

# ASP.NET- applikationer



# Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET Web Forms vid Linnéuniversitetet.

## Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Looch, förutom Linnéuniversitetets logotyp och symbol samt ikoner, bilder och fotografier, 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:

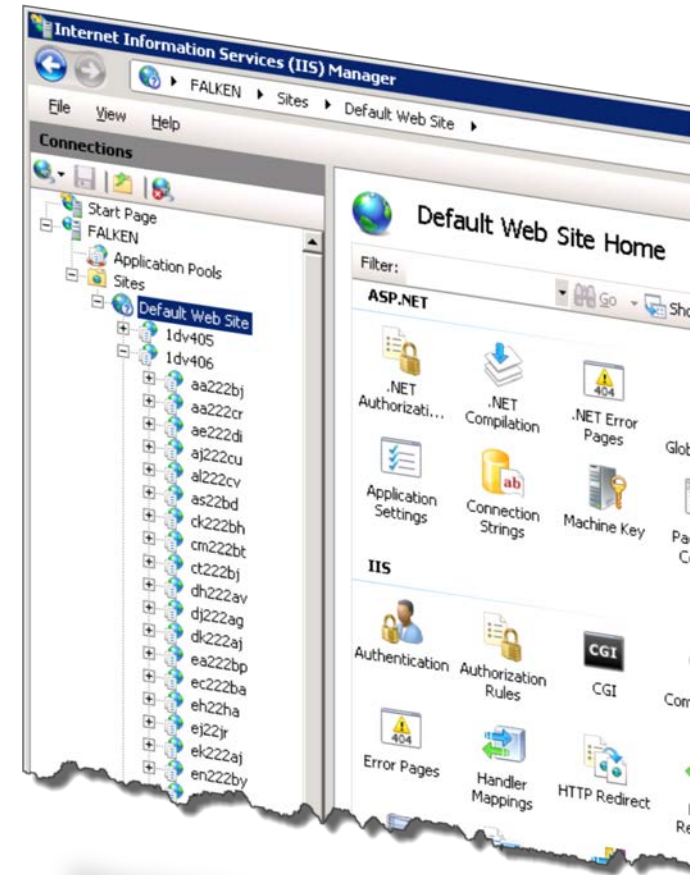
- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

