



Linnéuniversitetet

Kalmar Vaxjö

Instruktion

Räkna med ASP.NET MVC 4

Introduktionsuppgift



Författare: Mats Looch

Kurs: ASP.NET MVC

Kurskod: 1DV409



Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET MVC (1DV409) vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Loock, förutom Linnéuniversitetets logotyp, symbol och kopparstick, 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, symbol och/eller kopparstick i din nya version!

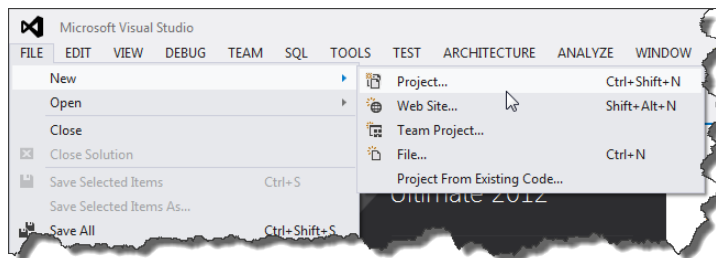
Vid all användning måste du ange källan: "Linnéuniversitetet – ASP.NET MVC" och en länk till <https://coursepress.lnu.se/kurs/aspnet-mvc> och till Creative Common-licensen här ovan.



Innehåll

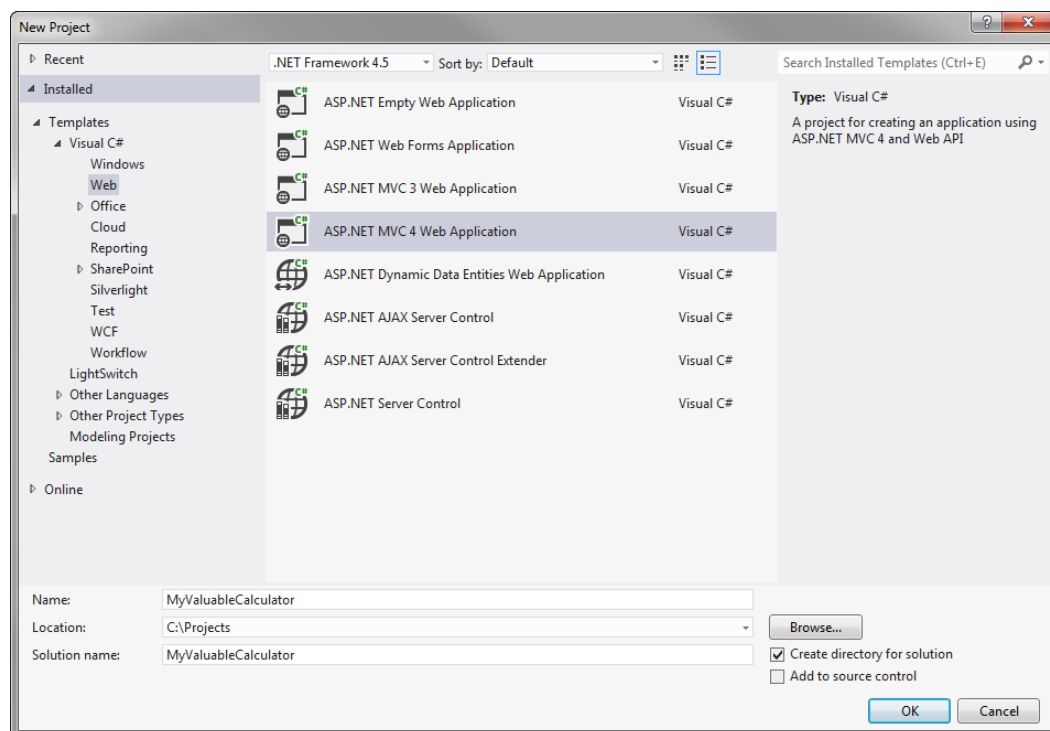
Du ska följa ”steg-för-steg”-instruktionen i denna introduktionsuppgift och skapa en ASP-NET MVC-applikation som ska addera två heltal en användare matar in i två textfält. Du kommer därefter att modifiera applikationen så att användaren kan välja vilket räkneseätt som ska användas.

1. Starta Visual Studio 2012.
2. Du ska skapa ett nytt projekt för en webbapplikation, välj därför **File ► New ► Project....**



Figur 1.

3. Dialogrutan **New Project** visas.

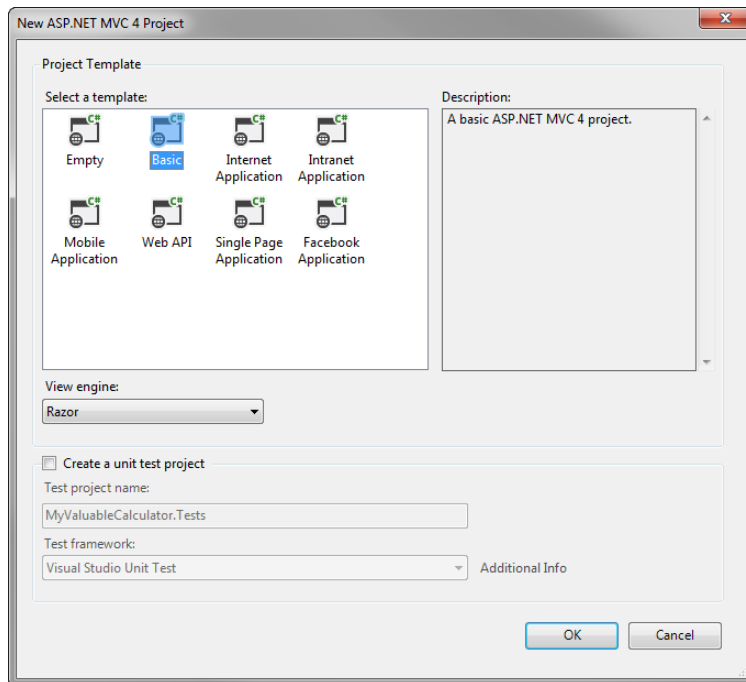


Figur 2.

- a) Under **Installed, Templates, Visual C#** markera **Web**.
- b) Kontrollera så att **.NET Framework 4.5** visas i den nedrullningsbara listrutan.
- c) Markera **ASP.NET MVC 4 Web Application**.
- d) Vid **Name** skriver du in projektets namn (MyValuableCalculator).
- e) Ange vid **Location** en lämplig katalog där projektet ska sparas (C:\Projects).
- f) Se till att kryssrutan vid **Create directory for solution** är markerad.

