

XML-materiale

Karen Asferg

2013



Introduktion.....	4
Hvad er XML	4
Syntaks.....	5
DTD og XML Schema.....	7
Parsing XML-dokumenter	7
XSLT	8
XPath.....	8
.NET Frameworket.....	8
.NET og XML.....	9
Assemblies og Namespaces.....	9
.NET Configurationsfiler	10
At manipulere XML-dokumenter ved at bruge Document Object Model (DOM).....	10
Hvornår skal man bruge DOM?	11
CSS og DataBinding.....	12
Basal Data Binding.....	13
XSL-Transformationer (XSLT stylesheet)	15
XPath.....	20
RSS-feed.....	25
Websitenavigation ved hjælp af XML.....	27
Sitemaps	27
At definere et siteMap.....	28
SiteMap og masterpage.....	31
Eksempler	34
RSS-feed.....	34
Oprette, redigere og slette i et XML-dokument.....	37
Eksempel med SiteMap	42
Bilag	52

Introduktion

XML er de facto standarden til datarepræsentation og -transport.

- Fordele og features ved XML
- Grammatiske regler
- Kort introduktion til andre teknologier som DTS, XML Schema, parsere, XSLT og XPath
- Oversigt over .NET Frameworket
- Brug af XML i .NET Frameworket

Hvad er XML

XML står for Extensible Markup Language. Det er et markup-language, der bruges til at beskrive data. Det er en standardiseret måde at repræsentere tekstdata på. Ofte refererer man til XML-data som et XML-dokument. XML-data kan ikke gøre noget af sig selv. For at kunne afvikle data skal man have en parser. I modsætning til HTML, som fokuserer på at præsentere data, så fokuserer XML på at repræsentere data.

XML består af brugerdefinerede tags. Dette betyder, at man definerer og bruger sine egne tags i et XML-dokument.

XML blev godkendt af World Wide Web Consortium (W3C) i 1998.

XML er selvbeskrivende. Fordi brugeren selv definerer sine tags, så er XML mere læsevenligt end f. eks. kommasepareret tekst.

Et markup-language som f. eks. HTML har et fast antal definerede tags og attributter. Man kan ikke selv tilføje egne tags. I XML derimod definerer man sine egne tags.

Oprindeligt var kommeseparerede tekster en almindelig måde at flytte og repræsentere data på. Men for at kunne få det fulde udbytte, skulle man vide, hvor separationstegnet var. Dette gjorde læsning og skrivning til teksten mere besværlig. Dette problem har man ikke med XML, blot man har en parser til at fortolke dokumentet.

Fordi XML er en industristandard er det nemt at flytte data mellem forskellige platforme

Fordi XML kun er tekst, kan det bruges af mange forskellige systemer.

Men hvad betyder dette egentlig i praksis.

Har man en almindelig webapplikation med serversidescripting, så anmoder klienten om en side (en request). Denne anmodning fortolker serveren og sender svaret retur til klienten i form af HTML. Dette er fint nok, men

- Det virker kun med webbrowsere som klienter
- Svaret fra serveren er altid HTML. Det betyder at en windowsapplikation ikke kan bruge det,
- Data og visning er tæt koblete. Ønsker man at ændre præsentationen af data, skal der ske store ændringer
- Hvis andre applikationer skal bruge data, kan de ikke deles så nemt

Hvis man bruger XML i stedet, hvad så?

- Applikationen kan have forskellige typer af klienter. Den er ikke kun bundet til webbrowserne.
- Der er en svag kobling mellem klienten og afviklingslogikken.
- Nye typer af klienter kan tilføjes på et hvilket som helst tidspunkt uden at ændre på afviklingslogikken på serveren.
- Data og præsenterationslogik er adskilt fra hinanden. Webklienter har et sæt præsenterationslogik, Windowsapplikationer har et andet sæt.
- At dele data bliver let fordi outputtet er i XML-format.

Syntaks

For at kunne lave de viste eksempler, skal der oprettes et nyt website. Kald det f. eks. XMLSite

I et XML-dokument er der to sektioner, nemlig en prolog og en datastruktur.

Prologen identificerer dokumentet som et XML-dokument og inkluderer andre muligheder for specificerede afviklinger af dokumentet.

Datasektionen indeholder hierarkiet af dataelementer, som repræsenterer dokumentets informationsindhold.

Prologen kan indeholde tre dele:

- XML-erklæringen
- Procesinstruktioner
- Dokument type definitionen

XML-erklæringen skal stå i den allerførste linie. Erklæringen består af et obligatorisk versionsnummer, en valgfri encoding-erklæring og en valgfri option på om dokumentet er standalone eller ej.

```
<?xml version="1.0" [encoding="utf-8"] [standalone="yes"] ?>
```

Den gældende version er 1.0 og skal med.

En encoding-erklæring er nødvendig, når man bruger et ikke-latinsk (non-English) tegnsæt. Default er UTF-8. Standalone-attributten = "no" bruges, når der laves en reference til en fil med yderligere erklæringer for det aktuelle dokument. Default er standalone="yes"

Nedenfor er vist et meget enkelt XML-dokument

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <!-- Kundeliste -->
3 <kunder>
4   <kunde id="001">
5     <navn>Tomater Engros APS</navn>
6     <telefon>86123456</telefon>
7     <note>
8       <![CDATA[Kunde siden 1997]]>
9     </note>
10  </kunde>
```

```

11 <kunde id="002">
12   <navn>Staudhaven.dk</navn>
13   <telefon>88123478</telefon>
14   <note>
15     <![CDATA[Nyopstartet. Virker stabil]]>
16   </note>
17 </kunde>
18 </kunder>

```

Linie 1 er en procesinstruktion. En processinstruktion er information til applikationen om at den afvikler et XML-dokument. Procesinstruktionen pakkes ind i et par `<? og ?>`. XML-processinstruktionen har to attributter: version, encoding og standalone. Den nuværende W3C-anbefaling til XML er 1.0. Derfor er attributten version sat til 1.0.

En procesinstruktion er en reference til et andet dokument, som indeholder procesinstruktioner, der er relevante for XML-dokumentet. Disse instruktioner kan f.eks. være i et eksternt stylesheet, der transformerer XML-data til visning som f. eks. en webside. En procesinstruktion der linker til et eksternt stylesheet kan have dette udseende:

