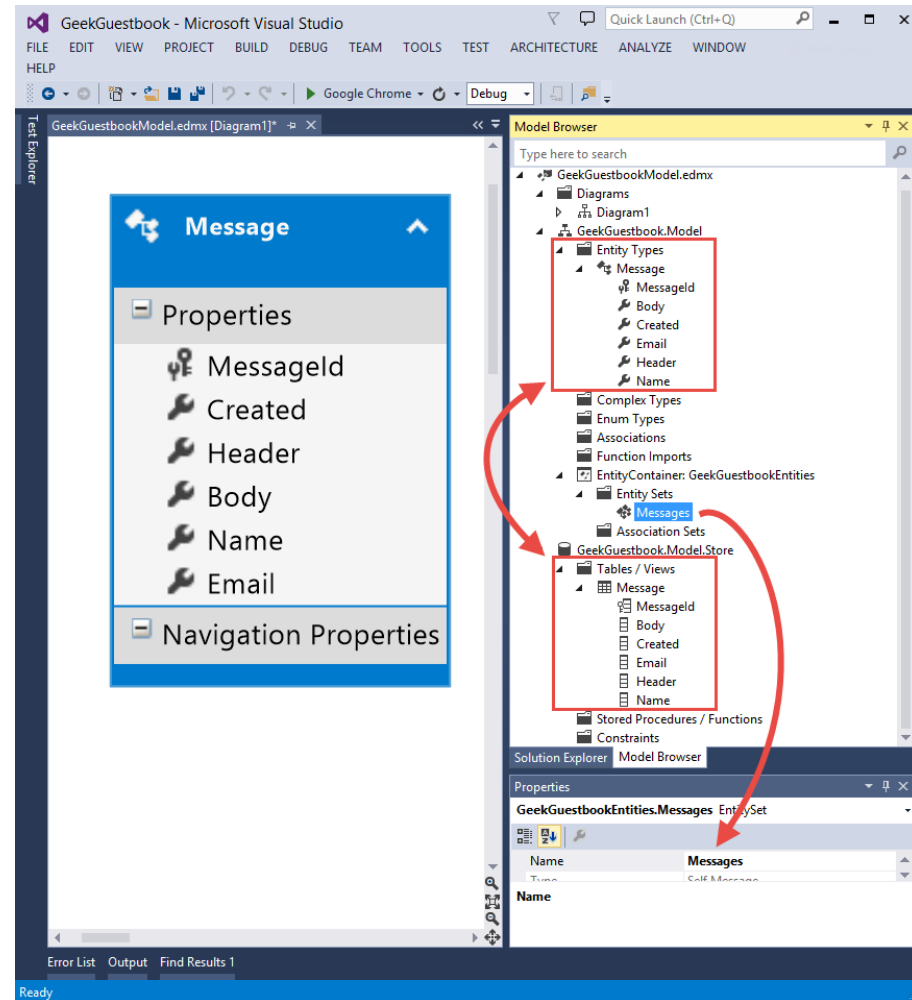
6. Solution Explorer

Var uppmärksam på namnrymden!

Entity Framework Object Relational Designer

- ✓ .edmx-filen *Entry Data Model Wizard* skapade kan redigeras med *Entity Framework Object Relational Designer*.
- ✓ För tabellen Message har entitetstypen Message skapats, d.v.s. samma namn som tabellen. Fälten i tabellen har implementerats som egenskaper i klassen.
- ✓ Klassers och egenskapers namn kan modifieras.
- ✓ I egenskaperna för entitetssamlingen kan namnet för samlingen ändras, ibland lämpligen till namnet på klass med suffixet Set.

(Vissa substantiv som t.ex. *aircraft*, heter samma sak i singular som i plural.)



Controller med ansvar för gästboken

- ✓ Namnet på controllern måste avslutas med just Controller.
- ✓ Metoden Index returnerar ett objekt av typen ViewResult.

The screenshot illustrates the process of creating a new MVC controller in Visual Studio. The 'Add Scaffold' dialog is open, showing a list of scaffolding templates. The 'MVC 5 Controller - Empty' template is selected and highlighted with a red box. Below the list, a link is provided: [Click here to go online and find more scaffolding extensions.](#)

Overlaid on the dialog is a code editor showing the generated code for the 'GuestbookController'. The code is as follows:

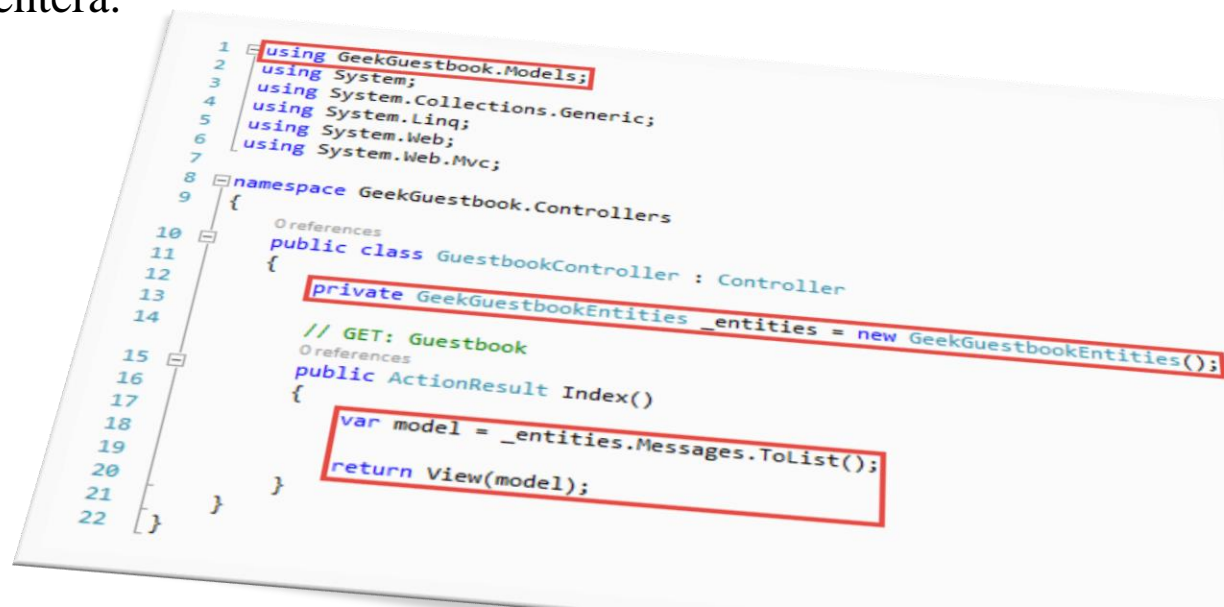
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace GeekGuestbook.Controllers
8 {
9     // GET: Guestbook
10     public class GuestbookController : Controller
11     {
12         // GET: Guestbook
13         public ActionResult Index()
14         {
15             return View();
16         }
17     }
18 }
```

Below the code editor, the 'Add Controller' dialog is open, showing the 'Controller name' field with the text 'GuestbookController' entered. The 'Add' button is highlighted with a red box.

In the background, the 'Solution Explorer' is visible, showing the project structure for 'GeekGuestbook'. The 'Controllers' folder is expanded, showing the 'GuestbookController.cs' file.

Skicka med modell med meddelanden till vyn

- ✓ Klassen `GeekGuestbookEntities`, som *Entry Data Model Wizard* skapade automatisk, kapslar in databasen. Data kan hämtas från databasen via en instans av klassen.
- ✓ Modellen utgörs av ett objekt av typen `List<Message>`, som skapas med hjälp av egenskapen `Messages`.
- ✓ Med till vyn skickas modell innehållande listan med referenser till `Message`-objekt som vyn ska presentera.



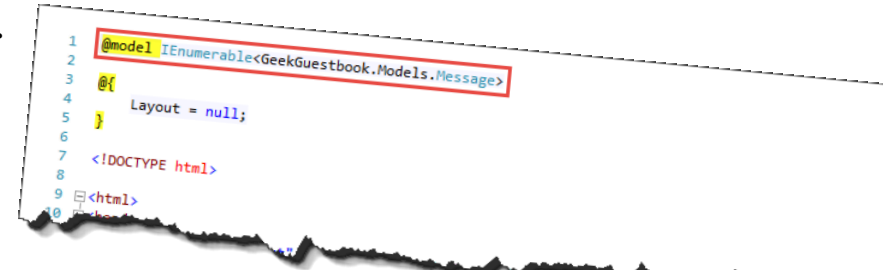
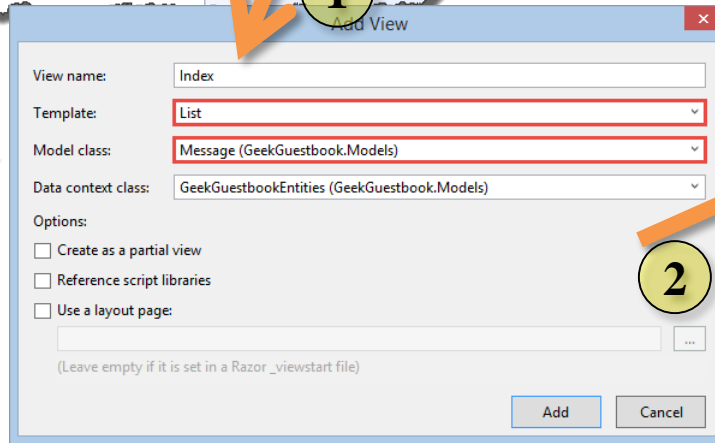
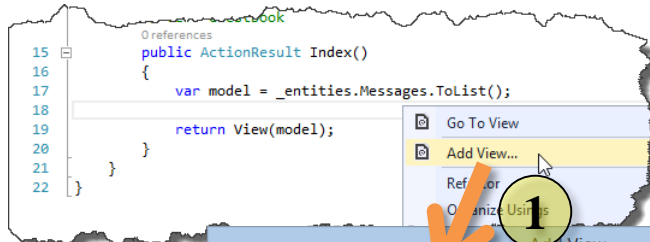
```