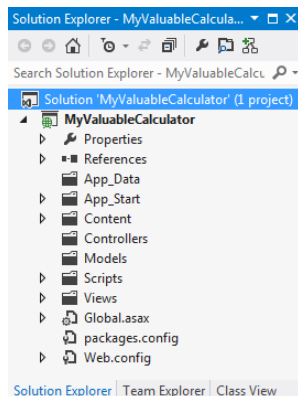


4. Dialogrutan **New ASP.NET MVC 4 Project** visas.



Figur 3.

- a) Under **Select a Template** markera **Basic**.
- b) Kontrollera så att **View engine** är satt till Razor.
5. Projektet Visual Studio skapar innehåller filer och kataloger.



Figur 4.

/App_Data

Används för att lagra applikationsdata, som t.ex. SQL-databaser, XML-filer.

/App_Start

Denna katalog är bara en konvention och den separerar en del kod från Global.asax.

/Content

Här placerar du CSS-filer, bilder och övrigt icke dynamiskt innehåll; Javascript placerar i en egen katalog.

/Controllers

Här placerar du Controller-klasser som hanterar URL-begäran som skickas till webbapplikationen.



/Models

Här placerar du klasser som representerar och manipulerar data och affärsobjekt.

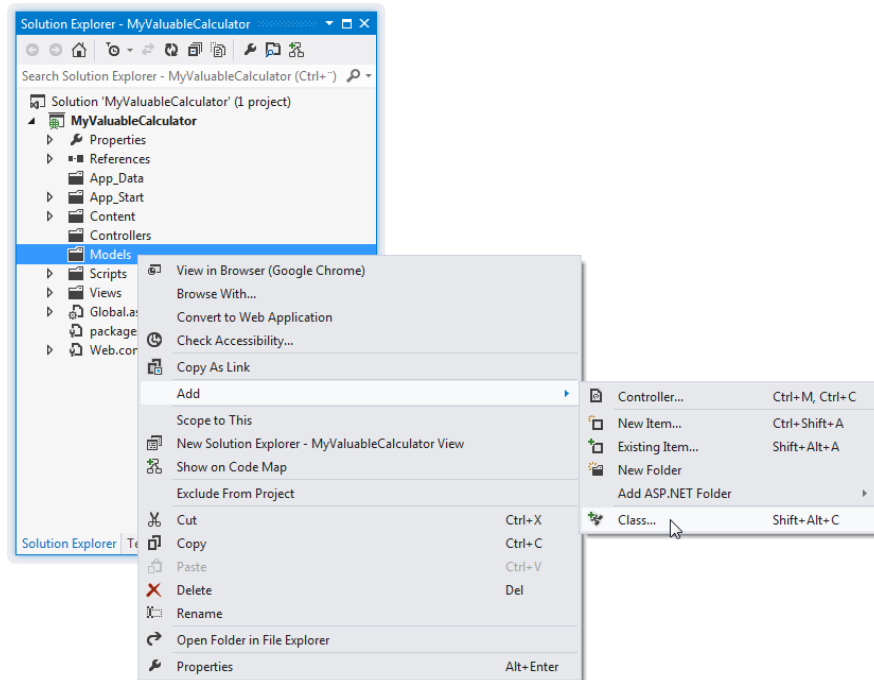
/Scripts

Här placerar du JavaScript-bibliotek och -skript (.js).

/Views

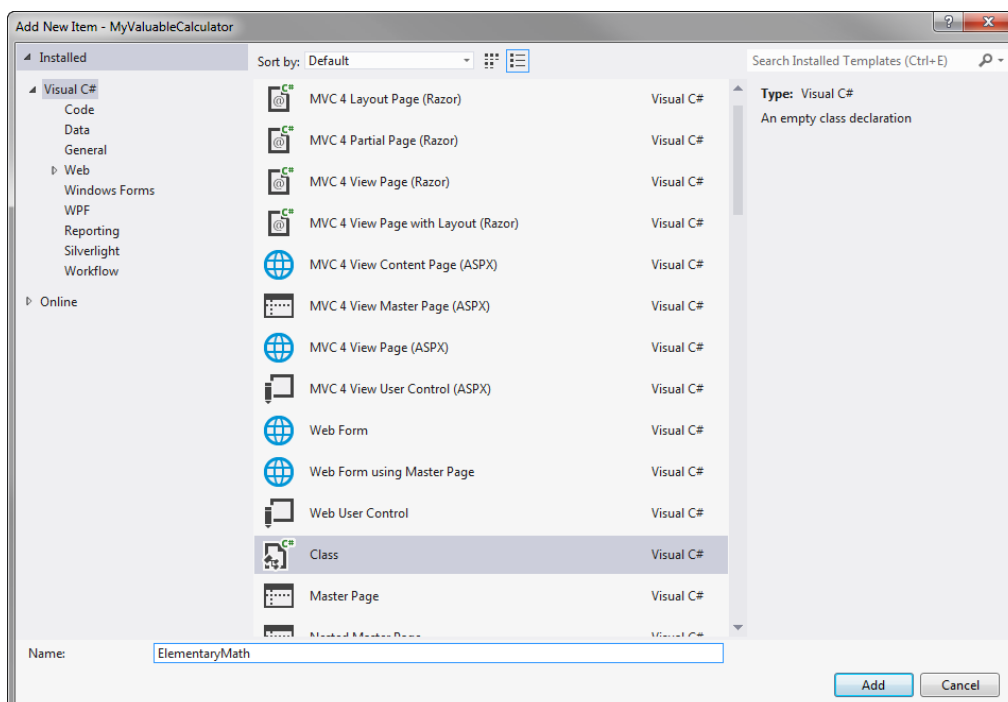
Här placerar du vy-filer som ansvar för rendering av t.ex. HTML.

6. Högerklicka på katalogen **Models** och välj **Add ► Class....**



Figur 5.

7. Dialogrutan **Add New Item** visas.



Figur 6.