```
<? xml-stylesheet type="text.xsl" href="url" ?>
```

Linie 2 er en kommentar. Kommentarer kan skrives alle steder i et XML-dokument og kan strække sig over flere linier.

Linie 3 indeholder dokumentelementet, også kaldet rodelementet. Et XML-dokument har et og kun et rodelement. Alle elementer består af et starttag og et slutttag. Starttaget er her `<kunder>` og sluttaget er `</kunder>`.

Når man taler XML er der 3 termer, man skal skelne mellem.

- Element
- Node
- Tag

Element refererer til start- og slutttag

Tag er enten et starttag eller et slutttag

Node refererer til et element og alt det indhold, der er, inklusiv childelementer og tekst.

I `<kunder>`-elementet er der to `<kunde>`-noder. `<kunde>`-elementet har en attribut, id. Attributværdier skal stå i anførselstegn. `<kunde>`-elementet har 3 child-elementer, `>navn<`, `<telefon>` og `<note>`. Tekstværdien inden i elementer kaldes ofte textnoder. Nogle gange kan den tekst man gerne vil skrive indeholde specialtegn, f. eks. `< og >`. For at kunne gengive dette indhold bruger man CDATA (character data). Alt hvad der står indenfor en CDATA-sektion betragtes som en streng.

Et XML-dokument kan være well-formed. For at være dette er der nogle regler, der skal overholdes.

- XML er case sensitive. Det vil sige at <kunde>, <Kunde> og <KUNDE> behandles som tre forskellige tags
- Et XML-dokument har et og kun et rodelement. Og der skal være et rodelement
- Et starttag skal have et slutttag
- Starttags og sluttags skal nestes korrekt
- Attributværdier skal stå i anførselstegn

DTD og XML Schema

For at kunne overføre strukturer på ét XML-dokument til andre, skal man bruge en DTD (Document Type Definition) eller et XML-Schema. Her får man et sæt regler, der fortolker dataelementerne og deres afhængighed. Man får med andre ord en eksakt ide til hvordan dokumentet skal se ud og hvad der er elementer, attributter og nesting.

XML-dokumenter, der overholder en DTD eller et XML-Schema kaldes valide dokumenter. DTD er en ældre måde at validere XML-dokumenter på.

En intern DTD kan kodes som en del af et XML-dokument

En ekstern DTD kan kodes i et separat dokument. Hvis DTD-specifikationerne er i eksternt dokument, som hentes via url'en, så skal standalone-attributten være lig no i xml-erklæringen. Et XML-dokument, der lever op til DTD-specifikationerne er valid foruden at det også er well-formed.

Nu er XML-Schemaer en mere almindelig måde at validere et XML-dokument på. Der er flere fordele ved at bruge et XML-Schema frem for en DTD.

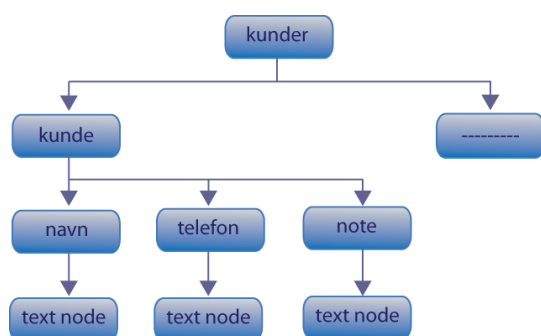
Parsing XML-dokumenter

XML-data kan ingenting selv. Derfor er man nødt til at have en parser. Med en parser kan man læse, skrive og ændre XML-dokumenter. Der er to kategorier af parsere.

- En DOM-baseret parser (Document Object Model)
- En SAX-baseret parser (Simple API for XML)

DOM-baserede hviler på W3C's anbefalinger og er den mest almindelige og mest populære form for parser.

En DOM-baseret parser betragter et XML-dokument som et omvendt træ. Se figur



DOM-visning af et XML-dokument

DOM-baserede parsere er read-write parsere. Det betyder, at man kan læse og skrive til XML-dokumenter. Man kan få fat i en hvilken som helst node i XML-dokumentet og det er derfor nødvendigt at hele XML-dokumentet indlæses i hukommelsen.

SAX-baserede parsere læser ikke hele XML-dokumentet ind i hukommelsen med det samme. De skanner dokumentet fra top til bund. Der laves forskellige events til forskellige dele af dokumentet og man kan så få disse events til at læse dokumentet. SAX-baserede parsere er read-only parsere og kan derfor ikke bruges til at ændre i et XML-dokument.

XSLT

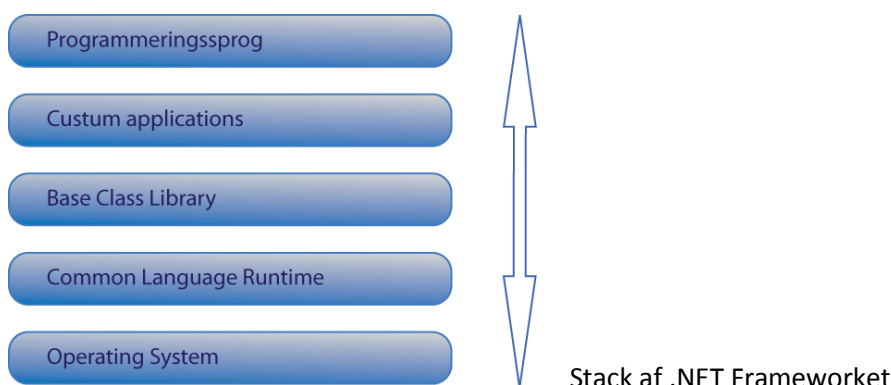
XSLT står for eXtensible Stylesheet Language Transformation og bruges til at transformere XML-dokumenter fra en form til en anden.

XPath

XPath er et expression language, der gør at man kan navigere gennem elementer og attributter i et XML-dokument. XPath består af forskellige XPath udtryk og funktioner, som man kan bruge til at leder efter og vælge elementer og attributter, der opfylder forskellige mønstre. XPath er også en W3C-anbefaling.

.NET Frameworket

Microsofts .NET Framework er en platform, som kan bruges til bl.a. webapplikationer, windowsapplikationer, forskellige komponenter og services. Til dette kan der bruges mange forskellige programmeringssprog.



Nederst har man operativsystemet. Dette kan f. eks. være Windows XP eller Windows 7. Oven på dette ligger Common Language Runtime (CLR) laget. Det er her alt det vigtige foregår, f. eks. styring af hukommelse, styring af tråde og sikkerhedstjek.

Oven over CLR findes base class Librariet, som er en enorm mængde af klasser, der gør at man kan gøre rigtig mange forskellige ting i ens applikationer, f. eks. arbejde med filer, databaser, XML og web for blot at nævne nogle få.

.NET og XML

I base class Librariet er der en del klasser, som gør at man kan arbejde med XML. Men sammenhængen mellem .NET Frameworket og XML stopper ikke her. Der er andre features i .NET, der gør brug af XML, f. eks.

- .NET Configurationsfiler
- ADO.NET
- ASP.NET Serverkontroller
- XML Serialization
- Remoting
- Web Services
- XML documentation
- SQL Server XML features
- XML Parsing
- XML-transformation

Assemblies og Namespaces

I assemblyet System.XML.dll findes forskellige XML-relaterede klasser, f. eks.

System.XML

Heri findes klasser, der muliggør læsning og skrivning af XML-dokumenter. I namespaceet finder man f. eks. en klasse som XmlDocument, der repræsenterer .NET Frameworkets DOM-baseret parser. Klasser som XmlTextReader og XmlTextWriter gør det muligt hurtigt og let at læse eller skrive XML-dokumenter. I dette namespace findes også klasser, der repræsenterer delelementer af et XML-Dokument, f. eks. XmlNode, XmlAttribute og XmlText.

System.XML.Schema

Dette namespace indeholder forskellige klasser, som gør det muligt at arbejde med Schemaer.

System.XML.Xsl

Dette namespace bruges som understøtning til XSLT-transformationer. Ved at bruge klasser fra dette namespace kan XML-data ændres fra en form til en anden. Klasser i dette namespace er bl.a. XslCompiledTransform, XslTransform og XslSettings.

System.XML.Serialization

I dette namespace findes klasser og attributter, der bruges til at serialize og deserialize objekter til og fra XML-formatet. Disse klasser bruges flittigt når man arbejder med web Services. Hovedklassen her er XmlSerializer. Af almindelig brugte attributklasser kan nævnes XmlAttributeAttribute, XmlRootAttribute og XmlTextAttribute.

System.XML.Linq

Dette namespace giver bl.a mulighed for at lade XML fra filer og streams, lave XML-træer via kode, spørge i XML-træer ved at bruge LINQ-operatorer, ændre XML-træer, validere XML-træer op

mod et Schema og ændre XML-træer. Nogle af de oftest brugte klasser er XDocument, XElement, XmlNode, XAttribute og XText.

.NET Configurationsfiler

Web.config-filen er en XML-fil. Man er her ikke begrænset til at bruge de prædefinerede tags. Man kan udvide web.config-filen

En web.config-fil kan indeholde fortrolige data. Men man kan meget let kryptere config-filen og derved øge sikkerheden på de data, der ligger i config-filen. Krypteringen er en indbygget del af frameworket og kræver derfor ingen kode fra udvikleren.

At manipulere XML-dokumenter ved at bruge Document Object Model (DOM)

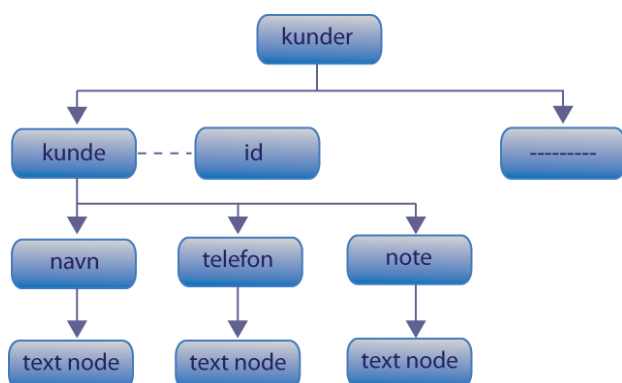
.NET Frameworkets DOM-parser betragter en XML-fil som et omvendt træ. Parseren loader XML-dokumentet og bygger træet bestående af elementer, attributter, kommentarer osv.