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

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

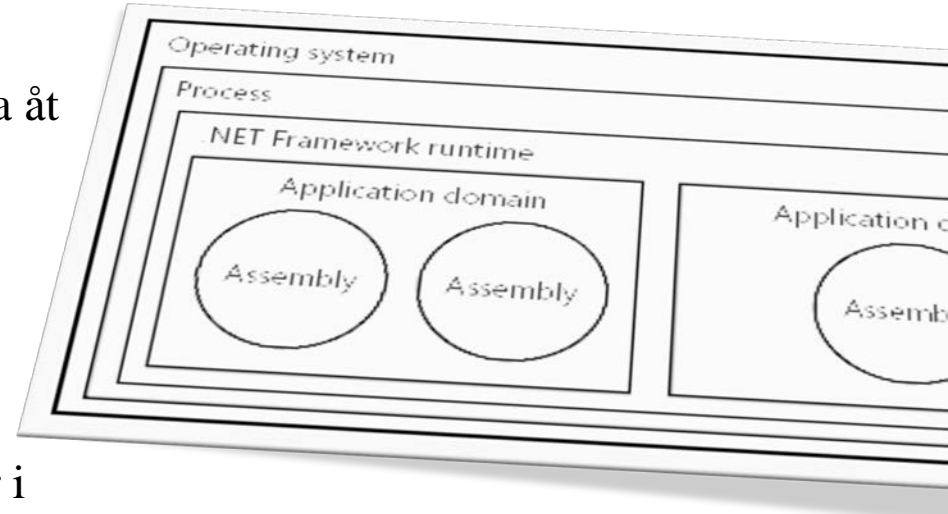
# ASP.NET-applikationer

- ✓ En webbapplikation skiljer sig "lite" från en vanlig desktopapplikation. Istället för en körbar exe-fil har en webbapplikation en kombination av webbsidor, "handlers", moduler, exekverbar kod m.m. som körs från en virtuell katalog på en webbserver.
- ✓ Användaren kör en webbapplikation genom att starta en webbläsare på klienten och via den efterfråga en specifik URL på en webbserver.
- ✓ Webbservern kan inte göra skillnad på olika webbapplikationer utan överlåter detta till ASP.NET som ser till att varje "request" hamnar i rätt applikationsdomän.
- ✓ Webbsidor placerade i en virtuella katalog (inklusive underkataloger) exekveras i samma applikationsdomän. Webbsidor i olika virtuella kataloger exekveras i separata applikationsdomäner.



# Applikationsdomänen

- ✓ Alla webbsidor i samma webbapplikation delar samma minne (global applikationsdata, "cachat" data, etc.).
- ✓ Olika webbapplikationer kan inte komma åt varandras minne.
- ✓ Alla webbsidor i en och samma webbapplikation delar samma konfigurationsinställningar (kan dock modifieras för enskilda underkataloger).
- ✓ Alla webbapplikationer skapar händelser i olika stadier under sin livstid. Dessa kan hanteras i filen `Global.asax`.