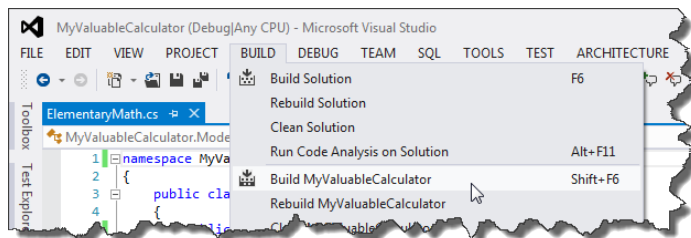


- a) Ange namnet på klassen (ElementaryMath) vid **Name**.
 - b) Klicka på **Add**.
8. Klassen `ElementaryMath` i namnområdet `MyValuableCalculator.Models`, ansvarar för att två heltal ska adderas. De två autoimplementerade egenskaperna `Op1` och `Op2` kommer att innehålla de värden användaren matar in i formulärets textfält. Metoden `Compute` utför additionen av `Op1` och `Op2` och summan lagras i "read-only"-egenskapen `Result`.

```
1 namespace MyValuableCalculator.Models
2 {
3     public class ElementaryMath
4     {
5         public int Op1 { get; set; }
6         public int Op2 { get; set; }
7         public int? Result { get; private set; }
8
9         public void Compute()
10        {
11            this.Result = this.Op1 + this.Op2;
12        }
13    }
14 }
```

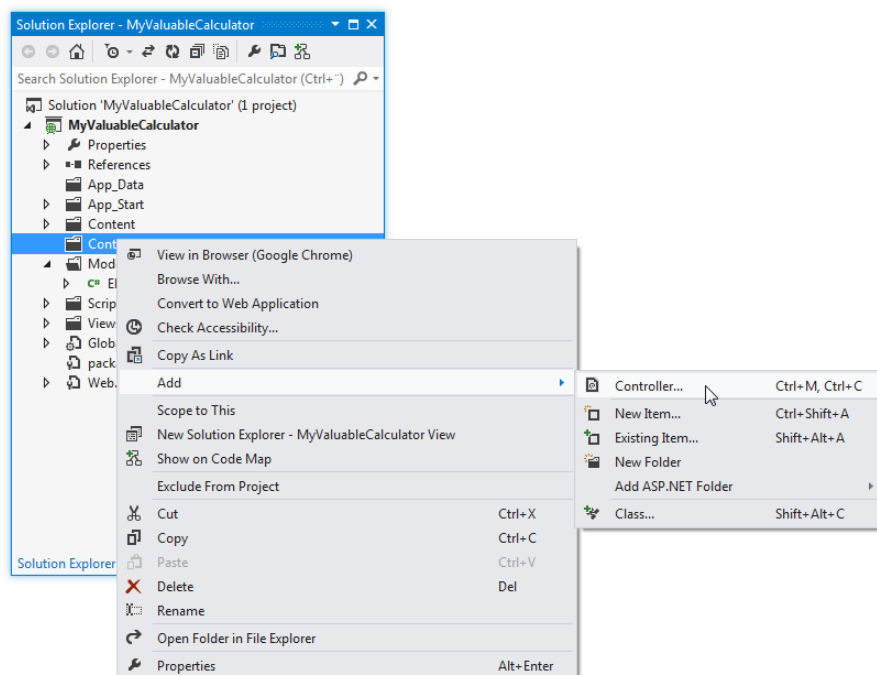
Figur 7.

9. För att Visual Studio ska bli fullt medveten om att en ny klass lagts till projektet bygger du projektet genom att trycka på **Skift+F6**. Du kan också välja **Build ► Build MyValuableCalculator**.



Figur 8.

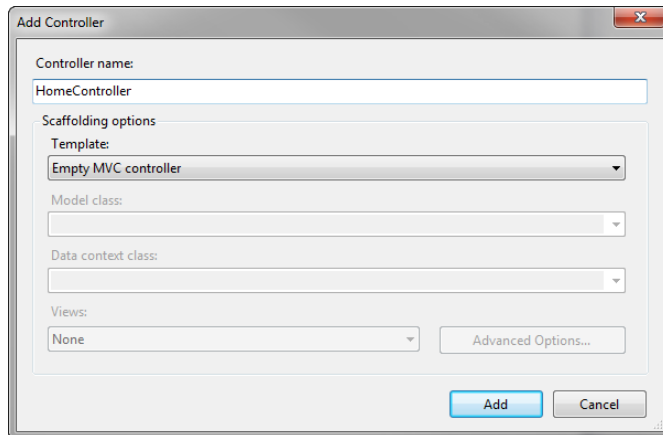
10. Högerklicka på katalogen **Controller** och välj **Add ► Controller....**



Figur 9.



11. Dialogrutan **Add Controller** visas.



Figur 10.

- Vid **Controller Name** skriver du in namnet, HomeController, på kontrollern. **OBS!** Det är mycket viktigt att namnet på kontrollern avslutas med Controller.
- Klicka på **Add**.

12. Kontrollern HomeController skapas.

```
6
7 namespace MyValuableCalculator.Controllers
8 {
9     public class HomeController : Controller
10    {
11        //
12        // GET: /Home/
13
14        public ActionResult Index()
15        {
16            return View();
17        }
18    }
19 }
20
21
```

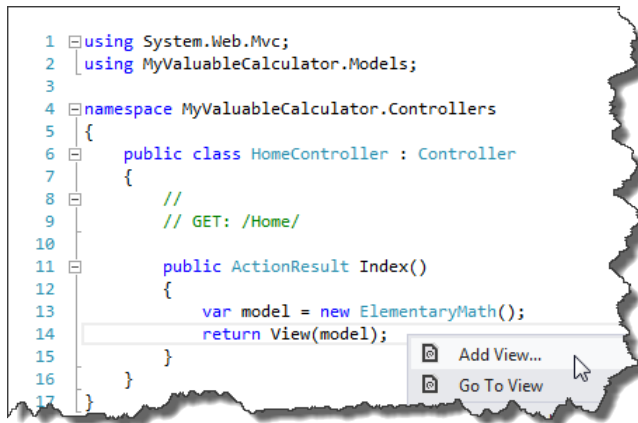
Figur 11.

13. Komplettera metoden Index för att skapa en modell, en instans av klassen ElementaryMath, som skickas till vyn.

```
1 using System.Web.Mvc;
2 using MyValuableCalculator.Models;
3
4 namespace MyValuableCalculator.Controllers
5 {
6     public class HomeController : Controller
7     {
8         //
9         // GET: /Home/
10
11        public ActionResult Index()
12        {
13            var model = new ElementaryMath();
14            return View(model);
15        }
16    }
17 }
```

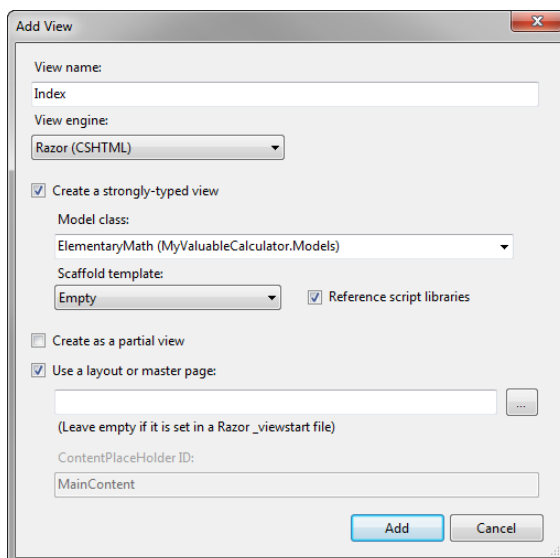
Figur 12.