```
<?xml version="1.0" encoding="utf-8" ?>
<kunder>
  <kunde id="001">
    <navn>Tomater Engros APS</navn>
    <telefon>86123456</telefon>
    <note>
      <![CDATA[Kunde siden 1997]]>
    </note>
  </kunde>
  <kunde id="002">
    <navn>Staudehaven.dk</navn>
    <telefon>88123478</telefon>
    <note>
      <![CDATA[Nyopstartet. Virker stabil]]>
    </note>
  </kunde>
</kunder>
```

Ovenstående XML-dokument består af følgende dele

<?xml?>	process instruktioner
kunder	dokument elementet/rodelementet
kunde	element
id	attribut til <kunde> elementet
navn	childelement til <kunde> elementet
telefon	childelement til <kunde> elementet
note	childelement til <kunde> elementet

Desuden indeholder <navn>, <telefon> og <note> elementerne tekstværdier, som kaldes textnoder.



Hver del er en node. I .NET repræsenteres en node af en abstrakt klasse, XmlNode. Endog tekstværdier og attributter er noder. De skal dog behandles anderledes end almindelige noder.

De enkelte dele i ovenstående tabel er repræsenteret i en klasse

Del af XML-dokument	Klasse, som repræsenterer delen
Process Instructions	XmlProcessInstruction
Document element	XmlElement
Element	XmlElement
Attribute	XmlAttribute
Text values	XmlText
Nodes	XmlNode

Alle disse klasser arver direkte eller indirekte fra en basisklasse, XmlNode.

Hvornår skal man bruge DOM?

Inden man begynder at bruge DOM på sine XML-dokumenter, skal man sætte sig ind i de områder, hvor DOM er godt at bruge. Man skal samtidig også være klar over, at der er områder, hvor DOM ikke er så godt.

Om man skal bruge DOM eller ej styres af følgende hovedfaktorer:

READ/WRITE adgang: DOM tillader, at man kan læse og skrive i XML-dokumentet. Man skal her gøre sig klart, om der skal ændres i dokumentet eller ej. Skal der ikke, er der ingen grund til at bruge DOM.

NODETILGANG: Skal man have fat i tilfældige forskellige noder, er det nødvendigt at have adgang til hele XML-dokumentet. Hertil er DOM godt.

STØRRELSE: Er det små XML-dokumenter, man arbejder med, så er DOM OK. Er det store XML-dokumenter og skal man ikke kunne ændre eller finde tilfældige noder, så er sekventiel adgang fint.

CSS og DataBinding

Når man i en browser skal se et XML-dokument med endelsen .xml får man hele dokumentet at se som en åbn/luk liste af dataelementerne med deres hierarkiske struktur

```
<?xml version="1.0" encoding="utf-8" ?>
- <books>
+ <bog id="001">
- <bog id="002">
  <isbn>987-1-4302-0997-3</isbn>
  <titel>Beginning XML with C# 2008 From Novice to Professional</titel>
  <fornavn>Bipin</fornavn>
  <efternavn>Joshi</efternavn>
  <udgiver>Apress</udgiver>
  <aar>2008</aar>
  <pris>325</pris>
  <beskrivelse>XML er betydningsfuld, både når man arbejder med websites og
    med andre applikationer. Det bruges meget i .NET.</beskrivelse>
  </bog>
+ <bog id="003">
</books>
```

Her er vist XML-dokumentet boeger.xml.

Den første og den sidste bog er lukket sammen, mens bog nummer to er vist med alle elementer.

En af måderne, hvorpå man kan få vist data på en pænere måde, er ved at bruge et stylesheet.

Derfor oprettes stylesheetet Stylesheet.css og tilføjes XML-dokumentet boeger.xml via en procesinstruktion.

XML-dokumentet: boeger.xml

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <?xml-stylesheet type="text/css" href="StyleSheet.css"?>
3 <books>
4   <bog id="001">
5     <isbn>978-0-672-33011-7</isbn>
6     <titel>ASP.NET 3.5 Unleashed</titel>
7     <fornavn>Stephen</fornavn>
8     <efternavn>Walther</efternavn>
9     <udgiver>SAMS Publishing</udgiver>
10    <aar>2008</aar>
11    <pris>450</pris>
12    <beskrivelse>Den mest omfattende bog om ASP.NET 3.5 frameworket. Der kommer langt
13 omkring</beskrivelse>
14   </bog>
15   <bog id="002">
16     <isbn>987-1-4302-0997-3</isbn>
17     <titel>Beginning XML with C# 2008 From Novice to Professional</titel>
18     <fornavn>Bipin</fornavn>
19     <efternavn>Joshi</efternavn>
20     <udgiver>Apress</udgiver>
21     <aar>2008</aar>
22     <pris>325</pris>
23     <beskrivelse>XML er betydningsfuld, både når man arbejder med websites og med andre
24 applikationer. Det bruges meget i .NET.</beskrivelse>
25   </bog>
26   <bog id="003">
27     <isbn>987-87-7055-813-6</isbn>
28     <titel>Nu er det længe siden</titel>
29     <fornavn>Hanne</fornavn>
30     <efternavn>Reintoft</efternavn>
31     <udgiver>People's Press</udgiver>
32     <aar>2008</aar>
33     <pris>150</pris>
34     <beskrivelse>Handler om tiden efter de danske troppers nederlag ved Dybbøl i 1864. Man
35 følger her familien Vemmingård</beskrivelse>
36   </bog>
37 </books>
```

Linie 2 er henvisningen til det eksterne stylesheet. Både XML-dokumentet og stylesheetet ligger her i samme mappe.

ASP.NET 3.5 Unleashed Stephen Walther SAMS Publishing 2008 Den mest omfattende bog om ASP.NET 3.5 frameworket. Der kommer langt omkring
Beginning XML with C# 2008 From Novice to Professional Bipin Joshi Apress 2008 XML er betydningsfuld, både når man arbejder med webistes og med andre applikationer. Det bruges meget i .NET.
Nu er det længe siden Hanne Reintoft People's Press 2008 Handler om tiden efter de danske troppers nederlag ved Dybbøl i 1864. Man følger her familien Vemmingård

Nu ser det kedelige XML-dokument straks lidt pænere ud.

Basal Data Binding

Man kan integrere xml i en standard webside, i stedet for som i ovenstående eksempel at vise selve xml-filen. Dette sker via databinding. Der er forskellige xhtml-tags, der kan bruges som placeholder til xml. På denne måde får man langt større kontrol over sit layout og sin styling af xml-elementerne.

For at man kan gøre dette bruger man Data Source Objects (DSO).

Det er en ekstern datakilde, der er bundet til xhtml-tags på en webside. En datasource skal kunne vises som en tabel, dvs. med rækker og kolonner. Dette krav opfyldes af mange xml-filer.

En DSO arrangerer data i et recordset.

Man henviser til et Data Source Object via et xml-tag i xhtml-filen

```
<xml id="idnavn" src="url" />
```

databundenxml.htm

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4  <html xmlns="http://www.w3.org/1999/xhtml">
5  <head>
6      <title>Bunden xml-fil</title>
7
8      <style type="text/css">
9          .frame {width:500px; height:350px; border:solid 1px black; background-color:#F0F0F0;
10             padding:10px}
11          .titel {font-family:times new roman; font-size:16pt; font-weight:bold}
12          .label {font-family:arial; font-size:9pt; font-weight:bold; width:60px}
13          .author {font-family:arial; font-size:11pt; font-weight:bold}
14          .desc {font-family:times new roman; font-size:10pt}
15      </style>
16
17  </head>
18  <body>
19
20      <xml id="Boeger" src="boeger.xml"/>
21
22      <div class="frame">
23          <div class="titel" datasrc="#Boeger" datafld="titel"></div><br/>
24          <table border="0">
25              <tr>
26                  <td class="label">Author: </td>
27                  <td class="author">
28                      <span datasrc="#Boeger" datafld="fornavn"></span>&nbsp;
29                      <span datasrc="#Boeger" datafld="efternavn"></span>
30                  </td>
31              </tr>

```

