



Linnéuniversitetet

Kalmar Vaxjö

Laborationsanvisning

Konvertera temperaturer

Steg 1, laborationsuppgift 3



Författare: Mats Looch

Kurs: ASP.NET Web Forms

Kurskod: 1DV406

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET Web Forms (1DV406) vid Linnéuniversitetet.

Du får använda detta verk så här:

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



Creative Commons Erkännande 4.0 Internationell licens.

<http://creativecommons.org/licenses/by/4.0>

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!

Innehåll

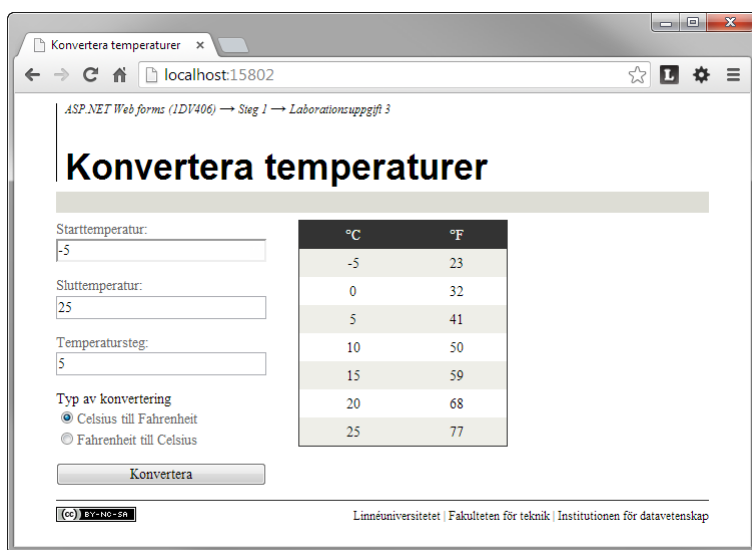
Uppgift	5
Inledning	5
Presentationslogiklagret	5
Textfälten och validering	6
Alternativknapparna	6
Kommandoknappen	6
Temperaturtabellen	7
Stilmallar	7
Affärslogiklagret	8
Hantering av fel på servern	8
Mål	8
Tips	9

Uppgift

Inledning

Du ska skriva en webbapplikation där användaren ska kunna skapa en tabell med temperaturer konverterade mellan grader Celsius och grader Fahrenheit eller mellan grader Fahrenheit och grader Celsius.

För att skapa en tabell med konverterade temperaturer matar användaren in i ett formulär vilken temperatur tabellen ska börja från och temperaturen tabellen ska sluta med. Användaren måste även ange i formuläret hur stor temperaturökningen ska vara mellan varje temperatur fram till och med sluttemperaturen samt om konverteringen ska ske från grader Celsius till grader Fahrenheit eller från grader Fahrenheit till grader Celsius.



°C	°F
-5	23
0	32
5	41
10	50
15	59
20	68
25	77

Figur 1. Temperaturtabell med konvertering från grader Celsius till grader Fahrenheit.

Figur 1 innehåller en tabell där tabellen börjar med -5 °C, slutar på 25 °C, temperaturen ökar med 5 °C mellan varje temperatur och konvertering sker från grader Celsius till grader Fahrenheit.

Applikationen ska delas upp i ett presentationslogiklager och ett affärslogiklager, d.v.s. kod som har att göra konvertering av temperaturer placeras i en separat klass som ensam får utgöra affärslogiklagret. Ett webbformulär, aspx-fil med tillhörande "code behind"-fil, utgör presentationslogiklagret.

Allt data i formuläret ska valideras innan det behandlas av webbservern. Validering ska ske på klienten för att användaren ska få en snabb återkoppling om något är fel. Validering ska också ske på servern så att felaktiga data inte orsakar att applikationen eventuellt kraschar.

Presentationslogiklagret

Du kan fritt utforma formuläret som användaren ska mata in uppgifterna i som sedan skickas till webbservern. Allt data som skickas till servern ska vara validerat.