14. Högerklicka någonstans i metoden Index och välj **Add View....**



Figur 13.

15. Dialogrutan **Add View** visas.

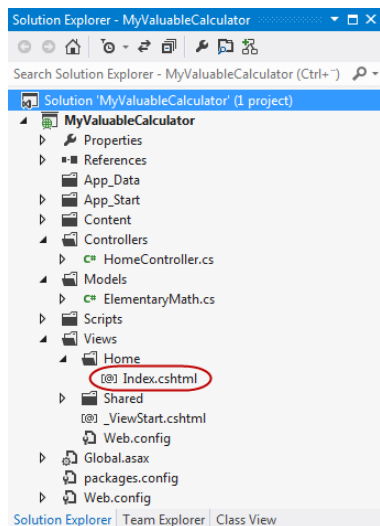


Figur 14.

- Det är viktigt att vyn har samma namn som metoden i kontrollern, d.v.s. Index.
- Kontrollera så att **Razor** är valt under **View engine**.
- Markera **Create a strongly-typed view**.
- Vid **Model class**: välj klassen **ElementaryMath (MyValuableCalculator.Models)**.
- Under **Scaffold template** ska **Empty** vara valt.
- Reference script libraries** ska vara markerad.
- Klicka på **Add**.

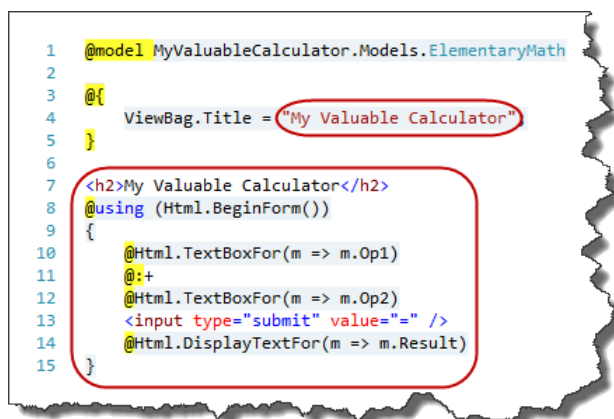
16. Filen `Index.cshtml` skapas i katalogen `\Views\Home`.

Vyer som tillhör kontrollern `HomeController` måste placeras i katalogen `\Views\Home`. Det är vanligt att vyernas namn är detsamma som kontrollmetoden som använder dem, men det behöver inte vara så.



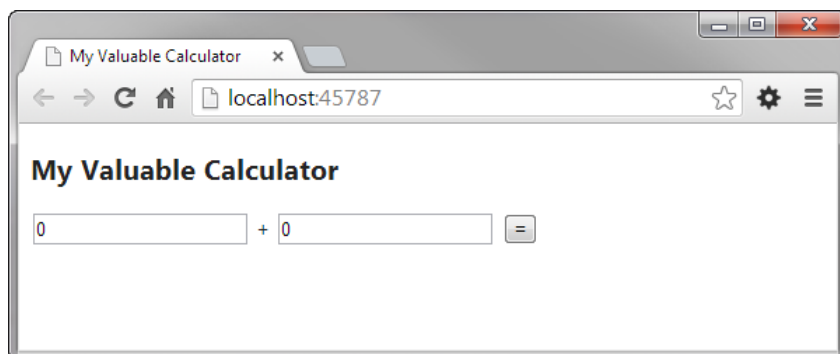
Figur 15.

17. Med Html-hjälpmetoder skapas ett formulär. Den första raden talar om att vyn är starkt typad och dess modell är av typen `ElementaryMath`.



Figur 16. På raderna 10, 12 och 14 refererar `m` till objektet som utgör modellen.

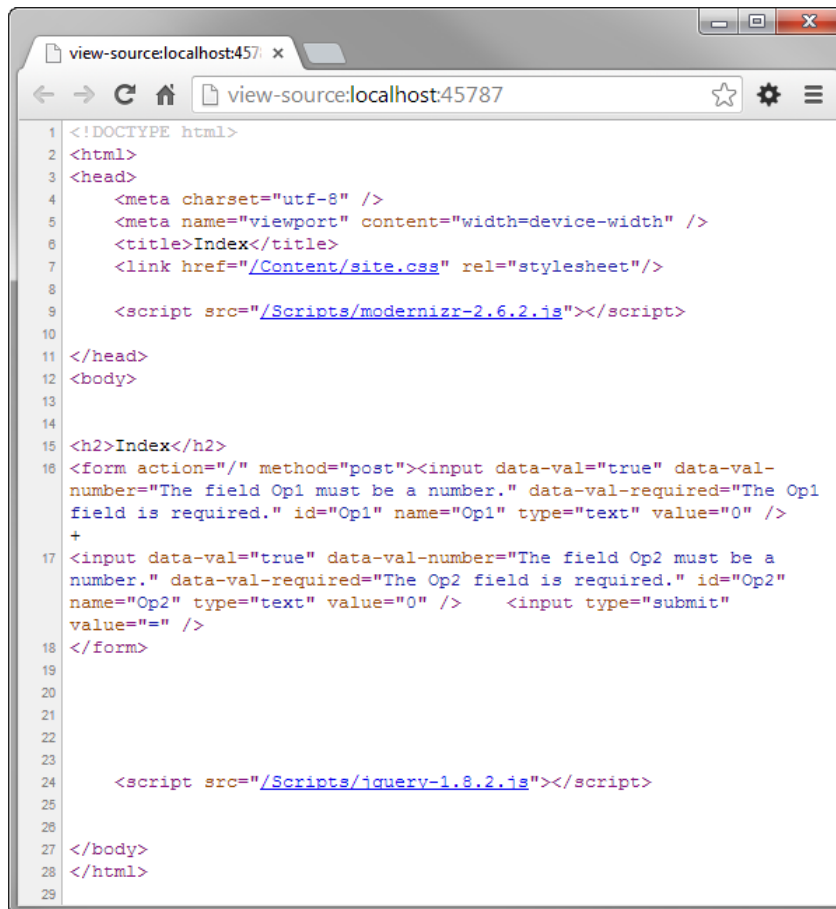
18. Webbapplikationen är nu färdig för att provköras för första gången. Enklast provkör du den genom att trycka på `Ctrl+F5`.



