Linnéuniversitetet
Kalmar Växjö

Laborationsanvisning

Galleriet

Steg 2, laborationsuppgift 1

Författare: Mats Loock

Kurs: ASP.NET Web Forms

*Kurskod:*1DV406

Linnéuniversitetet
Kalmar Växjö

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:

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!

Linnéuniversitetet
Kalmar Växjö

Innehåll

Uppgift

Inledning

Du ska skapa en webbapplikation där användaren kan ladda upp bilder, välja mellan tumnaglar av bilder och där vald tumnagel markeras och visas i ett större format.

Du har stor frihet att utforma webbapplikationen på det sätt du önskar. Figur 1 är ett exempel på hur användargränssnittet kan utformas, men du är fri att välja ett helt annat upplägg – under förutsättning att kraven uppfylls.

Då användaren valt en bild ska han/hon kunna välja att skapa ett bokmärke till sidan med bilden, d.v.s. information om vilken bild som ska visas måste vara en del av URL:en. Med andra ord måste du se till att en ”query string” innehåller lämplig information.

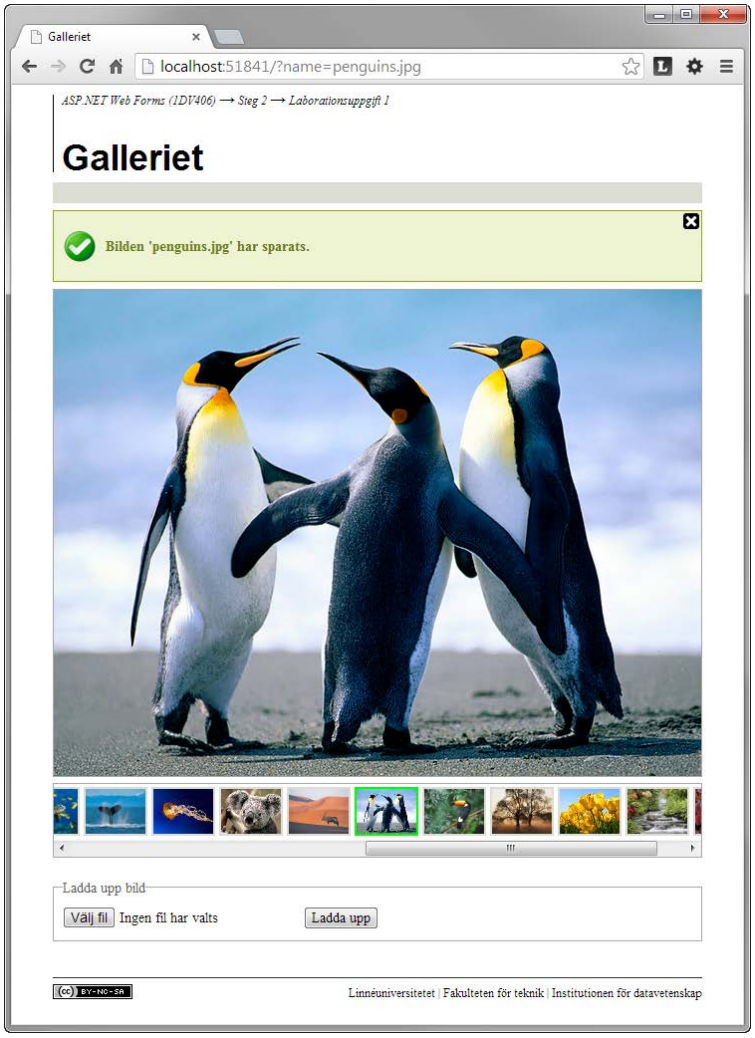
Figur 1. Webbsida efter att användaren klickat på en tumnagel.

Användargränssnittlagret

Viss del av funktionaliteten måste implementeras med JavaScript. Du väljer själv om du vill använda dig av ett JavaScript-bibliotek, som till exempel jQuery, eller inte.

Tumnaglarna ska presenteras så att användaren enkelt kan bläddra mellan dem. Vilken tumnagel som är vald ska indikeras. Helst ska du se till att även den valda tumnageln är synlig för användaren då sidan presenteras.