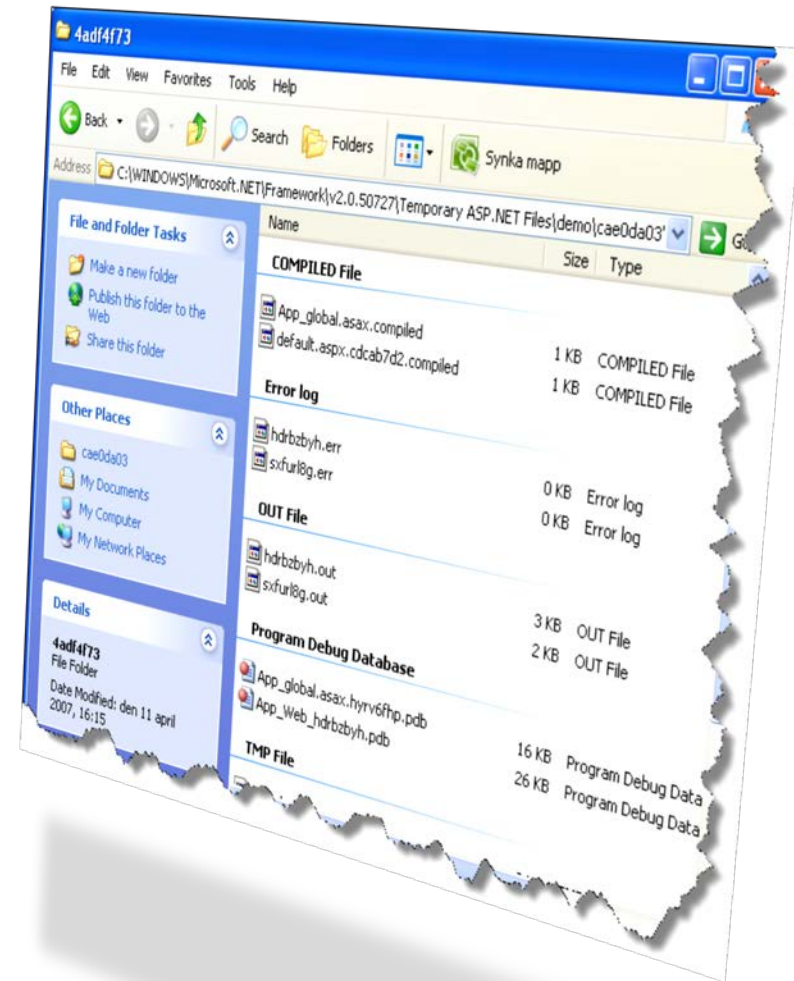


# En applikations liv

- ✓ Första gången en sida efterfrågas i en webbapplikation så skapar ASP.NET applikationsdomänen ("lazy initialization").
- ✓ Det finns flera orsaker till att en applikationsdomän kan stängas och en ny skapas:
  - Servern startas om.
  - Ett fel inträffar och applikationen startas om i ny domän.
  - Uppdatering av webbapplikationen har skett varför applikationen kan behöva startas i en ny domän.
  - Har en inställning att starta om sig efter viss tid eller efter att ett visst tillstånd inträffat.


# När du uppdaterar din applikation

- ✓ Du kan när som helst uppdatera din applikation utan att starta om servern eller att klienter eller andra applikationer ska bli lidande.
- ✓ CLR kör egentligen lokala kopior av filerna (som den låser, "shadow copy") men tack vare operativsystemet kan webbapplikationen få reda på om filer uppdaterats.
- ✓ En ny applikationsdomän startas, men för stunden köade anrop lever kvar i den gamla applikationens domän tills de slutfört sitt arbete.



# Globala händelser och Global.asax

- ✓ I Global.asax skriver du hanterarmetoder som abonnerar på globala händelser i webbapplikationen.
- ✓ En webbapplikation kan bara ha en Global.asax-fil (men det går bra utan också) som måste ligga i webbapplikationens rotkatalog.
- ✓ Även om det inte syns så deklarerar en Global.asax-fil metoder som tillhör en klass - applikationsklassen - som ärver från klassen `HttpApplication`.



```
<% Application Language="C#" %>
<script runat="server">

    void Application_Start(object sender, EventArgs e)
    {
        // Code that runs on application startup
    }

    void Application_End(object sender, EventArgs e)
    {
        // Code that runs on application shutdown
    }

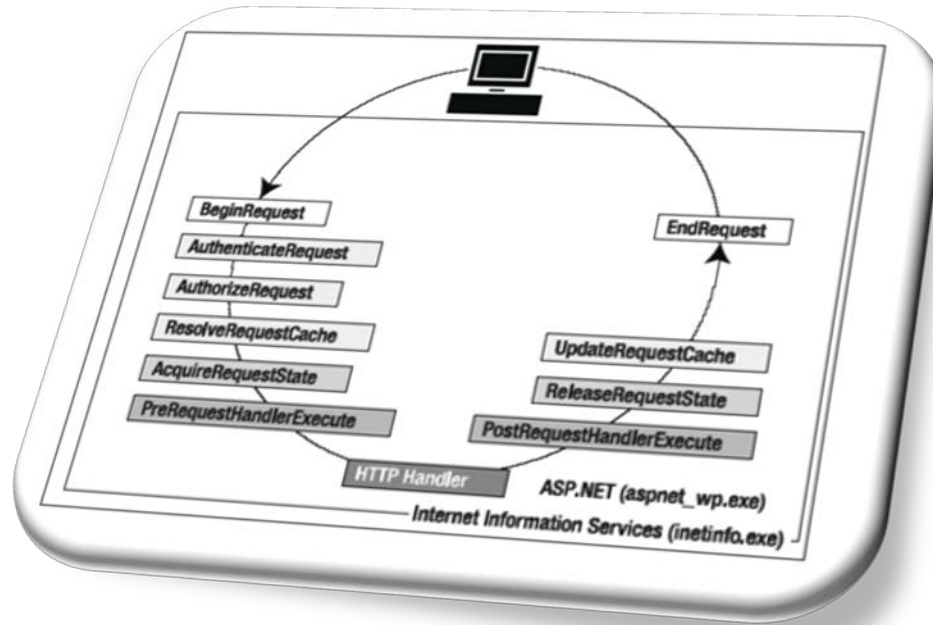
    void Application_Error(object sender, EventArgs e)
    {
        // Code that runs when an unhandled error occurs
    }

    void Session_Start(object sender, EventArgs e)
    {
        // Code that runs when a new session is started
    }

    void Session_End(object sender, EventArgs e)
    {
        // Code that runs when a session ends.
        // Note: The Session_End event is raised only when the sessionstate mode
        // is set to InProc in the Web.config file. If session mode is set to StateServer
        // or SQLServer, the event is not raised.
    }

    void Application_OnEndRequest()
    {
        Response.Write("Detta skrivs sist på sidan");
    }
}
```