```

32 <tr>
33   <td class="label">ISBN: </td>
34   <td><span datasrc="#Boeger" datafld="isbn"></span></td>
35 </tr>
36 <tr>
37   <td class="label">Udgiver: </td>
38   <td><span datasrc="#Boeger" datafld="udgiver"></span></td>
39 </tr>
40 <tr>
41   <td class="label">År: </td>
42   <td><span datasrc="#Boeger" datafld="aar"></span></td>
43 </tr>
44 <tr>
45   <td class="label">Pris: </td>
46   <td><span datasrc="#Boeger" datafld="pris"></span>&nbsp;kroner</td>
47 </tr>
48 </table>
49 <div class="desc" datasrc="#Boeger" datafld="beskrivelse"></div>
50 </div>
51 <br />
52
53 <!-- Vælg enten disse eller knapperne senere i koden -->
54 <!-- Navigation via links-->
55
56 <a href="#" onclick="Boeger.recordset.moveFirst()">Første</a>&nbsp;&nbsp;&nbsp;
57 <a href="#" onclick="Boeger.recordset.movePrevious()
58   if (Boeger.recordset.BOF) {
59     Boeger.recordset.moveFirst()
60   }">Forrige</a>&nbsp;&nbsp;&nbsp;
61 <a href="#" onclick="Boeger.recordset.moveNext()
62   if (Boeger.recordset.EOF) {
63     Boeger.recordset.moveLast()
64   }">Næste</a>&nbsp;&nbsp;&nbsp;
65 <a href="#" onclick="Boeger.recordset.moveNext()
66   if (Boeger.recordset.EOF) {
67     Boeger.recordset.moveLast()
68   }">Sidste</a>&nbsp;&nbsp;&nbsp;<br />
69
70 <!-- Navigation via knapper -->
71 <input type="button" value="|< Første bog"
72   onclick="Boeger.recordset.moveFirst()" />
73 <input type="button" value="< Forrige bog"
74   onclick="Boeger.recordset.movePrevious()
75   if (Boeger.recordset.BOF) {
76     Boeger.recordset.moveFirst()
77   }" />
78 <input type="button" value="Næste bog >"
79   onclick="Boeger.recordset.moveNext()
80   if (Boeger.recordset.EOF) {
81     Boeger.recordset.moveLast()
82   }" />
83 <input type="button" value="Sidste bog >|"
84   onclick="Boeger.recordset.moveLast()" />
85
86 </body>
87 </html>

```

For hvert eneste element specificerer datasrc-attributten Boegers DSO. Id-attributten i det tilknyttede xml-tag indeholder den værdi, der bruges til at binde xml-filen til siden. Datafld elementet viser feltnavnet (noden) til den værdi, der skal vises.

Navigationen gennem xml-filen sker fordi bog.noderne i det aktuelle eksempel bliver til individuelle records i recordsættet. En DSO er et ActiveX Data Object (ADO) og man kan derfor udnytte egenskaber og metoder i ADO'en. Disse scripts skrives ofte som JavaScript.

Tabellen viser egenskaber og metoder, der kan bruges sammen med DSO

Recordset-egenskaber	Beskrivelse
idnavn.recordset.BOF	Fortæller om den aktuelle recordposition er før den første record i recordsættet
idnavn.recordset.EOF	Fortæller om den aktuelle recordposition er efter den sidste record i recordsættet
idnavn.recordset.recordCount	Returnerer antallet af records i recordsættet

Recordset-metoder	Beskrivelse
idnavn.recordset.moveFirst()	Flytter sig til den første record i recordsættet
idnavn.recordset.moveLast()	Flytter sig til den sidste record i recordsættet
idnavn.recordset.moveNext()	Flytter sig til den næste record i recordsættet
idnavn.recordset.Previous()	Flytter sig til den forrige record i recordsættet

Man behøver ikke have sin xml-fil bundet til en separat webside. Man kan tilføje et <xml>-tag hvor som helst i en webside for at oprette en DSO, der er bundet til et relevant tag på siden.

Ikke alle tags kan bruges til at binde til. Nedenstående tabel viser, hvilke der kan bruges.

Tag	Attribut	Hvordan?
<a>	href	
<button>	innerText	<button datasrc="#id" datafld="field name"></button>
<div>	innerText	<div datasrc="#id" datafld="field name"></div>
<frame>	src	<frame datasrc="#id" datafld="field name"></frame>
<iframe>	src	<iframe datasrc="#id" datafld="field name"/>
	src	
<input type="button">	value	<input type="button" datasrc="#id" datafld="field name"/>
<input type="checkbox">	checked	<input type="checkbox" datasrc="#id" datafld="field name"/>
<input type="hidden">	value	<input type="hidden" datasrc="#id" datafld="field name"/>
<input type="password">	value	<input type="password" datasrc="#id" datafld="field name"/>
<input type="radio">	checked	<input type="radio" datasrc="#id" datafld="field name"/>
<input type="text">	value	<input type="text" datasrc="#id" datafld="field name"/>
<marquee>	innerText	<marquee datasrc="#id" datafld="field name"></marquee>
	innerText	
<textarea>	innerText	<textarea datasrc="#id" datafld="field name"></textarea>

Hvis man vil arbejde videre med dette emne, kan man f.eks. prøve at finde ud af, hvordan man viser flere records ad gangen og kan bladre i disse.

XSL-Transformationer (XSLT stylesheet)

De to foregående måder at vise et XML-dokument på er fine, hvis det er et simpelt XML-dokument, hvor der ikke stilles specielle krav som f. eks. at vise en enkelt node.

Til dette kan man bruge XSLT (eXtensible Stylesheet Language Transformations). Med XSLT kan man også programmere med betingelser og løkker. XSLT kan integreres fuldt med XHTML og CSS, så man faktisk får 100 procent bestemmelse over designet.

En XSLT-processor skal bruge to dokumenter. Et XML-dokument og et XSLT-dokument. XML-dokumentet med de data, der skal transformeres og XSLT-dokumentet med skabelonen for visningen. Dette er to

forskellige filer, der kædes sammen via XSLT-processoren, så XML-filen vises i forhold til de retningslinier, der er i XSLT-dokumentet. Både XML-filen og XSLT-filen læses ind i hukommelsen og parses over i et kildetræ og et skabelontræ. Ved at bryde skabelonen som en model konverterer XSLT-processoren XML-dokumentet til en form, der er kompatibel med det medie, der skal vise filen. F. eks. konverteres XML-dokumentet til XHTML, hvis det er en webside, der er mediet.

XSLT har elementer svarende til dem man finder i standardprogrammeringssprog. Det understøtter tal, strenge og boolske datatyper. Der er muligheder for at vælge, sortere og formatere og man kan lave betingelser og løkker.

At oprette og bruge et XSLT-dokument er ikke som at programmere i et almindeligt programmeringssprog. Man skal ikke erklære klasser og skrive funktioner eller bruge anden programmeringslogik. XSLT er et erklæringssprog. Oftest sker XSLT-transformationer via skabeloner, der henviser direkte til noder i et XML-dokument hvorved logikken er i XML-dokumentets struktur.

Nedenfor vises XSLT-dokumentet, som bruges til at vise et webside med boeger1.xml. Bemærk, at XSLT-filer har endelsen xsl eller xslt.

I boeger1.xml er der foretaget et par ændringer i forhold til boeger.xml. Fornavn og efternavn er pakket ind i et <forfatter>-tag og der er tilføjet et <billede>-tag.

Opret derfor en mappe med navnet Billeder og læg nogle billeder ind. Navnet på billedet skal svare til det, der står i xml-filen.

boeger1.xml

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <?xml-stylesheet type="text/xsl" href="boeger.xslt"?>
3 <books>
4   <bog id="001">
5     <isbn>978-0-672-33011-7</isbn>
6     <titel>ASP.NET 3.5 Unleashed</titel>
7     <forfatter>
8       <fornavn>Stephen</fornavn>
9       <efternavn>Walther</efternavn>
10    </forfatter>
11    <udgiver>SAMS Publishing</udgiver>
12    <aar>2008</aar>
13    <pris>450</pris>
14    <beskrivelse>Den mest omfattende bog om ASP.NET 3.5 frameworket. Der kommer langt
- omkring</beskrivelse>
15    <billede>Billeder/billede1.jpg</billede>
16  </bog>
17  <bog id="002">
18    <isbn>987-1-4302-0997-3</isbn>
19    <titel>Beginning XML with C# 2008 From Novice to Professional</titel>
20    <forfatter>
21      <fornavn>Bipin</fornavn>
22      <efternavn>Joshi</efternavn>
23    </forfatter>
24    <udgiver>Apress</udgiver>
25    <aar>2008</aar>
26    <pris>325</pris>
27    <beskrivelse>XML er betydningsfuld, både når man arbejder med websites og med andre
- applikationer. Det bruges meget i .NET.</beskrivelse>
28    <billede>Billeder/billede2.jpg</billede>
29  </bog>
30  <bog id="003">
31    <isbn>987-87-7055-813-6</isbn>
```



```

32 <titel>Nu er det længe siden</titel>
33 <forfatter>
34   <fornavn>Hanne</fornavn>
35   <etternavn>Reintoft</etternavn>
36 </forfatter>
37 <udgiver>People's Press</udgiver>
38 <aar>2008</aar>
39 <pris>150</pris>
40 <beskrivelse>Handler om tiden efter de danske troppers nederlag ved Dybbøl i 1864. Man
- følger her familien Vemmingård</beskrivelse>
41 <billede>Billeder/billede3.jpg</billede>
42 </bog>
43 </books>

```

I linie 2 er tilføjet en henvisning til xslt-stylesheetet boeger.xslt. Type skal her være text/xsl. Href er placering og navn på fil.

boeger.xslt

```

1 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2
3   <xsl:template match="/">
4
5     <html>
6       <head>
7         <style type="text/css">
8           table {border-collapse:collapse}
9           tr#head {background-color:#E6E6E6}
10          td {padding:3px}
11          td#title {font-style:italic}
12        </style>
13
14      </head>
15      <body>
16
17        <h3>Data fra den første bognode</h3>
18
19        <table border="1">
20          <tr id="head">
21            <th>ISBN</th>
22            <th>Title</th>
23            <th>Author</th>
24            <th>Publisher</th>
25            <th>Year</th>
26            <th>Price</th>
27          </tr>
28
29          <tr>
30            <td>
31              <xsl:value-of select="books/bog/isbn"/>
32            </td>
33            <td id="title">
34              <xsl:value-of select="books/bog/titel"/>
35            </td>
36            <td>
37              <xsl:value-of select="books/bog/forfatter/fornavn"/>&#160;
38              <xsl:value-of select="books/bog/forfatter/etternavn"/>
39            </td>
40            <td>
41              <xsl:value-of select="books/bog/udgiver"/>
42            </td>
43            <td>
44              <xsl:value-of select="books/bog/aar"/>
45            </td>
46            <td>
47              <xsl:value-of select="books/bog/pris"/>
48            </td>
49          </tr>
50
51        </table>
52

```

53	<code></body></code>
54	<code></html></code>
55	
56	<code></xsl:template></code>
57	
58	<code></xsl:stylesheet></code>

Når man viser filen boeger1.xml i browseren får man dette resultat, som viser data fra den første bognode i XML-dokumentet.