Användaren ska på lämpligt sätt informeras om en uppladdning av en bild lyckats. Väljer du att visa rättmeddelandet på samma sida som uppladdningssidan ska användaren kunna stänga rättmeddelandet. Rättmeddelande kan alternativt vara placerat i en helt separat sida.



Figur 2. Resultatet av en lyckad uppladdning där rättmeddelandet kan stängas.

Klickar användaren på kommandoknappen för att ladda upp en bild ska ett felmeddelande visas om ingen fil har valts eller då filändelsen inte är gif, jpg eller png.

Presentationslogiklagret

Du får fritt utforma formuläret där tumnagelbild väljs och bilder laddas upp. Presentationslogiklagret ska i möjligaste mån endast innehålla kod som har med hanteringen av kontroller att göra. Kod som har med hantering, förutom påfyllning av databundna kontroller och validering, av bilder får under inga omständigheter placeras i presentationslogiklagret (detta gäller så väl aspx- som ”code-behind”-fil) utan ska placeras i en affärslogiklagerklass.

Då det sker en GET av sidan ska det åtminstone serverkontrollerna `FileUpload` och `Button` renderas till klienten och därmed göra det möjligt för användaren att ladda upp en bild.

Finns det uppladdade bilder ska tumnaglar av dessa presenteras med hjälp av antingen en `ListView`- eller en `Repeater`-kontroll. Den databundna kontrollen du väljer ska innehåll kontroller som kan visa en bild, t.ex. `ImageButton` eller en kombination av `HyperLink` och `Image`.

Väljer du att använda kontroller av typen `ImageButton` tänk då på att du är tvungen att använda egenskapen `PostBackUrl`, för att skapa en URL som innehåller lämplig ”query string”, vilket leder till att JavaScript måste kunna exekveras på klienten.

Använder du istället `HyperLink` och egenskapen `NavigateUrl` behöver inte klienten kunna exekvera JavaScript.

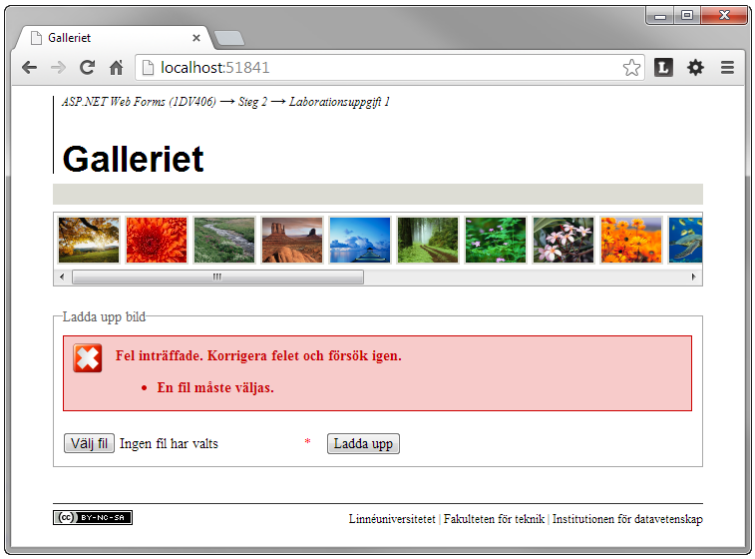
Validering

Bildens filnamn ska valideras så filändelsen är gif, jpg eller png. Valideringen ska utföras i första hand på klienten. Men det är oerhört viktigt att all data valideras på servern oavsett om det validerats på klienten eller inte.