# Applikationshändelser



- ✓ Du kan hantera två kategorier av händelser i `Global.asax`:
  - Händelser som inträffar vid varje förfrågan ("request").
  - Händelser som bara inträffar vid särskilda omständigheter.
    - `Applications_Start`, `Application_Error`, `Session_Start`, `Session_End`, `Application_End`, `Application_Disposed`



# Application Events

## ✓ Application\_Start()

- Utlöses när applikationsdomänen startar. Kan vara bra att ha för att initiera data som är konstant (navigeringssystem t.ex.).

## ✓ Application\_Error()

- När något ett oväntat fel inträffar. Användbart vidloggning.

## ✓ Application\_End()

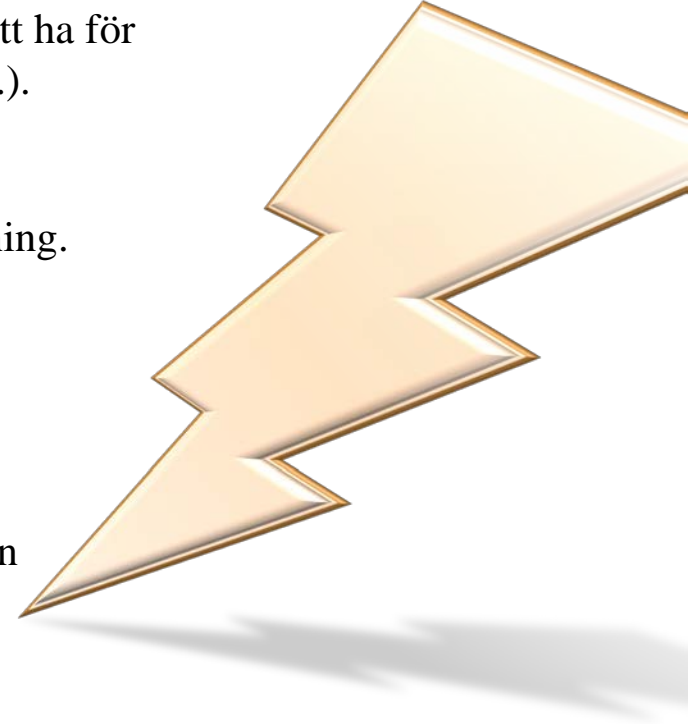
- När applikationen avslutas av någon anledning.

## ✓ Session\_Start()

- Varje gång en session startas. Bra att initiera data för den enskilde användaren.