HTML-elementen i figur 1 har renderats ut med hjälp av serverkontroller som TextBox, Label, RadioButton, Button och Table.

Genom att använda TextBox och de valideringskontroller som finns är det enkelt att kontrollera om användaren matat in data på rätt sätt. Valideringskontrollerna har du glädje av både på klienten och på server. På klienten sköts valideringen automatiskt med JavaScript.

Valideringen ska fungera även om användaren inte tillåter att JavaScript körs varför du även alltid måste validera på servern. På servern behöver du bara använda dig av egenskapen IsValid i klassen Page för att undersöka om datat som postats är giltigt eller inte.

Textfälten och validering

Till textfälten använder du lämpligen tre TextBox-kontroller, som renderas ut som input-element med attributet type satt till text.

Alla textfälten måste vara ifyllda på ett korrekt sätt.

- De får inte vara tomma.
- Texten i dem måste kunna tolkas som heltal.
- Sluttemperaturen måste vara högre än starttemperaturen.
- Temperatursteget måste ligga mellan 1 och 100.

Med hjälp av kontrollen `RequiredFieldValidator` kan du säkerställa att ett textfält inte är tomt.

Kontrollen `CompareValidator` använder till att undersöka om ett textfälts innehåll kan tolkas som ett heltal eller inte. Egenskaperna `Type` och `Operator` måste du sätta till lämpliga värden.

Med kontrollen `RangeValidator` kan du enkelt säkerställa att ett värde användaren matar in i ett textfält verkligen befinner sig i ett intervall du bestämt.

Felmeddelanden måste presenteras för användaren genom användning av kontrollen `ValidationSummary`. Du är fri att välja mellan två olika alternativ på vilket sätt felmeddelanden ska visas om användaren inte fyllt i ett eller flera textfält på ett giltigt sätt.

- Kontrollen `ValidationSummary` kan presentera alla meddelanden på ett och samma ställe (se figur 2).
- Kontrollen `ValidationSummary` kan presentera alla meddelanden i en meddelanderuta (se figur 3).

Figur 2. Inget av textfälten ifyllda. En summering av meddelanden visas med hjälp av kontrollen `ValidationSummary`.

Figur 3. Sluttemperaturen är lägre än starttemperaturen och temperatursteget är inte ett heltal i det slutna intervallet från 1 till 100. En meddelanderuta visas med hjälp av kontrollen `ValidationSummary`.

Alternativknapparna

Använd gärna två `RadioButton`-kontroller så användaren enkelt kan välja mellan de två olika sätten att konvertera. Egenskapen `GroupName` använder du för att gruppera kontrollerna tillsammans vilket medför att bara ett av alternativen kan vara vald. Egenskapen `Checked` ger information om ett alternativ är valt eller inte. Vilket bör vara intressant att undersöka i en hanterarmetod i "code behind"-filen. Glöm inte att se till att en av alternativknapparna är vald vid en GET av sidan.

Kommandoknappen

Kommandoknappen ska se till att formulärdata postas till servern. Själva klicket på knappen tar du hand om i "code behind"-filen genom att koppla en hanterarmetod till händelsen `Click`. Observera att

det är av största vikt att det första du gör i hanterarmetoden är att kontrollera om formulärdatat är giltigt – du gör det med hjälp av egenskapen `IsValid` – innan du hämtar datat från de olika kontrollerna. Om `IsValid` är `true` kan du riskfritt tolka det egenskapen `Text` innehåller för de olika kontrollerna som heltal. Då `IsValid` är `false` ska du inte göra någonting i hanterarmetoden.

Temperaturtabellen

Du väljer helt själv hur du vill presentera temperaturtabellen. I figur 1 visas tabellen till höger om textfälten, men du kan välja var du vill placera den på sidan. Det ska inte renderas någon tabell vid en GET av sidan.