Data fra den første bognode

ISBN	Title	Author	Publisher	Year	Price
978-0-672-33011-7	ASP.NET 3.5 Unleashed	Stephen Walther	SAMS Publishing	2008	450

Selve XSLT-dokumentet bliver identificeret via

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    ..... KODE
</xsl:stylesheet>
```

Linie 3 er `<xsl:template>` elementet, som omkranser stylekode og xhtml-tags. Denne template beskriver XHTML-formatteringen. Derefter kommer XHTML-tags, et embedded stylesheet og de værdier fra xml-dokumentet, som skal vises

Det fortæller browseren, at det er en xslt-fil. I elementet skal der stå `version="1.0"` og der skal være et `xmlns` namespace.

XSLT-dokumenter har forskellige skabeloner, der beskriver de transformationer, der skal ske. Hvis output er en XHTML-fil, så vil skabelonen sætte XHTML-kode ind sammen med de referencer, der skal laves til XML-dataerne.

I ovenstående eksempel vises 7 child-noder fra den første `<bog>`-parent node. Node-værdier vises med syntaksen `<xsl:value-of>`, hvor `select`-attributten peger på den node, der skal vises.

Nedenstående tabel viser de tre mest almindelige XSLT-elementer, som bruges til at definere og tilføje formatterings skabeloner for at kunne vise en XML-fil som en XHTML-fil.

XSLT-element	Beskrivelse
<code><xsl:template match="udtryk"/></code>	Definerer en skabelon som genererer det ønskede output for noder af en bestemt type og indhold. <i>Udtryk</i> identificerer kildenoden eller kildenoderne, som skabelonen skal gælde for.
<code><xsl:apply-templates select="udtryk"/></code>	Viser XSLT-processoren, hvilken skabelon, der skal bruges, baseret på type og indhold af den enkelte valgte node. Værdien af <code>select</code> -attributten er et udtryk, der evaluerer på et nodesæt. Det valgte nodesæt afvikles i dokumentrækkefølge med mindre der er specificeret en sorteringsrækkefølge.
<code><xsl:value-of select="udtryk"/></code>	Returnerer værdien af en node, der er identificeret i <i>udtryk</i> .

Man kan selvfølgelig også få vist hele xml-dokumentet eller dele af det. For at kunne gøre dette skal der bruges et nyt xslt-dokument. Opret derfor et nyt xslt-dokument. Kald det boeger1.xslt. Husk at rette referencen i xml-filen.

boeger1.xslt

```

1 <xsl:stylesheet version="1.0"
2 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3
4 <xsl:template match="/">
5
6 <html>
7 <body>
8
9 <style>
10 table {border-collapse:collapse}
11 tr#head {background-color:#E6E6E6}
12 td {padding:3px}
13 td#title {font-style:italic}
14 </style>
15
16 <h3>Alle data vist</h3>
17
18 <table border="1">
19 <tr id="head">
20 <th>Billede</th>
21 <th>ISBN</th>
22 <th>Titel</th>
23 <th>Forfatter</th>
24 <th>Udgiver</th>
25 <th>År</th>
26 <th>Pris</th>
27 </tr>
28
29 <xsl:apply-templates select="books/bog"/>
30
31 <!-- <xsl:apply-templates select="books/bog[aar > 2007]">
32 <xsl:sort select="pris" order="descending"/>
33 </xsl:apply-templates> -->
34 </table>
35
36 </body>
37 </html>
38
39 </xsl:template>
40
41 <xsl:template match="bog">
42
43 <tr>
44 <td>
45 
46 </td>
47 <td>
48 <xsl:value-of select="isbn"/>
49 </td>
50 <td id="title">
51 <xsl:value-of select="titel"/>
52 </td>
53 <td>
54 <xsl:apply-templates select="forfatter" />
55 </td>
56 <td>
57 <xsl:value-of select="udgiver"/>
58 </td>
59 <td>
60 <xsl:value-of select="aar"/>
61 </td>
62 <td>
63 <xsl:apply-templates select="pris" />
64 </td>
65 </tr>
66
67 </xsl:template>
68
69 <xsl:template match="forfatter">
70 <xsl:value-of select="efternavn"/>&#160;
71 <xsl:value-of select="fornavn"/>
72 </xsl:template>

```

```






73
74 <xsl:template match="pris">
75   <xsl:value-of select="format-number(text(), '###,##0.00 kr.')" />
76 </xsl:template>
77
78 </xsl:stylesheet>

```




Man kan selvfølgelig også få vist hele xml-dokumentet eller dele af det. For at man kan dette skal man bruge `<xsl:apply-templates>` elementet. Dette element returnerer en nodeliste som tilknytter et `<xsl:template>`-element til hver node i listen. På den måde løber man hele listen igennem. For forfatter og pris er der specielle styles. Pris skal formateres og forfatter skal holde sammen på fornavn og efternavn. Derfor skal disse to have en `<xsl-apply template>` tilknyttet. Se linie 54 og 63.

Fra line 31 til 33 er der en udkommenteret kode. Gøres den aktiv og linie 29 udkommenteres i stedet, får man kun vist de bøger, der er nyere end 2007.

Alle data vist

Billede	ISBN	Titel	Forfatter	Udgiver	År	Pris
	978-0-672-33011-7	ASP.NET 3.5 Unleashed	Walther Stephen	SAMS Publishing	2008	450.00 kr.
	987-1-4302-0997-3	Beginning XML with C# 2008 From Novice to Professional	Joshü Bipin	Apress	2008	325.00 kr.
	987-87-7055-813-6	Nu er det længe siden	Reintoft Hanne	People's Press	2008	150.00 kr.
	987-0-9802858-4-0	The art & science of JavaScript	Adams Cameron	SitePoint	2007	225.00 kr.
	978-1-59059-614-2	CSS Mastery: Advanced web Standard Solutions	Budd Andy	Friends of ED	2006	369.00 kr.

Alle bøger vises

Billede	ISBN	Titel	Forfatter	Udgiver	År	Pris
	978-0-672-33011-7	ASP.NET 3.5 Unleashed	Walther Stephen	SAMS Publishing	2008	450.00 kr.
	987-1-4302-0997-3	Beginning XML with C# 2008 From Novice to Professional	Joshü Bipin	Apress	2008	325.00 kr.
	987-87-7055-813-6	Nu er det længe siden	Reintoft Hanne	People's Press	2008	150.00 kr.

Kun bøger nyere end 2007 vises

XPath

XPath er et sprog specielt beregnet til at finde noder i et XML-dokument. Med XPath kan man finde dele af et XML-dokument. Man kan arbejde med strenge, tal og boolean.

XPath-udtryk identificerer de enkelte noder på basis af deres type, navn og værdier såvel som på deres indbyrdes forhold.

Vil man f. eks. finde priser, der er over 200 kr. så skriver man

`/books/bog[pris > 200.00]/pris`

Dette udsagn bruges så i select-attributten i <xsl:apply-template> elementet, hvorefter man så får returneret en liste med noder, der opfylder søgekriteriet.

Et XPath-udtryk beskriver en sti gennem et XML DOM hierarki med en /-separeret liste til at identificere de enkelte noder.

Man kan bruge den præcise sti som vist ovenfor eller man kan bruge wildcards til at beskrive ukendte elementer, når man skal vælge noder.

For eksempel henter udtrykket `books/bog/*` alle de noder, der er childnoder til bog og som findes inden for rodnoden `<books>`.

Ved at bruge firkantede parenteser kan man finde grene på ens sti. For eksempel så siger udtrykket

`books/bog[forfatter]/efternavn`

at man kun skal lede noden efternavn inden for noden forfatter.

Yderligere kan man gøre dette udtryk endnu mere præcist ved at sætte en værdi på

`books/bog[forfatter/efternavn='Adams']`

Når man bevæger sig ned gennem et xml-træ kan stien enten være relativ eller absolut.

En absolut begynder altid med /. Det vil sige, at `/books/bog/titel` begynder at lede i roden og derefter arbejder sig ned i gennem bog-noderne indtil den kan hente titelnoderne.

`//titel` er det sammen som `/books/bog/titel`. Hver slash indikerer et step. Man kan også bruge wildcards som nodenavn `/*/titel`. For overskuelighedens skyld er det oftest bedst at skeve den eksakte sti.

De anvendte operatorer er de gængse, kendt fra andre programmeringssprog

=	er lig
!=	er forskellig fra
<	mindre end
>	større end
<=	mindre end eller lig med
>=	større end eller lig med

Desuden kan AND og OR bruges til at lave sammenligninger med

Udtrykket `position()` returnerer eller sætter rækkefølgen på en node (starter med 1) baseret på dets fysiske placering i træet.

`books/bog[position()=1]/titel` (kan forkortes til `books/bog[1]/titel`)

vælger titelnoden fra den første <bog>-node .

Betingelser og løkker

Ved at bruge `<xsl:for-each>` elementet får man fuld kontrol over en løkke. Med if-strukturen kan man sætte betingelser op.

Opret et nyt xslt-dokument. Kald det for `boegerforeachif.xslt`

`boegerforeachif.xslt`