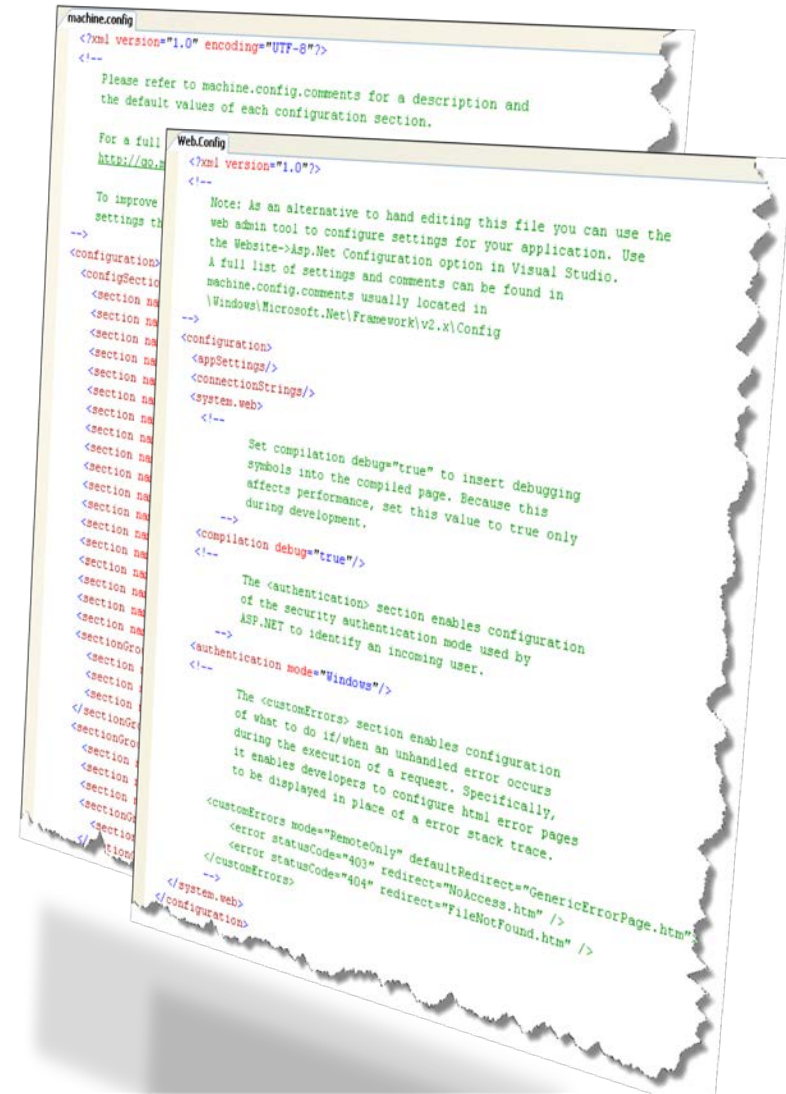
## ✓ Session\_End()

- Inträffar när sessionen avslutas. Användaren loggar ut, time-out.