1  using GeekGuestbook.Models;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Web;
6  using System.Web.Mvc;
7
8  namespace GeekGuestbook.Controllers
9  {
10     References
11     public class GuestbookController : Controller
12     {
13         private GeekGuestbookEntities _entities = new GeekGuestbookEntities();
14
15         // GET: Guestbook
16         References
17         public ActionResult Index()
18         {
19             var model = _entities.Messages.ToList();
20             return View(model);
21         }
22     }

```

Starkt typad vy för meddelanden

- ✓ En starkt typad vy underlättar arbetet med det data modellen innehåller. I detta fall refererar egenskapen `Model` till listan, av typen `IEnumerable<Message>`, innehållande referenser till `Message`-objekt.



För att modellklasser ska visas måste projektet ha kompilerats.

Formulär för att skriva ett meddelande

- ✓ Metoden Create skrivs i klassen GeekGuestbookController.
- ✓ En starkt typad vy, med namnet Create, skapas.

```

1 using GeekGuestbook.Models;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Web;
6 using System.Web.Mvc;
7
8 namespace GeekGuestbook.Controllers
9 {
10     public class GuestbookController : Controller
11     {
12         private GeekGuestbookEntities _entities;
13
14         // GET: Guestbook
15         public ActionResult Index()
16         {
17             var model = _entities.Messages;
18
19             return View(model);
20         }
21
22         // GET: Guestbook/Create
23         public ActionResult Create()
24         {
25             return View();
26         }
27     }
28 }

```

Add View

View name: Create

Template: Create

Model class: Message (GeekGuestbook.Models)

Data context class: GeekGuestbookEntities (GeekGuestbook.Models)

Options:

- ☐ Create as a partial view
- ☐ Reference script libraries
- ☐ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

```

1 @model GeekGuestbook.Models.Message
2
3 @{
4     Layout = null;
5 }
6
7 <!DOCTYPE html>
8
9 <html>
10 <head>
11     <meta name="viewport" content="width=device-width" />
12     <title>Create</title>
13 </head>
14 <body>
15     @using (Html.BeginForm())
16     {
17         @Html.AntiForgeryToken()
18
19         <div class="form-horizontal">
20             <h4>Message</h4>
21             <hr />
22             @Html.ValidationSummary(true, "", new { @class = "text-danger" })
23             <div class="form-group">
24                 @Html.LabelFor(model => model.Created, htmlAttributes: new { @class = "control-label" })
25                 <div class="col-md-10">
26                     @Html.EditorFor(model => model.Created, new { htmlAttributes = new { @class = "form-control" } })
27                     @Html.ValidationMessageFor(model => model.Created, "", new { @class = "text-danger" })
28                 </div>
29             </div>
30
31             <div class="form-group">
32                 @Html.LabelFor(model => model.Header, htmlAttributes: new { @class = "control-label" })
33                 <div class="col-md-10">
34                     @Html.EditorFor(model => model.Header, new { htmlAttributes = new { @class = "form-control" } })
35                     @Html.ValidationMessageFor(model => model.Header, "", new { @class = "text-danger" })
36                 </div>
37             </div>

```

Ta hand om postat formulärdata

- ✓ En Create-metod (till), med en parameter av typen Message, skrivs och märks med attributet `HttpPost` så metoden bara kan hantera förfrågningar av typen HTTP POST.
- ✓ Parametern `message` refererar till ett objekt innehållande formulärdata. Alla fält i formuläret har bundits automatiskt till motsvarande egenskap i objektet.
- ✓ Meddelandet sparas i databasen och klienten omdirigeras till Index.