```
1  <xsl:stylesheet version="1.0"
2  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3
4  <xsl:template match="/">
5
6  <html>
7  <body>
8
9  <style>
10   table {border-collapse:collapse}
11   tr#head {background-color:#E6E6E6}
12   td {padding:3px}
13   td#title {font-style:italic}
14   img#billed {width:50px;}
15 </style>
16
17 <h3>Alle bøger vist med forløkke</h3>
18
19 <table border="1">
20 <tr id="head">
21 <th>Billede</th>
22 <th>ISBN</th>
23 <th>Titel</th>
24 <th>Forfatter</th>
25 <th>Udgiver</th>
26 <th>År</th>
27 <th>Pris</th>
28 </tr>
29
30 <xsl:for-each select="books/bog">
31 <tr>
32 <td>
33 
34 </td>
35 <td>
36 <xsl:value-of select="isbn"/>
37 </td>
38 <td id="title">
39 <xsl:value-of select="titel"/>
40 </td>
41 <td>
42 <xsl:value-of select="forfatter/efternavn"/>,&#160;
43 <xsl:value-of select="forfatter/fornavn"/>
44 </td>
45 <td>
46 <xsl:value-of select="udgiver"/>
47 </td>
48 <td>
49 <xsl:value-of select="aar"/>
50 </td>
51 <td>
52 <xsl:value-of select="format-number(pris, '###,##0.00 kr.')"/>
53 </td>
54 </tr>
55 </xsl:for-each>
56 </table>
57 <br/>
58 <br/>
59 <h3>Bøger nyere end 2007 (betingelse)</h3>
```

```

60 <table border="1">
61 <tr id="head">
62 <th>Billede</th>
63 <th>ISBN</th>
64 <th>Titel</th>
65 <th>Forfatter</th>
66 <th>Udgiver</th>
67 <th>År</th>
68 <th>Pris</th>
69 </tr>
70 <xsl:for-each select="books/bog">
71 <xsl:if test="aar > 2007">
72 <tr>
73 <td>
74 
75 </td>
76 <td>
77 <xsl:value-of select="isbn"/>
78 </td>
79 <td id="title">
80 <xsl:value-of select="titel"/>
81 </td>
82 <td>
83 <xsl:value-of select="forfatter/efternavn"/>, &#160;
84 <xsl:value-of select="forfatter/fornavn"/>
85 </td>
86 <td>
87 <xsl:value-of select="udgiver"/>
88 </td>
89 <td>
90 <xsl:value-of select="aar"/>
91 </td>
92 <td>
93 <xsl:value-of select="format-number(pris, '###,##0.00 kr.')"/>
94 </td>
95 </tr>
96 </xsl:if>
97
98 </xsl:for-each>
99
100 </table>
101 <br/>
102 <br/>
103 <h3>Bøger vist med farve i forhold til udgivelsesår (choose)</h3>
104 <table border="1">
105 <tr id="head">
106 <th>Billede</th>
107 <th>ISBN</th>
108 <th>Titel</th>
109 <th>Forfatter</th>
110 <th>Udgiver</th>
111 <th>År</th>
112 <th>Pris</th>
113 </tr>
114 <xsl:for-each select="books/bog">
115 <tr>
116 <td>
117 
118 </td>
119 <td>
120 <xsl:value-of select="isbn"/>
121 </td>
122 <xsl:choose>
123 <xsl:when test="aar < 2007">
124 <td style="color:green">
125 <b>
126 <xsl:value-of select="titel"/>
127 </b>
128 </td>
129 </xsl:when>
130 <xsl:when test="aar = 2008">
131 <td style="color:red">
132 <b>
133 <xsl:value-of select="titel"/>

```

```

134         </b>
135     </td>
136 </xsl:when>
137 <xsl:otherwise>
138     <td style="color:blue">
139         <b>
140             <xsl:value-of select="titel"/>
141         </b>
142     </td>
143 </xsl:otherwise>
144 </xsl:choose>
145 <td>
146     <xsl:value-of select="forfatter/efternavn"/>, &#160;
147     <xsl:value-of select="forfatter/fornavn"/>
148 </td>
149 <td>
150     <xsl:value-of select="udgiver"/>
151 </td>
152 <xsl:choose>
153 <xsl:when test="aar < 2007">
154     <td style="color:green">
155         <b>
156             <xsl:value-of select="aar"/>
157         </b>
158     </td></xsl:when>
159 <xsl:when test="aar = 2008">
160     <td style="color:red">
161         <b>
162             <xsl:value-of select="aar"/>
163         </b>
164     </td>
165 </xsl:when>
166 <xsl:otherwise>
167     <td style="color:blue">
168         <b>
169             <xsl:value-of select="aar"/>
170         </b>
171     </td>
172 </xsl:otherwise>
173 </xsl:choose>
174 <td>
175     <xsl:value-of select="format-number(pris, '###,##0.00 kr.')"/>
176 </td>
177 </tr>
178
179 </xsl:for-each>
180
181 </table>
182 </body>
183
184 </html>
185
186 </xsl:template>
187
188 </xsl:stylesheet>














```

Linie 30 til 55 er en forløkke, der løber hele XML-dokumentet igennem.

Linie 70 til 98 er endnu en forløkke, der løber hele XML-dokumentet igennem igen. Men indeni er der en if-sætning med en betingelse, der tjekker om bogen er udgivet efter 2007 (linie 71 til 96).

Linie 114 til 179 er endnu en forløkke, der løber hele XML-dokumentet igennem igen. Men indeni er der en choose, hvor man der her er specificeret farver på titel og årstal i forhold til udgivelsesåret (linie 122 til 144 og linie 152 til 173).

Resultatet af ovenstående kode skal gerne se sådan ud:

Alle bøger vist med forløkke						
Billede	ISBN	Titel	Forfatter	Udgiver	År	Pris
	978-0-672-33011-7	ASP.NET 3.5 Unleashed	Walther, Stephen	SAMS Publishing	2008	450.00 kr.
	987-1-4302-0997-3	Beginning XML with C# 2008 From Novice to Professional	Josh, Bipin	Apress	2008	325.00 kr.
	987-87-7055-813-6	Nu er det længe siden	Reintoft, Hanne	People's Press	2008	150.00 kr.
	987-0-9802858-4-0	The art & science of JavaScript	Adams, Cameron	SitePoint	2007	225.00 kr.
	978-1-59059-614-2	CSS Mastery: Advanced web Standard Solutions	Budd, Andy	Friends of ED	2006	369.00 kr.
Bøger nyere end 2007 (betingelse)						
Billede	ISBN	Titel	Forfatter	Udgiver	År	Pris
	978-0-672-33011-7	ASP.NET 3.5 Unleashed	Walther, Stephen	SAMS Publishing	2008	450.00 kr.
	987-1-4302-0997-3	Beginning XML with C# 2008 From Novice to Professional	Josh, Bipin	Apress	2008	325.00 kr.
	987-87-7055-813-6	Nu er det længe siden	Reintoft, Hanne	People's Press	2008	150.00 kr.
Bøger vist med farve i forhold til udgivelsesår (choose)						
Billede	ISBN	Titel	Forfatter	Udgiver	År	Pris
	978-0-672-33011-7	ASP.NET 3.5 Unleashed	Walther, Stephen	SAMS Publishing	2008	450.00 kr.
	987-1-4302-0997-3	Beginning XML with C# 2008 From Novice to Professional	Josh, Bipin	Apress	2008	325.00 kr.
	987-87-7055-813-6	Nu er det længe siden	Reintoft, Hanne	People's Press	2008	150.00 kr.
	987-0-9802858-4-0	The art & science of JavaScript	Adams, Cameron	SitePoint	2007	225.00 kr.
	978-1-59059-614-2	CSS Mastery: Advanced web Standard Solutions	Budd, Andy	Friends of ED	2006	369.00 kr.

I den første tabel vises alle bøger med en foreach-løkke.

I den anden tabel er der desuden sat en betingelse på, nemlig at der kun skal vises bøger, der er nyere end 2007.

I den tredje tabel er der sat betingelser på som farver forfatter og udgivelsesår i forhold til udgivelsesår med choose.

Man kan lave mange flere ting med XPath. F.eks. kan man kalde skabeloner, bruge funktioner og sortere. Dette er områder du selv kan udforske.

RSS-feed

RSS (Really simple Syndication) er en metode til at formidle links til websider via XML-filer. Disse filer indeholder beskrivelser af og links til webindhold. Ved at bruge forskellige desktopprogrammer kan brugere abonnere på disse filer og dermed holde sig up to date på forskellige områder. En meget populær brug af RSS er til nyheder med et kort nyhedsresume eller andre sites, hvor der hyppige opdateringer.

Man kan abonnere på RSS-feeds i enten dedikerede RSS-readers, i sin browser eller på sin mobiltelefon.

En RSS-Reader er en applikation, som man bruger til at se RSS feeds. På samme måde som en Email applikation kan samle mails fra forskellige email-konti, så kan en RSS Reader samle feeds fra forskellige sites. Der findes to typer Rss Readers. Web Readers og Desktop Readers. Forskellen mellem dem er minimal. Fordelen ved en web Reader er at alle ens abonnement-feeds gemmes på nettet, så de kan tilgås fra en hvilken som helst computer. Desktop Readers downloader abonnement-feeds til harddisken, så man kan se sine feeds selv om man ikke har internetadgang.

Hvis man åbner et RSS-feed direkte i browseren, vil det blive sat "anstændigt" op - se fx <http://poguemahone.dk/files/rss.xml>. Linket til RSS-feedet kan også copy-pastes ind i en dedikeret RSS-reader eller outlook.

Og endelig kan man tilføje et <link>-tag til sit <head>, så browseren er klar over at der er knyttet et feed til siden.

fx:

```
<link rel="alternate" type="application/rss+xml" title="Nyheder" href="nyheder.xml" />
```


Der findes flere forskellige RSS Readers: Google Reader (web), FeedDemon (desktop) og NewsFire (Mac) er populære readere, der er gratis.

RSS-filen indeholder som minimum 3 dele i XML-format. Dette er <title>, <link> og <description>, hvor title er titlen på Feeded, link er et link til ens websteds hovedside og description er en beskrivelse af RSS feedet . Dernæst kommer et element, der skal hedde <item>. I hver <item>-node indsættes den information, der skal vises for brugeren. Som minimum skal <item> indeholde <link> men man sagtens indsætte flere elementer. Ofte indeholder <item> noderne <title>, <link> og <description> som childnoder, hvor link er URL'en til websiden med den fulde beskrivelse af emnet. <title> er emnets titel/overskrift og <description> er en kort beskrivelse af emnet. De tre hovednoder plus <item>-noderne er omkranset af en <channel> node, som derved kommer til at indeholde site information. <rss version="2.0"> fortæller at det er et XML-dokument af typen rss.

<rss version="2.0"> starter alle rss-dokumenter

</rss> afslutter alle rss-dokumenter.

Filen gemmes som en xml-fil med endelsen .xml

Når man så har fået lavet en valid RSS-fil skal den være tilgængelig for brugerne, så de kan abonnere på den. En måde at gøre dette på er at reklamere for ens RSS-feed på ens website. Dette gøres ofte med knappen , som er et link til en eller flere RSS feeds, som brugere kan abonnere på.

En anden måde man kan gøre sine RSS feeds synlige på, er gennem en RSS syndikationservice. Et website som f. eks. SearchEngineWatch lister mange forskellige RSS og søgemaskiner, som tillader at man sender sin URL dertil.

Der findes en del forskellige desktopprogrammer, som man kan downloade, når man vil abonnere på RSS. Disse kaldes aggregatorer.

<channel>-elementet kan indeholde flere underelementer end de tre, der allerede er nævnt f. eks. (der findes flere):

- <category> Definerer en eller flere kategorier, som denne <channel> tilhører.
- <copyright> Hvem har copyrighten på dette RSS
- <image> Hvis man gerne vil have et billede på sin RSS

```
<image>
  <url>http://urlTilBillede</url>
  <title>Billedtitel/Titel på site</title>
  <link>Url til site</link>
</image>
```

<url>, <title> og <link> er obligatorisk, når man bruger <image>. <width>, <height> og <description> er valgfrie

Også i <item> findes der flere valgfrie underelementer, f. eks.

- <author></author> hvem har skrevet artiklen
- <comments></comments> En url-henvisning til en side med kommentarer
- <guid></guid> (Globally Unique Identifier) Bruges til at definere en unik identifikation til et <item>

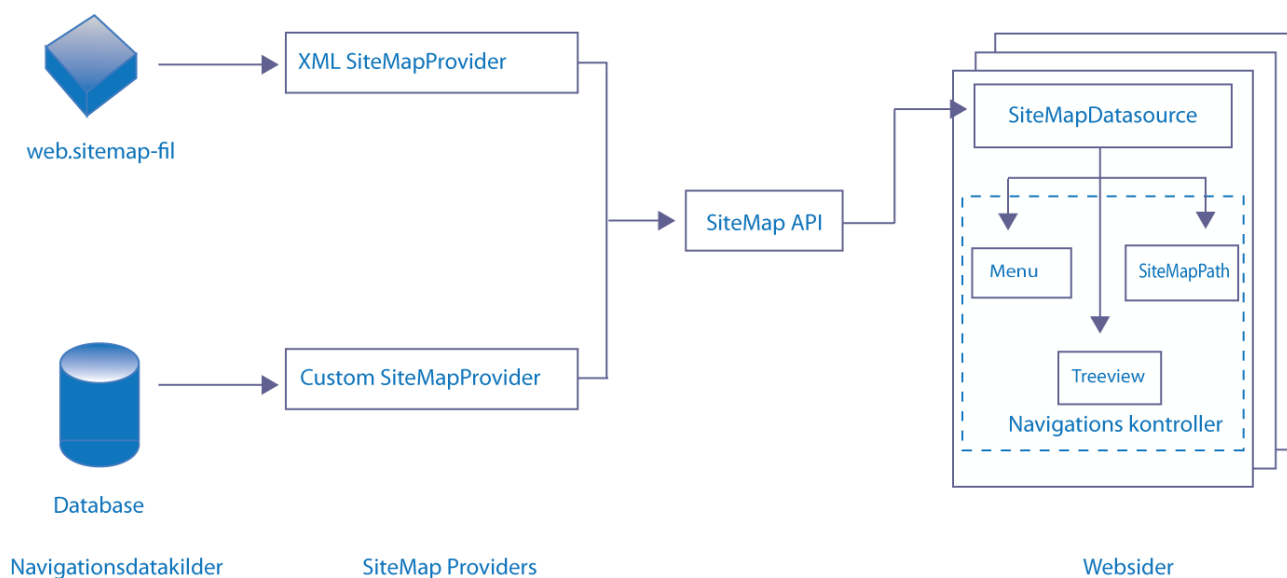
Websitenavigation ved hjælp af XML

Sitemaps

I ASP.NET findes der tre features, der kan forenkle navigationen på et site. Disse er:

- En måde, der definerer navigationsstrukturen på websitet. Denne del er XML SiteMap, der som default gemmes i en web.sitemap-fil.
- En nem måde at læse informationerne i en sitemap-fil og konvertere den til en objektmodel. Dette gøres ved hjælp af kontrollerne XmlSiteMapProvider og SiteMapDataSource.
- En nem måde at bruge sitemap informationerne på, så man kan vise brugerens aktuelle position og give brugeren mulighed for at navigere fra et sted til et andet. Dette gøres ved at binde menu, treeview, lister og breadcrumbs til SiteMapDataSourcen.

Man kan selvfølgelig tilpasse og udvide disse separat.



At definere et siteMap

SiteMapProvideren er startpunktet når man taler om sitemapbaseret navigation på et website. ASP.NET har en enkelt indbygget SiteMapProvider nemlig XmlSiteMapProvider. Den kan hente information til et sitemap fra en XML-fil. Skal man have data fra et andet format, f.eks. direkte fra en database, skal man lave sin egen SiteMapProvider eller finde en løsning fra en tredje part.

XmlSiteMapProvideren leder efter en fil med navnet Web.sitemap i roden af sitet. XmlSiteMapProviderens job er, som det er for alle SiteMapProvidere, at trække sitemapdata ud og oprette det tilhørende sitemap-object. Sitemap-objectet bliver derefter tilgængeligt for SitemapDataSourcen, som skal placeres på hver enkelt side, som bruger navigationen. SitemapDataSourcen giver sitemap-information til navigationskontrollerne. Disse er sidste trin i kæden.

Men før man begynder at skrive et sitemap, er der visse regler, som et ASP.NET-siteMap skal følge.

Med Visual Studio er nemt at oprette et siteMap. Der højreklikkes på mappen i roden af sitet. Vælg insert new item. Find filtypen web.sitemap. Vælg den og filen er oprettet. Man får her den grundlæggende struktur foræret, men de 5 regler er:

1: SiteMaps starter med <siteMap>-elementet

Alle web.sitemap-filer begynder med at erklære <siteMap> og slutter med </siteMap>. Al aktuel sitemap-information placeres mellem start- og slut-tag. Xmlns-attributten er nødvendig og skal skrives præcist som vist. For det fortæller ASP.NET at XML-filen anvender ASP.NET sitemap-standard.