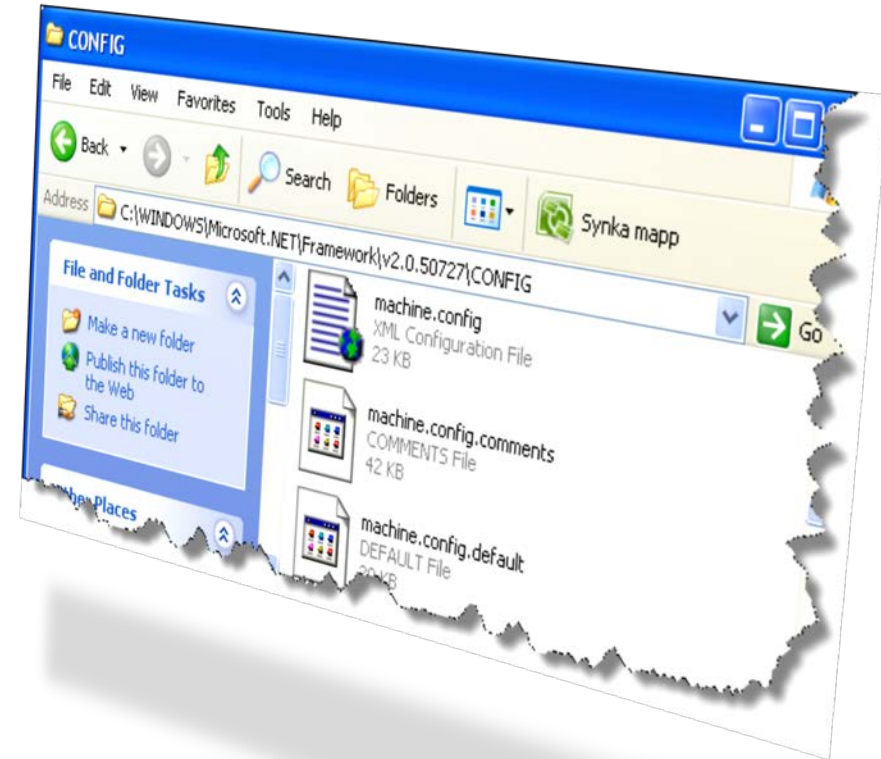
# Konfigurationsfiler

- ✓ ASP.NET sparar inställningar i XML-filer.
  - De är aldrig låsta.
  - De är lätta att komma åt, uppdatera och att kopiera via t.ex. FTP.
  - Enkla att editera och förstå(?).
  - Enkelt att skriva egna administrationsverktyg.



# Machine.config & co

- ✓ Innehåller inställningar för hur servern ska jobba med ASP.NET.
- ✓ Genom att studera filerna kan du se vilka standardinställningarna är.
- ✓ Kanske inte så vanligt att man som utvecklare redigerar men det är bra att känna till den.



# Web.config

- ✓ Innehåller inställningar för din specifika applikation.
- ✓ Exempel på inställningar är:
  - "authentication"
  - "debugging"
  - "default language"
  - "custom errorpage"
  - "connection strings"

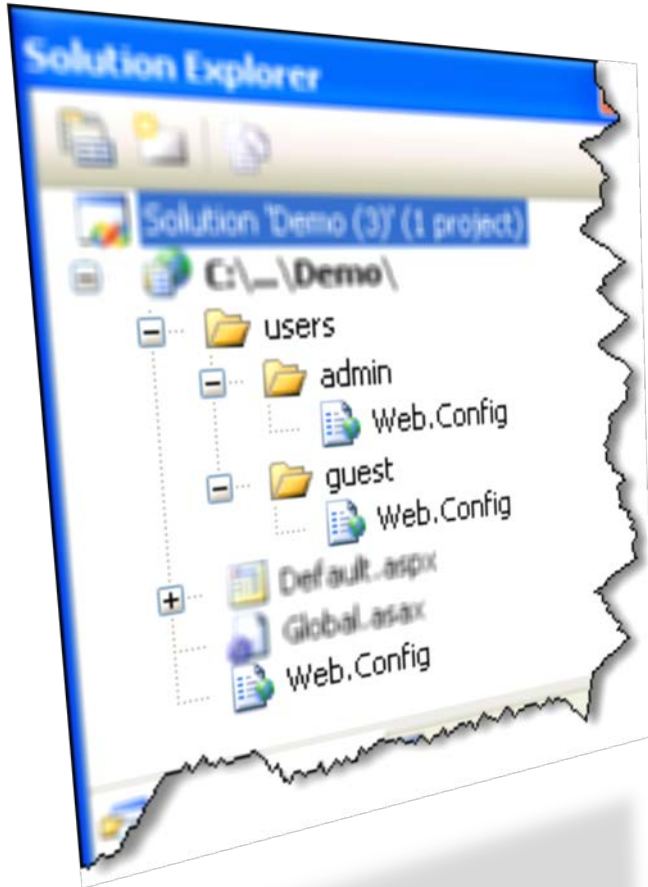


```
Web.config X
1 <?xml version="1.0"?>
2
3 <!--
4 For more information on how to configure your ASP.NET application, please visit
5 http://go.microsoft.com/fwlink/?LinkId=169433
6 -->
7
8 <configuration>
9   <connectionStrings>
10     <add name="ApplicationServices"
11         connectionString="data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDir|
12         providerName="System.Data.SqlClient" />
13   </connectionStrings>
14
15   <system.web>
16     <compilation debug="false" targetFramework="4.0" />
17
18     <authentication mode="Forms">
19       <forms loginUrl="~/Account/Login.aspx" timeout="2880" />
20     </authentication>
21
22     <membership>
23       <providers>
24         <clear/>
25         <add name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider" connect
26             enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="false"
27             maxInvalidPasswordAttempts="5" minRequiredNonalphanumericCharacters="6" minRequiredPasswordLength="6"
28             applicationName="/" />
29       </providers>
30     </membership>
31
32     <customErrors defaultRedirect="Error.htm" mode="RemoteOnly"/>
33   </system.web>
34 </configuration>
```