```
// POST: Guestbook/Create
[HttpPost]
public ActionResult Create(Message message)
{
    message.Created = DateTime.Now;

    _entities.Messages.Add(message);
    _entities.SaveChanges();

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

```
// POST: Guestbook/Create
[HttpPost]
public ActionResult Create(Message message)
{
    message.Created = DateTime.Now;
    _entities.Messages.Add(message);
    _entities.SaveChanges();
    return RedirectToAction("Index");
}
```

Modifiera (hjälpigt) vyn för lista med meddelanden

- ✓ Meddelanden presenteras i flera article-element istället för i en tabell.

```
1 @model IEnumerable<GeekGuestbook.Models.Message>
2 using System.Globalization
3
4 @{
5     Layout = null;
6 }
7
8 <!DOCTYPE html>
9
10 <html>
11 <head>
12     <meta name="viewport" content="width=device-width" />
13     <title>Guestbook</title>
14 </head>
15 <body>
16     <h1>
17         Guestbook
18     </h1>
19     <p>
20         @Html.ActionLink("Have your say", "Create")
21     </p>
22     @foreach (var item in Model)
23     {
24         <article>
25             <header>
26                 <h2>
27                     @Html.DisplayFor(modelItem => item.Header)
28                 </h2>
29             <p>
30                 @String.Format(CultureInfo.InvariantCulture, "{0:F}", item.Created)
31                 @if (!String.IsNullOrEmpty(item.Name) &&
32                     !String.IsNullOrEmpty(item.Email))
33                 {
34                     @:by <a href="@item.Email">@item.Name</a>
35                 }
36                 else if (!String.IsNullOrEmpty(item.Name))
37                 {
38                     @:by <span>@item.Name</span>
39                 }
40             </p>
41         </header>
42         <div>
43             @Html.DisplayFor(modelItem => item.Body)
44         </div>
45     </article>
46 }
47 </body>
48 </html>
```



Modifiera (hjälpigt) vyn för nytt meddelande

- ✓ Mindre justeringar behövs, bl.a. så att ett TextArea-element används för innehållet.

```
1 @model GeekGuestbook.Models.Message
2
3 @{
4     Layout = null;
5 }
6
7 <!DOCTYPE html>
8
9 <html>
10 <head>
11 <meta name="viewport" content="width=device-width" />
12 <title>Create</title>
13 </head>
14 <body>
15     <@using (Html.BeginForm())
16     {
17         @Html.AntiForgeryToken()
18
19         <div class="form-horizontal">
20             <h1>Message</h1>
21             @Html.ValidationSummary(true, "", new { @class = "text-danger" })
22
23             <div class="form-group">
24                 @Html.LabelFor(model => model.Header, htmlAttributes: new { @class = "control-label col-md-2" })
25                 <div class="col-md-10">
26                     @Html.EditorFor(model => model.Header, new { htmlAttributes = new { @class = "form-control", maxlength = 100 } })
27                     @Html.ValidationMessageFor(model => model.Header, "", new { @class = "text-danger" })
28                 </div>
29             </div>
30
31             <div class="form-group">
32                 @Html.LabelFor(model => model.Body, htmlAttributes: new { @class = "control-label col-md-2" })
33                 <div class="col-md-10">
34                     @Html.TextAreaFor(model => model.Body, 5, 50, new { @class = "form-control" })
35                     @Html.ValidationMessageFor(model => model.Body, "", new { @class = "text-danger" })
36                 </div>
37             </div>
38
39             <div class="form-group">
40                 @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })
41                 <div class="col-md-10">
42                     @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control", maxlength = 50 } })
43                     @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
44                 </div>
45             </div>
46
47             <div class="form-group">
48                 @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
49                 <div class="col-md-10">
50                     @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control", maxlength = 100 } })
51                     @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
52                 </div>
53             </div>
54
55             <div class="form-group">
56                 <div class="col-md-offset-2 col-md-10">
57                     <input type="submit" value="Create" class="btn btn-default" />
58                 </div>
59             </div>
60         </div>
61     }
62
63     <div>
64         @Html.ActionLink("Back to List", "Index")
65     </div>
66 </body>
67 </html>
```

OBS! Kan göras på ett mycket smartare och generellare sätt!

HTML-attribut sätts med hjälp av en anonym typ.



Synpunkter och kritik

Synpunkter och kritik

- appUser har fullständiga rättigheter i databasen.
- Någon CSS används inte.
- Det finns inget ”*respository*” som kontrollern kan arbeta mot.
- Controllermetoderna anger inte vyns namn explicit (vilket försvårar testning).
- Avsaknaden av validering på server och klient är total.
- Någon ”*layout*” används inte.
- Felhantering saknas.
- Kommentarer saknas.
- Det finns inga tester (enhetstestprojekt skapades inte initialt).
- ...