Väljer användaren att konvertera temperaturer från grader Celsius till grader Fahrenheit ska kolumnen till vänster innehålla grader Celsius och kolumnen till höger innehålla de konverterade värdena i grader Fahrenheit. Väljer användaren att konvertera från grader Fahrenheit till grader Celsius ska kolumnen med grader Fahrenheit vara till vänster och kolumnen för grader Celsius vara till höger.

En tabell kan skapas på flera olika sätt. Du kan t.ex. använda dig av kontrollen `Table` som renderas ut som ett `table`-element med `tr`- och `td`-element.

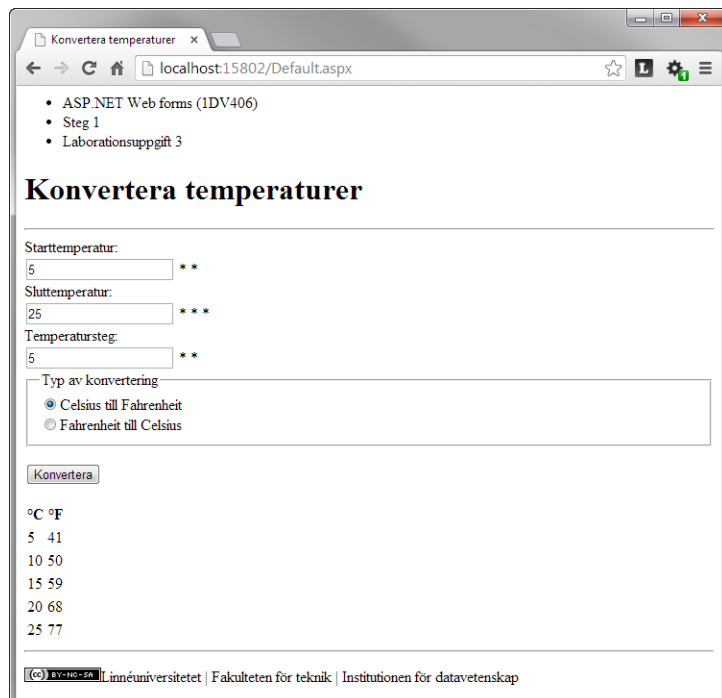
Med fördel kan du placera en `Table`-kontroll på `aspx`-sidan och sätta egenskapen `Visible` till `false`. I ”code behind”-filen skapar du `tr`- och `td`-element med hjälp av klasserna `TableRow` och `TableCell` som läggs till `Table`-kontrollen. Först då `Table`-kontrollen innehåller temperaturer är det mening med att visa den genom att i ”code behind”-filen sätta egenskapen `Visible` till `true`.

Med hjälp av egenskapen `Row` i `Table`-klassen och metoden `Add` lägger du till nya rader till ett `Table`-kontrollen. Egenskapen `Cells` i klassen `TableRow` och metoden `AddRange` använder du för att enkelt lägga till en array med referenser till `TableCell`-objekt.

Vill du inte använda dig av `Table`-kontrollen kan en `Placeholder`-kontroll som du fyller med en `LiteralControl`-kontroll med lämplig HTML vara ett alternativ. Med detta sätt får du en fullständig kontroll över det som renderas ut. Nackdelen är att det i regel tar mycket längre tid än att använda färdiga kontroller som finns och att det är lättare att göra fel¹.

Stilmallar

Självklart ska du använda dig av en, eller flera, stilmallar som ger utformningen av dokumentet. Lika självklart är det att dokumentet ska kunna användas även då en klient stängt av stilmallar.



Konvertera temperaturer

Starttemperatur: 5 **

Sluttemperatur: 25 ***

Temperatursteg: 5 **

Typ av konvertering

☒ Celsius till Fahrenheit

☐ Fahrenheit till Celsius

Konvertera

°C	°F
5	41
10	50
15	59
20	68
25	77

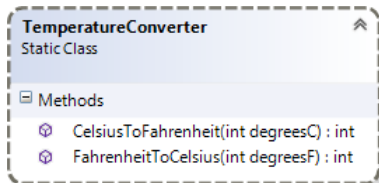
(c) BY-NC-SA Linneuniversitetet | Fakulteten för teknik | Institutionen för datavetenskap

Figur 4. Sida utan stilmall.