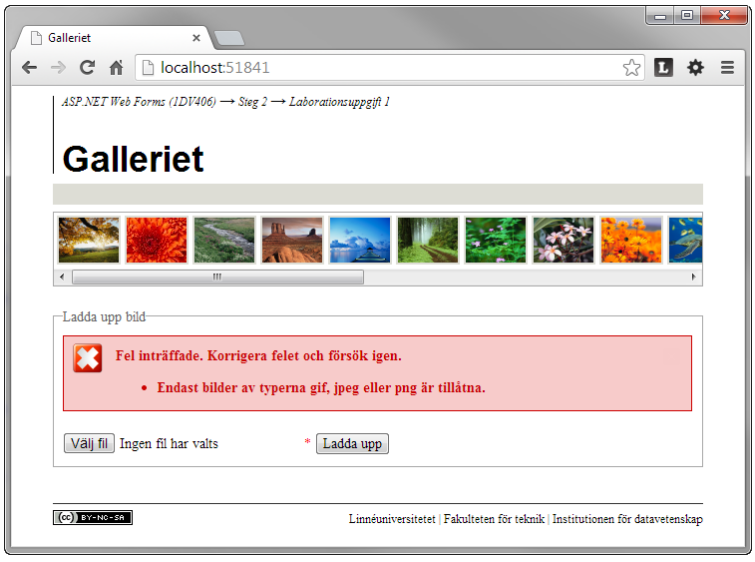
En fil av korrekt typ måste vara vald.

- Det måste finna ett filnamn.
- Giltiga filändelser är gif, jpg och png.

Med hjälp av kontrollen `RequiredFieldValidator` kan du säkerställa att en fil har valts av användaren. Kontrollen `RegularExpressionValidator` du till att undersöka om filnamnet avslutas med någon av de giltiga filändelserna.



Figur 3. Felmeddelande då användaren inte valt någon fil.



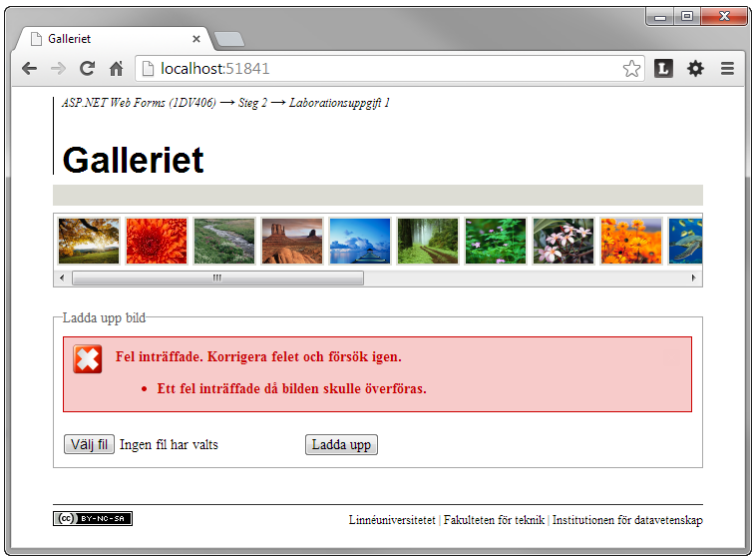
Figur 4. Felmeddelande då fil med otillåten filändelse valts.

Resultat av en uppladdning

Då en uppladdning av en bild lyckats ska den uppladdade bilden visas, dess tumnagel markeras och ett rättmeddelande visas. Klientens adressfält ska också uppdateras så att det innehåller en ”query string” med information om bilden som just blev uppladdad.

För att uppdatera klientens adressfält måste du se till att klienten gör en GET av sidan. Detta görs enklast genom att anropa metoden `Repsonse.Redirect` och skicka med lämplig URL.

Misslyckas uppladdningen av någon orsak kan det vara lämpligt att använda sig av samma `ValidationSummery`-kontroll som används av andra valideringskontroller för att visa ett felmeddelande. Detta görs enklast genom att instansiera och initiera ett objekt av typen `CustomValidator` vars referens läggs till samlingen `Page.Validators` med metoden `Add`.



Figur 5. Felmeddelande då användaren försöker överföra en fil som inte är en bild av tillåtet format.

Ingen form av filhantering får ske i presentationslogiklagret. Kontroller av typen `FileUpload` har metoden `SaveAs` men den får alltså inte användas. Istället ska den uppladdade filens innehåll skickas till ett affärslogiklagerobjekt. Den uppladdade filens innehåll kommer du åt via egenskapen `FileContent` som `FileUpload`-kontrollen har; egenskapen `FileName` ger filens namn.