Figur 17.

Den renderade HTML-koden innehåller i princip inget du inte skulle skrivit för hand. Textfälten namnges automatiskt med namnet egenskaperna, `Op1` respektive `Op2`, har i klassen

ElementaryMath. Någon HTML för egenskapen Result renderas inte eftersom den har värdet null.



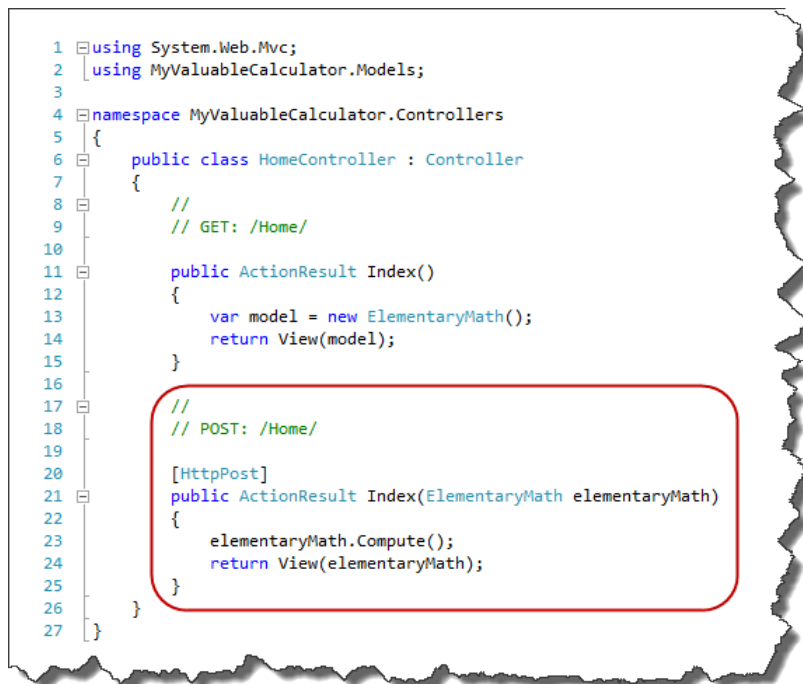
```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width" />
6   <title>Index</title>
7   <link href="/Content/site.css" rel="stylesheet"/>
8
9   <script src="/Scripts/modernizr-2.6.2.js"></script>
10
11 </head>
12 <body>
13
14
15 <h2>Index</h2>
16 <form action="/" method="post"><input data-val="true" data-val-
17   number="The field Op1 must be a number." data-val-required="The Op1
18   field is required." id="Op1" name="Op1" type="text" value="0" />
19 +
20 <input data-val="true" data-val-number="The field Op2 must be a
21   number." data-val-required="The Op2 field is required." id="Op2"
22   name="Op2" type="text" value="0" /> <input type="submit"
23   value=" " />
24 </form>
25
26
27 <script src="/Scripts/jquery-1.8.2.js"></script>
28
29 </body>
30 </html>

```

Figur 18.

19. För att kunna behandla formulärdata behöver klassen HomeController kompletteras.



```

1 using System.Web.Mvc;
2 using MyValuableCalculator.Models;
3
4 namespace MyValuableCalculator.Controllers
5 {
6     public class HomeController : Controller
7     {
8         //
9         // GET: /Home/
10
11         public ActionResult Index()
12         {
13             var model = new ElementaryMath();
14             return View(model);
15         }
16
17         //
18         // POST: /Home/
19
20         [HttpPost]
21         public ActionResult Index(ElementaryMath elementaryMath)
22         {
23             elementaryMath.Compute();
24             return View(elementaryMath);
25         }
26     }
27 }

```

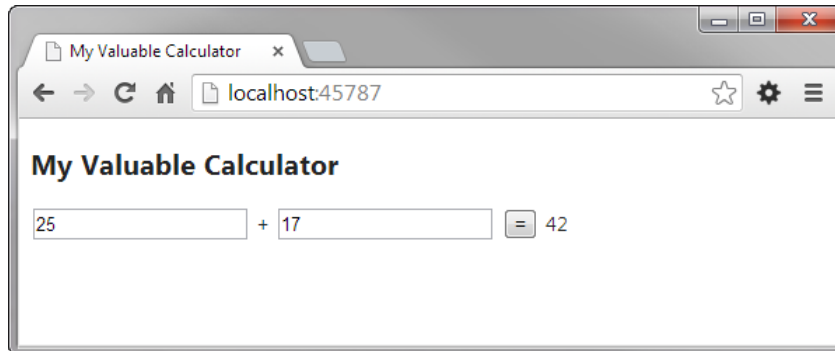
Figur 19.

Metoden med attributet `HttpPost` tar hand om URL-begäran av typen POST. Formulärdata



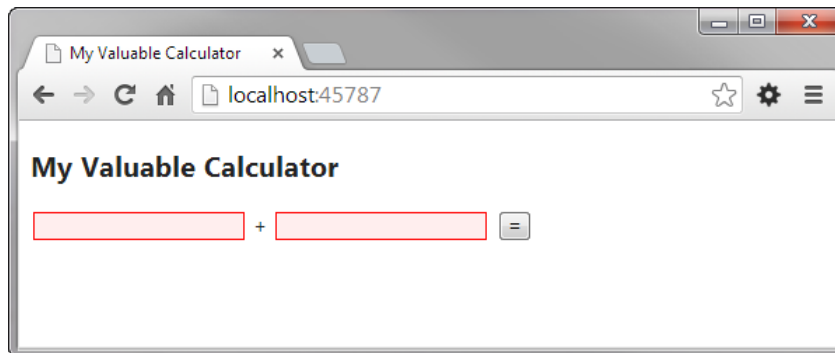
används för att initiera ElementaryMath-objektet parametern elementaryMath refererar till. Egenskaperna Op1 och Op2 kommer att innehålla de värden användaren matat in i respektive textfält i formuläret.

20. Provkör webbapplikationen med Ctrl+F5. Skriv in värden 25 och 17. Klicka på kommandoknappen och konstatera att den beräkna summan helt korrekt blir 42.