web.config X

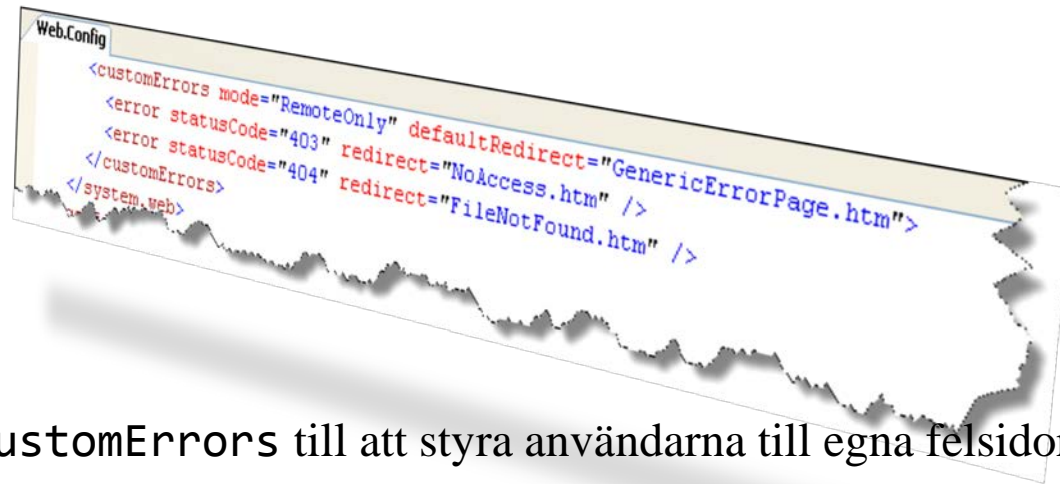
```
1 <?xml version="1.0"?>
2 <!--
3 For more information on how to configure your ASP.NET application, please visit
4 http://go.microsoft.com/fwlink/?LinkId=169433
5 -->
6 <configuration>
7   <system.web>
8     <compilation debug="true" targetFramework="4.0"/>
9     <customErrors defaultRedirect="Error.htm" mode="RemoteOnly"/>
10   </system.web>
11 </configuration>
```