Affärslogiklagret

Affärslogikklassen ansvara för all hantering av bilderna. Bilderna ska sparas i en underkatalog till den fysiska applikationsroten. Tumnagelbilderna placerar du lämpligen i en underkatalog i sin tur. Du väljer själv när tumnagelbilderna ska skapas. Antingen gör du det i samband med att en bild överförs och sparas, eller då de efterfrågas.

Innan en bild sparas måste den undersökas så att den har rätt filändelse och format. Om så inte är fallet ska ett undantag av typen `ArgumentException` kastas.

Hantering av namnkollisioner

Att undvika att skriva över filer som har samma namn som en fil som överförs kan du lösa på flera sätt. Ett enkelt sätt är att lägga till ett index, t.ex. (2), efter namnet om det redan är upptaget. Du använder `File.Exists` för att undersöka om en fil redan finns med samma namn som filen som ska överföras. Metoder som kan komma till användning för att skapa ett nytt namn kan t.ex. vara `Path.GetFileNameWithoutExtension`, `Path.GetExtension`.

Att skapa en tumnagel

Att skapa en tumnagel med utgångspunkt från en bild är mycket enkelt, om du känner till vad klassen `System.Drawing.Image` kan göra. Genom att använda metoderna `FromFile`, `GetThumbnailImage` och `Save` kan du skapa och spara en tumnagel av önskad storlek.

Satserna nedan skapar och sparar en tumnagel med storleken 60 x 45 pixlar med utgångspunkt från en bild som laddats upp.

```
var image = System.Drawing.Image.FromStream(stream); // stream -> ström med bild
var thumbnail = image.GetThumbnailImage(60, 45, null, System.IntPtr.Zero);
thumbnail.Save(path); // path -> fullständig fysisk filnamn inklusive sökväg
```

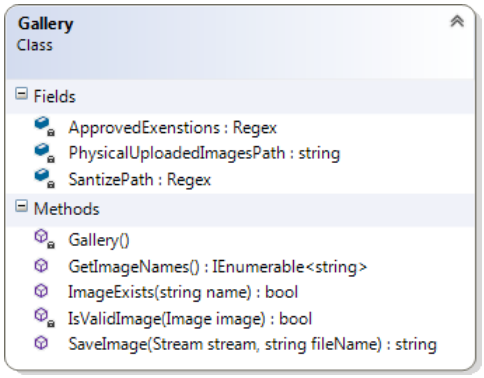
OBS! Du måste använda klassen `Image` i namnområdet `System.Drawing`, en helt annan klass än den `Image`-klass som finns i namnområdet `System.Web.UI.WebControls`.

Förslag på klass

I figur 8 nedan ser du ett klassdiagram med förslag på hur du kan utforma klassen som ska hantera bilderna. Du behöver inte utforma din klass på samma sätt utan kan skapa en egen om du så vill.

Genom att omsorgsfullt utforma klassen kommer det att bli enklare att för dig att skriva koden som tillhör användargränssnitt- och presentationslogiklagren.

Tänk på att klassen inte ska innehålla några beroenden och exempelvis inte innehålla någon kod som har med ASP.NET att göra.



Figur 6. Förslag på medlemmar i klassen som hanterar bilder. Det framgår inte av figuren att fälten, metoden ImageExists och konstruktorn är statiska.

Fält

`ApprovedExenstions` är ett privat statiskt fält, som är ”readonly”[1], av typen `Regex` och används för att med hjälp av ett reguljärt uttryck undersöka om en fil har en tillåten filändelse. Det reguljära uttrycket `^.*\.(gif|jpg|png)$` kan användas för att under söka om ett filnamn har filändelsen gif, jpg eller png.

Fältet initieras i den statiska konstruktorn.

`PhysicalUploadImagePath` är en privat statisk sträng som innehåller den fysiska sökvägen till katalogen där uppladdade filer sparas.

Med `AppDomain.CurrentDomain.GetData("APPBASE").ToString()` kan den fysiska sökvägen till applikationsroten erhållas. Använd gärna `Path.Combine()` för att skapa sökväg till katalogen där uppladdade filer sparas.