Figur 20.

Radera innehållet i textfälten och klicka på kommandoknappen. Konstatera att textfälten nu formateras annorlunda och att någon summa inte presenteras. Varför? Någon validering är i egentlig mening inte implementerad!



Figur 21.

21. För att implementera validering måste klassen ElementaryMath kompletteras så att validering sker med hjälp av "data annotations".

```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace MyValuableCalculator.Models
4 {
5     public class ElementaryMath
6     {
7         [Required]
8         public int Op1 { get; set; }
9
10        [Required]
11        public int Op2 { get; set; }
12
13        public int? Result { get; private set; }
14
15        public void Compute()
16        {
17            this.Result = this.Op1 + this.Op2;
18        }
19    }
20 }
```

Figur 22.

22. I kontrollermetoden måste modellens status undersökas. Om objektet uppfyller de krav valideringen ställer ska en beräkning ske och objektet, innehållande resultatet skickas vidare till vyn. Klarar inte objektet valideringen skickas objektet utan beräkning gjord men innehållande felmeddelanden tillbaka till vyn.

```
9 // GET: /Home/
10
11 public ActionResult Index()
12 {
13     var model = new ElementaryMath();
14     return View(model);
15 }
16
17 //
18 // POST: /Home/
19
20 [HttpPost]
21 public ActionResult Index(ElementaryMath elementaryMath)
22 {
23     if (ModelState.IsValid)
24     {
25         elementaryMath.Compute();
26     }
27     return View(elementaryMath);
28 }
29 }
30 }
```

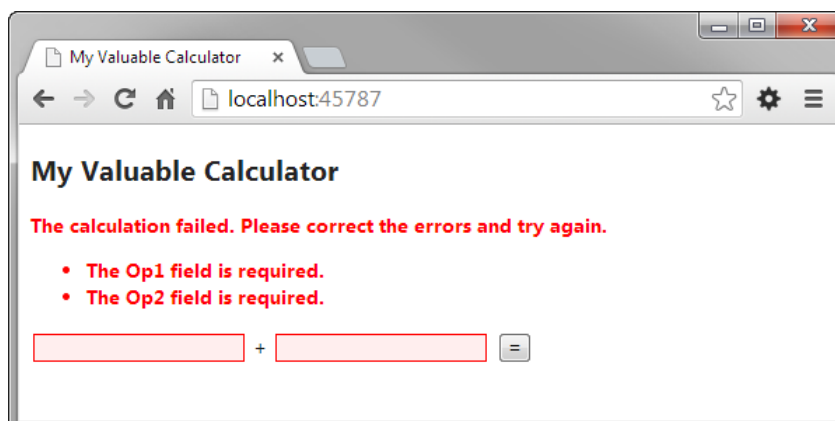
Figur 23.

23. Även vyn Home/Index.aspx måste modifieras så valideringsmeddelanden visas.

```
1 @model MyValuableCalculator.Models.ElementaryMath
2
3 @{
4     ViewBag.Title = "My Valuable Calculator";
5 }
6
7 <h2>My Valuable Calculator</h2>
8 @using (Html.BeginForm())
9 {
10     @Html.ValidationSummary("The calculation failed. Please correct the errors and try again.")
11     @Html.TextBoxFor(m => m.Op1)
12     @: +
13     @Html.TextBoxFor(m => m.Op2)
14     <input type="submit" value="=" />
15     @Html.DisplayTextFor(m => m.Result)
16 }
```

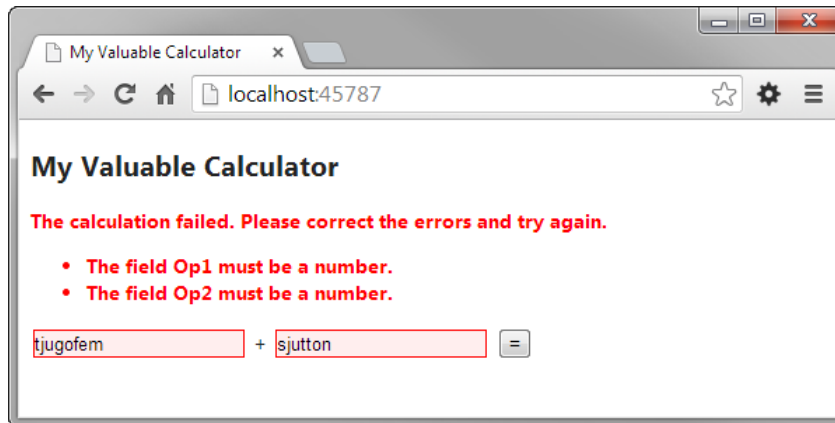
Figur 24.

24. Nu visas felmeddelande om användaren inte matar in något i textfälten och klickar på kommandoknappen.



Figur 25.

Felmeddelanden visas även då användaren matar in något som inte kan tolkas som ett heltal.



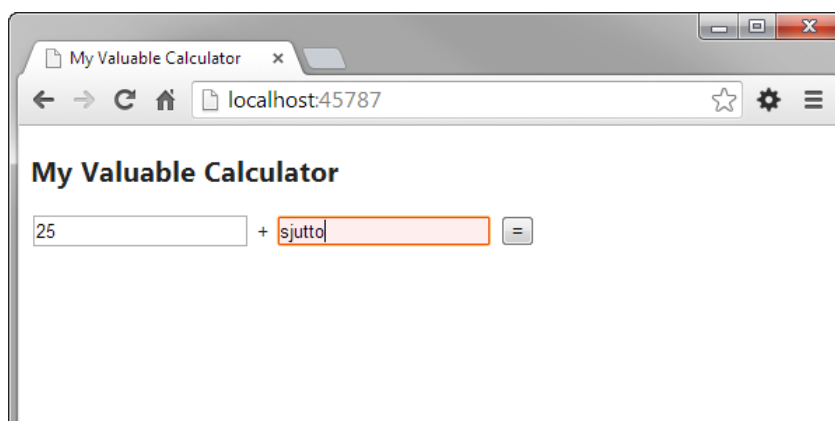
Figur 26.

25. Nu sker all validering på servern. För att valideringen även ska ske på klienten måste `Shared/_Layout.cshtml` kompletteras.