¹ Ta gärna tillfället i akt och undersök båda alternativen för att skaffa dig en egen uppfattning.

Affärslogiklagret

Du ska placera koden som konverterar temperaturer i affärslogiklagret så att beräkningarna separeras från hanteringen av användargränssnittet. Du ska därför placera allt som har med temperaturkonvertering att göra i en separat klass som placeras i foldern `Model`, som du själv måste skapa. Implementera den statiska klassen `TemperatureConverter` enligt klassdiagrammet i figur 5.



Figur 5. Klassen `TemperatureConverter`.

Klassen innehåller två statiska metoder som ska användas för att konvertera temperaturer från grader Celsius till grader Fahrenheit, och tvärtom. Formeln

$$[^{\circ}\text{F}] = [^{\circ}\text{C}] \cdot 1,8 + 32$$

använder du för att konvertera från grader Celsius till grader Fahrenheit.

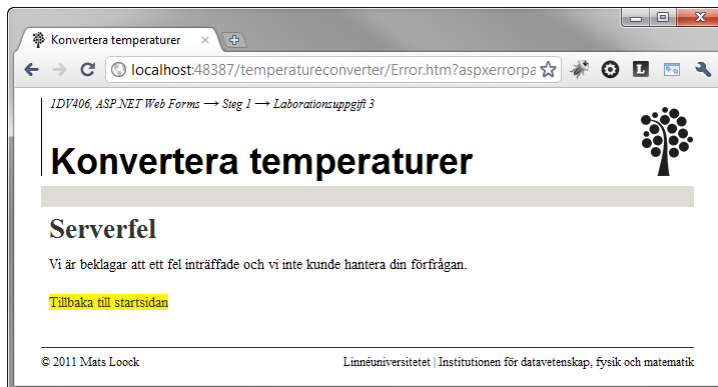
Formeln

$$[^{\circ}\text{C}] = ([^{\circ}\text{F}] - 32) \cdot \frac{5}{9}$$

använder du för att konvertera från grader Fahrenheit till grader Celsius.

Hantering av fel på servern

Skulle ett oväntat fel av något slag inträffa ska användaren slippa se en ”gul-ful” sida med ett automatgenererat felmeddelande. Istället ska du visa en sida med ett användarvänligare(?) meddelande. Figur 6 visar ett exempel på en sådan sida.



Figur 6. Anpassad sida som visas vid fel istället för en ”gul-ful” sida.

Mål

- Du ska förstå vikten av att validera data innan det behandlas vidare, både på klienten och på servern.
- Förhoppningsvis ska du inse att de olika valideringskontroller som finns underlättar valideringen.
- Att i en ”code behind”-fil programmatiskt kunna skapa kontroller som i slutänden renderas ut till klienten.

Tips

- På sidan "*ASP.NET Page Life Cycle Overview*", [http://msdn.microsoft.com/en-us/library/ms178472\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/ms178472(v=vs.100).aspx), hittar du information om i vilken ordning saker och ting sker på servern då ett webbformulär körs.
- På sidan "*Types of Validation for ASP.NET Server Controls*", [http://msdn.microsoft.com/en-us/library/bwd43d0x\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/bwd43d0x(v=vs.100).aspx), kan du läsa om valideringskontrollerna.
- På sidan "*How to: Add Rows and Cells Dynamically to a Table Web Server Control*", <http://msdn.microsoft.com/en-us/library/7bewx260.aspx> hittar du ett och annat som är intressant om Table-kontrollen. Även sidan *Table Class*, [http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.table\(en-us\).aspx](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.table(en-us).aspx), kan vara av intresse.
- Hur du använder en stilmall för att utforma dokumentet, gör det inte "inline", hittar du på sidan *Working with CSS Overview*, <http://msdn.microsoft.com/en-us/library/bb398931.aspx>.