Fältet initieras i den statiska konstruktorn.

`SantizePath` är ett privat statiskt fält, som är ”readonly”[2], av typen `Regex` och används för att med hjälp av ett reguljärt uttryck se till att filnamn innehåller godkända tecken. Koden som följer kan användas för att skapa lämpligt `Regex`-objekt:

```
// http://stackoverflow.com/questions/146134/how-to-remove-illegal-characters-from-path-and-filenames/146162#146162
var invalidChars = new string(Path.GetInvalidFileNameChars());
SantizePath = new Regex(string.Format("[{0}]", Regex.Escape(invalidChars)));
```

För att sedan ersätta otillåtna tecken använd metoden `Regex.Replace()`.

Fältet initieras i den statiska konstruktorn.

Konstruktor

Konstruktorn är statisk och dess uppgift är att initiera de statiska ”readonly” fälten.

[1] Då fältet är statiskt och dessutom ”readonly” är det närmast att betrakta som en konstant vilket förklarar att fältets namn inleds med stor bokstav.

[2] Då fältet är statiskt och dessutom ”readonly” är det närmast att betrakta som en konstant vilket förklarar att fältets namn inleds med stor bokstav.

Metoder
`GetImagesNames` returnerar en referens av typen `IEnumerable<string>` till ett `List`-objekt innehållande bildernas filnamn sorterade i bokstavsordning. Klassen `DirectoryInfo` med metoden `GetFiles` är användbar. Det kan vara en god idé att se till att bara filer med filändelserna gif, jpg och png finns i listan.

`ImageExists` är en statisk metod som returnerar `true` om en bild med angivet namn finns katalogen för uppladdade bilder; annars `false`.

`IsValidImage` returnerar `true` om den uppladdade filens innehåll verkligen är av typen gif, jpeg eller png. Om image refererar till ett `System.Drawing.Image`-objekt kan dess MIME-typ undersökas med `image.RawFormat.Guid == System.Drawing.Imaging.ImageFormat.Gif.Guid`.

`SaveImage` verifierar att filen är av rätt MIME-typ (annars kastas ett undantag), säkerställer att filnamnet är unik, sparar bilden samt skapar och sparar en tumnagelbild. Filnamnet bilden sparas under returneras.

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 7 visar ett exempel på en sådan sida.

Figur 7. Anpassad sida som visas vid fel istället för en ”gul-ful” sida.

Krav

Sammanställning av krav som måste uppfyllas:

1. Då användaren valt en bild ska han/hon kunna välja att skapa ett bokmärke till sidan med bilden, d.v.s. information om vilken bild som ska visas måste vara en del av URL:en. Med andra ord måste du se till att en ”*query string*” innehåller lämplig information.
2. Bilderna ska sparas i en underkatalog i webbapplikationen på servern. Tumnagelbilderna placerar du lämpligen i en underkatalog i sin tur. Du väljer själv när tumnagelbilderna ska skapas. Antingen gör du det i samband med att en bild överförs, eller då de efterfrågas.
3. Användaren ska kunna välja bland tumnaglar den bild som ska visas i ett större format. Tumnaglarna ska presenteras med hjälp av en databunden kontroll och du väljer helt och hållet själv vilken. Den databundna kontrollen du väljer ska innehålla webbkontroller av lämplig typ för att kunna presenterar tumnagelbilder användaren kan klicka på för att få se bilden i ett större format.
4. Det ska vara möjligt att överföra en bild från klienten till servern och webbapplikationen. Finns redan en bild med samma namn som den som överförs ges den nya bilden samma namn fast med tillägget (2). Om t.ex. användaren överför bilden MinBild.png och en sådan fil redan finns på webbserver ges den nya filen namn MinBild (2).png. Finns redan MinBild (2).png ges den nya filen namnet MinBild (3).png, o.s.v.