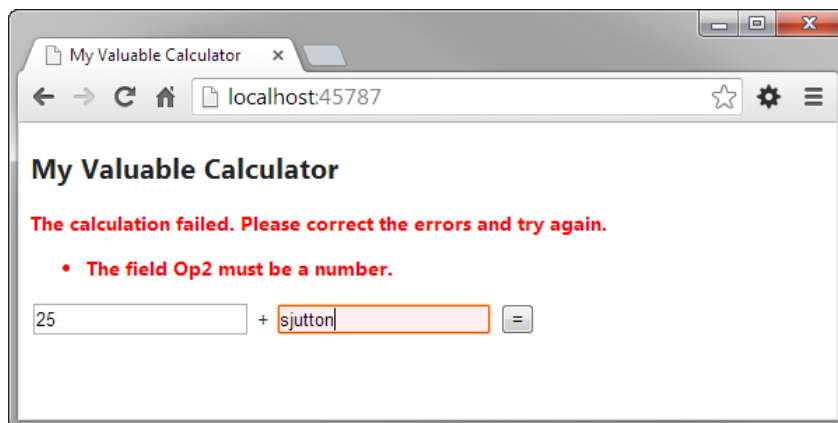
Figur 27.

26. Nu sker valideringen omedelbart när användaren skriver i textfälten.



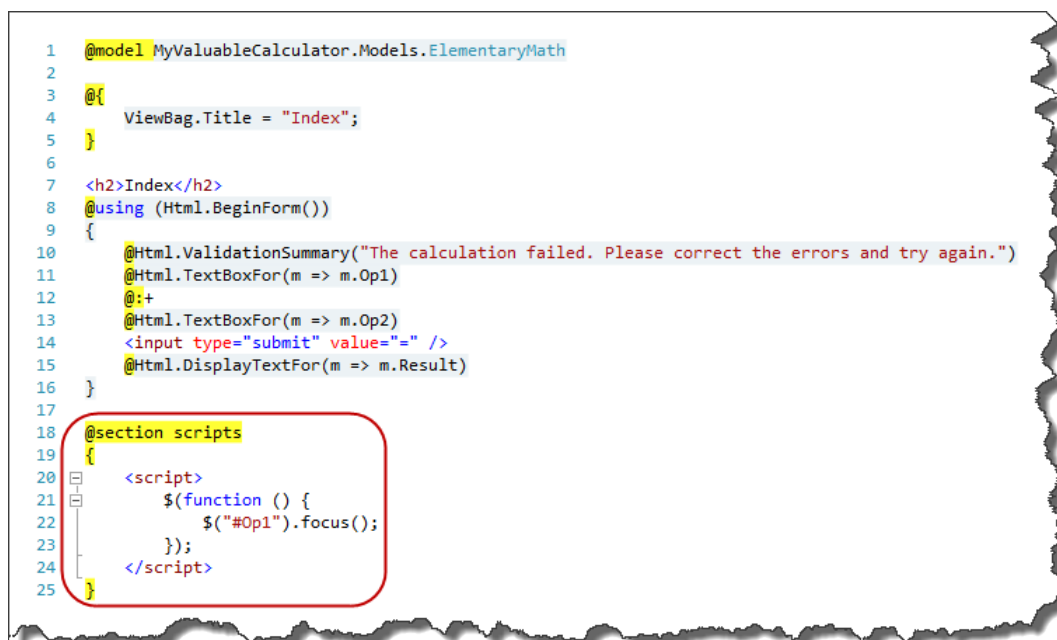
Figur 28.

Klickar användaren på kommandoknappen postas inte formuläret men felmeddelanden visas ändå.



Figur 29.

27. Som service till användaren av webbapplikationen skulle det vara trevligt om det första textfältet har fokus när sidan laddat klart. Det kan ordnas enkelt med jQuery. Vyn Home/Index.aspx behöver modifieras.



Figur 30.

(Fungerar skriptet verkligen? Det gör det eftersom en funktion som skickas som ett argument till jQuery-konstruktorn binds automatiskt till händelsen ready, varför `$(document).ready()` inte behöver anges explicit.)

28. Nu är webbapplikationen nästan klar. Punkterna som följer visar hur en nedrullningsbar listruta kan användas för att användaren ska kunna välja räknesätt.

29. Modellen, d.v.s. klassen `ElementaryMath`, måste modifieras så att den kan hantera de fyra räknesätten.

```

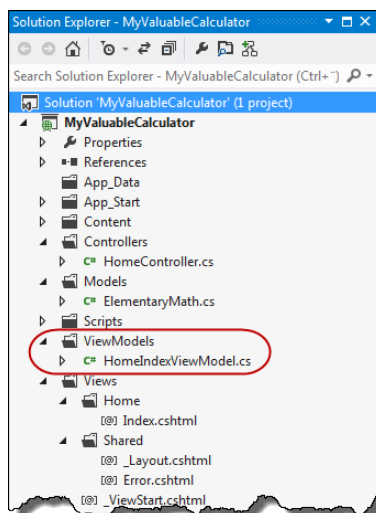
1  using System.ComponentModel.DataAnnotations;
2
3  namespace MyValuableCalculator.Models
4  {
5      public enum ArithmeticOperation { Addition, Subtraction, Multiplication, Division }
6
7      public class ElementaryMath
8      {
9          public ArithmeticOperation ArithmeticOperation { get; set; }
10
11          [Required]
12          public int Op1 { get; set; }
13
14          [Required]
15          public int Op2 { get; set; }
16
17          public int? Result { get; private set; }
18
19          public void Compute()
20          {
21              switch (this.ArithmeticOperation)
22              {
23                  case ArithmeticOperation.Addition:
24                      this.Result = this.Op1 + this.Op2;
25                      break;
26
27                  case ArithmeticOperation.Subtraction:
28                      this.Result = this.Op1 - this.Op2;
29                      break;
30
31                  case ArithmeticOperation.Multiplication:
32                      this.Result = this.Op1 * this.Op2;
33                      break;
34
35                  case ArithmeticOperation.Division:
36                      this.Result = this.Op1 / this.Op2;
37                      break;
38              }
39          }
40      }
41  }

```

Figur 31.

30. Skapa en ny katalog, **ViewModels**, under projektroten. I denna katalog placerar du filer med klasser, innehållande vspecifikt data som inte har med modellen att göra, controllrar kan skapa instanser av för att skicka till vyer.

Lägg till en ny klass med namnet `HomeIndexViewModel` i katalogen **ViewModels**.



Figur 32.