# Web.config och arv



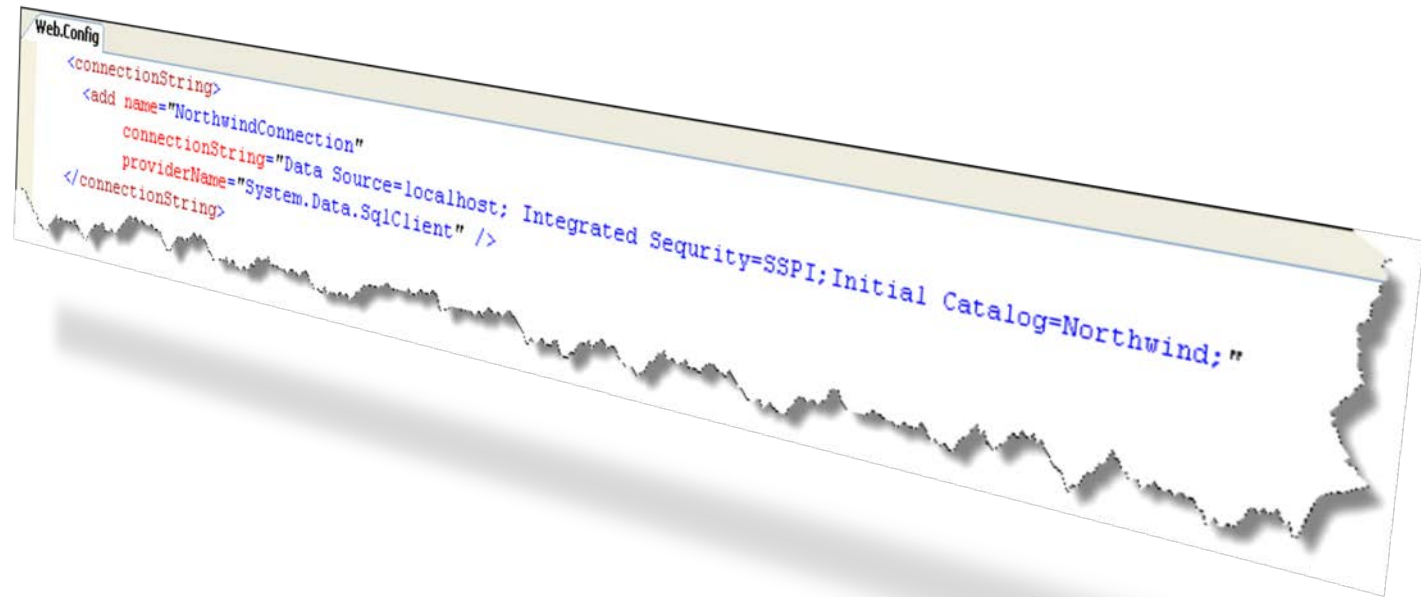
- ✓ `web.config`-filerna i de olika mapparna ärver inställningarna.
- ✓ `web.config`-filen i applikationsroten ärver från `machine.config` och en global `web.config`.
- ✓ Den `web.config` som ligger i mappen `users` ärver från `web.config` i applikationsroten.
- ✓ De `web.config` som ligger i mappen `admin` och `guest` ärver från `web.config` i mappen `users`.

# <customErrors>



- ✓ Du använder `customErrors` till att styra användarna till egna felsidor.
- ✓ Exemplet ovan definierar att om...
  - ... HTTP-felet 403 inträffar skickas användaren till sidan `NoAccess.htm`.
  - ... HTTP-felet 404 inträffar skickas användaren till sidan `FileNotFound.htm`.
  - ...övriga fel skickar användaren till `GenericErrorPage.htm`.
- ✓ `mode="RemoteOnly"` anger att dessa felsidor gäller endast för "remote"-klienter. Körs en klient på servern visas den "riktiga" felsidan som ger ett mer uttömmande felmeddelande.

# <connectionString>



- ✓ Anslutningssträngar till databaser kan med fördel definieras på en central plats, `web.config`, istället för att skriva den i "code-behind"-filerna.
- ✓ Då man skapar en `DataSource`-kontroll automatgenereras en anslutningssträng (mer om detta när ni kommer in på hur man skapar databaskoppling i ASP.NET).



# <appSettings>

- ✓ Ett utmärkt ställe att lägga statiska strängar, t.ex. sökvägar, URL:er, och meddelanden, som kan förekomma på flera sidor.
- ✓ Från "code-behind"-filerna kommer du åt värdena under `appSettings` med hjälp av klassen `ConfigurationManager`.
- ✓ Fördelen med att använda `appSettings` är att källkoden inte behöver modifieras, endast värdena i `web.config`.



```
<web.config>
  <appSettings>
    <add key="message" value="Tjabb! Välkommen!" />
  </appSettings>
</web.config>
```



```
using System.Configuration;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Message.Text = ConfigurationManager.AppSettings["message"];
    }
}
```