5. Användaren måste bli informerad med ett meddelande då en överföring är gjord. Både ”rätt”- och felmeddelande måste visas. Anledningar till att en överföring kan betraktas som misslyckad kan vara att användaren försöker överföra en fil med felaktig MIME-typ. En annan felorsak kan vara att en fil utan innehåll överförs av användaren, eller att en för stor fil försöker överföras. Då både MIME-typ eller filändelse enkelt kan manipuleras måste det säkerställas att filen verkligen innehåller en bild av de tillåtna formaten.
6. Validering av att en fil är vald ska ske på klienten och servern då användaren försöker ladda upp en fil.
7. Validerings av vald fils filändelse ska ske på klienten och servern då användaren försöker ladda upp en fil. Endast filer med filändelsen gif, jpg och png är tillåtna.
8. Presentationslogiklagret får inte innehålla någon kod för hantering av bilder, annat den kod som krävs för att fylla på kontroller med information om bilder.
9. Affärslogikklassen får inte använda sig av några typer som tillhör namnutrymmen för webbtjänster.

Mål

Efter denna laboration kommer du bildat kunskap gällande hur du kan använda databundna kontroller. Under laborationen kommer du även att arbeta med ett antal klasser som har med filer att göra – allt från klasser du använder för att gå igenom alla filer i en katalog till klasser som har med att överföra filer från klient till server till klassen du använder för att skapa en tumnagel utifrån ett original. Du ska:

- Förstå hur databundna kontroller kan användas för att rendera multipla kontroller till klienten.
- Kunna använda `FileUpload`-kontrollen för att ladda upp filer från klienten till servern.
- Veta hur du arbetar med en ”query string”.
- Förstå vikten av att validera data innan det behandlas vidare, både på klienten och på servern samt vara väl förtrogen med hur du validerar innehållet i ett textfält med hjälp av valideringskontrollerna som ASP.NET erbjuder.
- Ha fått insikt om att data som skickas till affärslogiklagret måste valideras av affärslogiklagret då valideringen kan vara felaktigt implementerad, eller omöjligt att implementera, i användargränssnitt- och presentationslogiklager.
- Förstå att då användaren klickar på en knapp sker en ”postback” och en händelse skapas som kan tas om hand i ”code-behind”-filen.
- Kunna ta hand om händelsen `Click` som inträffar då användaren klickar på en knapp.

Tips

- FileUpload-kontrollen
  - Pro ASP.NET 4 in C# 2010, Fourth Edition, 550-552.
  - “*[How Do I:] Simple File Uploads in ASP.NET*”, http://www.asp.net/general/videos/how-do-i-simple-file-uploads-in-aspnet.
- Binda data till kontroll
  - “*Introducing Model Binding in Web Forms*”, http://www.asp.net/web-forms/tutorials/hands-on-labs/whats-new-in-web-forms-in-aspnet-45#Task_2_-_Introducing_Model_Binding_in_Web_Forms
  - Pro ASP.NET 4 in C# 2010, Fourth Edition, 360-363.

    - "*ASP.NET Data-Bound Web Server Controls Overview*", http://msdn.microsoft.com/en-us/library/ms228214.aspx.
    - "*Data-Binding Expressions Overview*", http://msdn.microsoft.com/en-us/library/ms178366.aspx.
- ListView-kontrollen
    - Pro ASP.NET 4 in C# 2010, Fourth Edition, 447-453.
    - " *ListView Class*", http://msdn.microsoft.com/en-us/library/bb459902.aspx
    - " *The Only Data-binding Control You'll Ever Need*", http://msdn.microsoft.com/en-us/magazine/cc337898.aspx
- "*Query String*"
    - Pro ASP.NET 4 in C# 2010, Fourth Edition, 248-249.
- DirectoryInfo Class
    - http://msdn.microsoft.com/en-us/library/8s2fzb02.aspx.
- System.Drawing.Image Class
    - http://msdn.microsoft.com/en-us/library/k7e7b2kd.aspx.
    - "*Create Thumbnail in C#*", http://ranafaisal.wordpress.com/2008/07/11/create-thumbnail-in-c/.
- File.Exists Method
    - http://msdn.microsoft.com/en-us/library/hzecd14s.aspx.
- Path Class
    - http://msdn.microsoft.com/en-us/library/3bdzys9w.aspx.