31. Klassen `ElementaryMathViewModel` innehåller två publika egenskaper. Egenskapen `ElementaryMath` refererar till objektet som utgör själva modellen. "Read-only"-egenskapen `ArithmeticOperations` returnerar en lista med de fyra räknesätten som den nedrullningsbara listrutan i vyn ska visa.

```
1 using System.Collections.Generic;
2 using System.Web.Mvc;
3 using MyValuableCalculator.Models;
4
5 namespace MyValuableCalculator.ViewModels
6 {
7     public class HomeIndexViewModel
8     {
9         public ElementaryMath ElementaryMath { get; set; }
10
11         public SelectList ArithmeticOperations
12         {
13             get
14             {
15                 var items = new Dictionary<ArithmeticOperation, string>
16                 {
17                     { ArithmeticOperation.Addition, "+" },
18                     { ArithmeticOperation.Subtraction, "-" },
19                     { ArithmeticOperation.Multiplication, "*" },
20                     { ArithmeticOperation.Division, "/" },
21                 };
22                 return new SelectList(items, "Key", "Value");
23             }
24         }
25     }
26 }
27 }
```

Figur 33.

32. Vyn `Home/Index.aspx` måste modifieras så den använder vymodellklassen `HomeIndexViewModel` istället för modellklassen `ElementaryMath`. Dessutom ska +-tecknet ersättas med en nedrullningsbar listruta.

```
1 @model MyValuableCalculator.ViewModels.HomeIndexViewModel
2 @{
3     ViewBag.Title = "Index";
4 }
5 <h2>
6     Index</h2>
7 @using (Html.BeginForm())
8 {
9     @Html.ValidationSummary("The calculation failed. Please correct the errors and try again.")
10    @Html.EditorFor(m => m.ElementaryMath.Op1)
11    @Html.DropDownListFor(m => m.ElementaryMath.ArithmeticOperation, Model.ArithmeticOperations)
12    @Html.EditorFor(m => m.ElementaryMath.Op2)
13    <input type="submit" value="=" />
14    @Html.DisplayTextFor(m => m.ElementaryMath.Result)
15 }
16 <script type="text/javascript">
17     $(function () {
18         $("#ElementaryMath_Op1").focus();
19     });
20 </script>
```

Figur 34.

Lägg märke till att `Op1`, `Op2` och `Result` nu måste förgås av `ElementaryMath`, som ju egenskapen i vymodellklassen heter som refererar till modellobjektet av typen `ElementaryMath`.

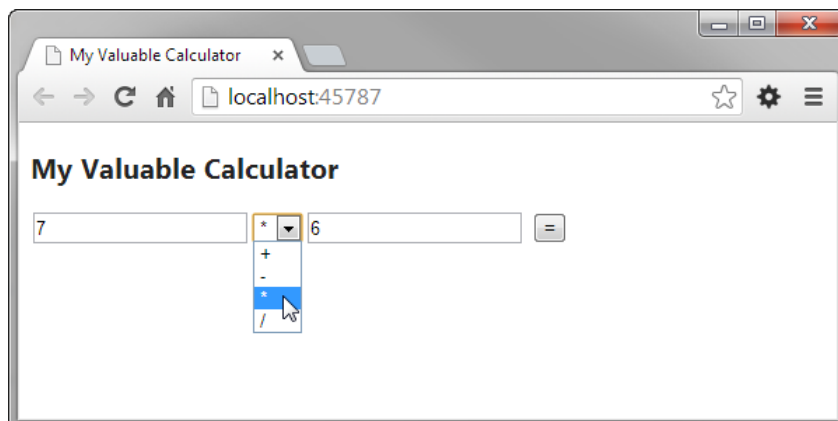
33. Även metoderna i klassen HomeController måste modifieras att använda klassen HomeIndexViewModel.

```
1 using System;
2 using System.Web.Mvc;
3 using MyValuableCalculator.Models;
4 using MyValuableCalculator.ViewModels;
5
6 namespace MyValuableCalculator.Controllers
7 {
8     public class HomeController : Controller
9     {
10         //
11         // GET: /Home/
12
13         public ActionResult Index()
14         {
15             var model = new HomeIndexViewModel
16             {
17                 ElementaryMath = new ElementaryMath()
18             };
19             return View(model);
20         }
21
22         //
23         // POST: /Home/
24
25         [HttpPost]
26         public ActionResult Index(ElementaryMath elementaryMath)
27         {
28             if (ModelState.IsValid)
29             {
30                 try
31                 {
32                     elementaryMath.Compute();
33                 }
34                 catch (Exception ex)
35                 {
36                     ModelState.AddModelError(String.Empty, ex.Message);
37                 }
38             }
39
40             var model = new HomeIndexViewModel
41             {
42                 ElementaryMath = elementaryMath
43             };
44             return View(model);
45         }
46     }
47 }
48
49 }
```

Figur 35.

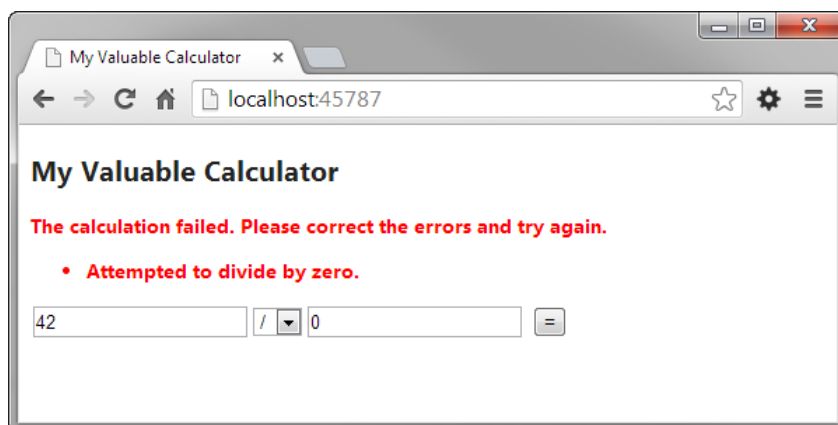
Det enda som behöver ändras är att objektet som skickas till vyn ska vara av typen HomeIndexViewModel. ASP.NET MVC kan fortfarande binda formulärdata till objektet av typen ElementaryMath. Detta fungerar eftersom parametern (elementaryMath) har samma namn som egenskapen har i vymodellen (ElementaryMath).

34. Nu kan användaren välja om de två talen ska summeras, subtraheras, multipliceras eller divideras.



Figur 36.

Valideringen fungerar fortfarande trots förändringarna som gjorts. Det är bara att prova...



Figur 37.

35. Nu är webbapplikationen klar. Förhoppningsvis har du nu fått en introduktion till ASP.NET MVC och förstår grundläggande koncept, även om det är mycket mer att lära.