```
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
```

2. Hver side er repræsenteret af et <siteMapNode>-element.

Dybest set definerer et sitemap hvordan websider er organiserede. For at indsætte en side i et sitemap tilføjer man <siteMapNode>-element plus nogle basis informationer. Dette er siden titel, en beskrivelse (er valgfri) og URL'en. Man tilføjer disse tre informationer ved at bruge attributterne title, description og url.

```
<siteMapNode url="~/Default.aspx" title="Forside" description="Velkommen" />
```

Bemærk at dette element slutter med />. Det er fordi det er et tomt element, hvor start- og sluttag er i et tag. Tomme elementer indeholder ALDRIG andre noder. Nedenfor er vist et fuldstændigt valid siteMap med præcis en side.

```
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode url="~/Default.aspx" title="Forside" description="Velkommen" />
</siteMap>
```

Bemærk at URL'en til en side begynder med ~/.

~/ tegnene repræsenterer rodmappen. Dvs. at ~/default.aspx peger på default.aspx i rodmappen. Det er ikke nødvendigt at skrive stier på denne måde, men det anbefales fordi man på den måde altid er sikker på at få vist den rigtige side.

Hvis man blot skriver stien default.aspx, vil ASP.NET lede efter default.aspx i den aktuelle mappe. Det kan give problemer, hvis man har en webapplikation med sider i flere mapper.

3. Et <siteMapNode>-element kan indeholde andre <siteMapNode>-elementer

Sitemaps består ikke af en simpel liste med sider. De deler i stedet sider ind i grupper. For at få vist dette i en siteMap-fil placerer man et <siteMapNode>-element inden i et andet.

```
1  <?xml version="1.0" encoding="utf-8" standalone="no"?>
2  <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
3    <siteMapNode url="~/Default.aspx" title="Forside" description="Velkommen">
4      <siteMapNode url="~/kontakt.aspx" title="Kontakt" description="Kontakt os">
5        <siteMapNode url="~/kontakt1.aspx" title="Kontakt1" description="Kontakt1" />
6      </siteMapNode>
7      <siteMapNode url="~/pizza.aspx" title="Pizzaer" description="Se pizzaer her">
8        <siteMapNode url="~/pizzaEmne.aspx?id=1" title="Børnepizza" description="Børnepizzaer"
9        />
10       <siteMapNode url="~/pizzaEmne.aspx?id=2" title="Voksenpizzaer"
11       description="Voksenpizzaer" />
12     </siteMapNode>
13     <siteMapNode url="~/retning.aspx" title="Retningslinier" description="retningslinier" />
14   </siteMap>
```

Der er ingen begrænsninger på, hvor mange niveauer man kan have, men generelt bør man kun have to eller tre niveauer for ikke at brugeren skal miste overblikket.

4. Ethvert siteMap begynder med en enkelt <siteMapNode>

Ethvert siteMap skal ALTID have en enkelt rodnode. Alle andre noder skal være inden i denne rodnode. Nedenstående kode er derfor IKKE korrekt.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode url="~/kontakt.aspx" title="Kontakt" description="Kontakt os" />
  <siteMapNode url="~/pizza.aspx" title="Pizzaer" description="Se pizzaer her" />
</siteMap>
```

I stedet skal koden være

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
<siteMapNode url="~/default.aspx" title="Home" description="Velkommen" />
  <siteMapNode url="~/kontakt.aspx" title="Kontakt" description="Kontakt os" />
  <siteMapNode url="~/pizza.aspx" title="Pizzaer" description="Se pizzaer her" />
</siteMapNode>
</siteMap>
```

5. Gentagne URL'er er IKKE tilladt

Man kan ikke have to siteMapNoder, der har den samme URL. Det kan være et problem, hvis man skal bruge det samme link på flere sider. Men der er indbygget i ASP.NET at man kan gemme sine noder i en collection, hvor hvert enkelt emne er indekseret via den unikke URL.

Men til trods for denne begrænsning kan godt lave URLs, der peger på den samme side, blot der er små forskelle, som f.eks. forskellige querystrenger.

Nedenfor er vist en eksempel med 7 noder

```
1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2 <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
3   <siteMapNode url="~/Default.aspx" title="Forside" description="Velkommen">
4     <siteMapNode url="info.aspx" title="Informaton" description="Se information om
- firmaet">
5       <siteMapNode url="om.aspx" title="Om os" description="Hvordan firmaet blev
- grundlagt" />
6     <siteMapNode url="kontakt.aspx" title="Kontakt os" description="Her kan du kontakte
- os" />
7   </siteMapNode>
8   <siteMapNode url="produkter.aspx" title="Produkter" description="Se vore produkter">
9     <siteMapNode url="produktEmne.aspx?id=1" title="Børnepizza" description="Se alle
- vore lækre børnepizzaer" />
10    <siteMapNode url="produktEmne.aspx?id=2" title="Voksenpizzaer" description="Se alle
- vore lækre pizzaer til voksne" />
11  </siteMapNode>
12 </siteMapNode>
13 </siteMap>
```

Når først web.sitemap-filen er lavet, er man klar til at bruge den. Lav alle de sider, der henvises til, også selv om de er uden indhold, for at se om navigationen virker.

Træk dernæst en SiteMapDataSource-kontrol ind på siden.

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
```

Det næste er at tilføje kontroller, der linker til SiteMapDataSourcen. De tre gængse er:

Treeview: Viser et træ af links, der er grupperet. Viser hele sitemappet på en gang.

Menu: Menuen viser en flerniveaus menu.

SiteMapPath: SiteMapPath er den enkleste af navigationskontrollerne. Den viser den fulde sti til den aktuelle side

Home > Produkter > Børnepizzaer

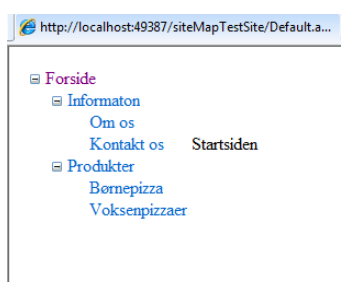
hvis man står på siden boernepizza.aspx

Med SiteMapPath kan man kun komme tilbage i navigations-hierarkiet og ikke begge veje, som man kan med treeview og menu.

For at binde til en kontrol til SiteMapDataSourcen skal man blot fortælle hvilken SiteMapDataSource, der skal bruges.

```
<asp:TreeView ID="TreeView1" runat="server" DataSourceID="SiteMapDataSource1" />
```

Så ordner ASP.NET resten, som man kan se her



SiteMap og masterpage

At bruge en masterpage, hvorpå man placerer en SiteMapDataSource og de navigationskontroller man skal bruge, vil være den bedste udnyttelse af disse funktionaliteter.

I et nyt site oprettes en masterpage og to sider med statisk indhold (default.aspx og produkter.aspx)

Opret også et Web.sitemap med følgende indhold

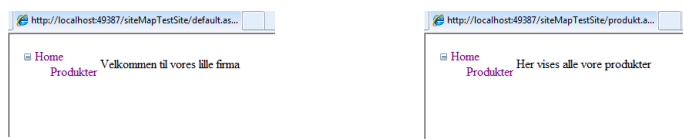
```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
3   <siteMapNode url="~/default.aspx" title="Home" description="Velkommen">
4     <siteMapNode url="~/produkt.aspx" title="Produkter" description="Her kan De se vore
-   produkter" />
5   </siteMapNode>
6 </siteMap>
```

På masteren kan man sætte følgende kode i body'en

```
1 <form id="form1" runat="server">
2   <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
3   <div>
4     <table>
5       <tr>
6         <td>
7           <asp:TreeView ID="TreeView1" runat="server" DataSourceID="SiteMapDataSource1">
8             </asp:TreeView>
9         </td>
10        <td>
11          <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
12            </asp:ContentPlaceHolder>
13        </td>
14      </tr>
15    </table>
16  </div>
17 </form>
```

På default.aspx skrives "Velkommen til vores lille firma"

På produkt.aspx skrives "Her vises alle vore produkter"



Treeview'et gør det muligt at navigere frem og tilbage mellem de to sider, som vist ovenover.

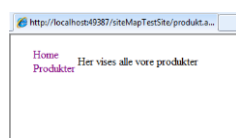
Det foregående eksempel viser siteMap-filens eksakte struktur. Men det er ikke altid det man ønsker. Ofte vil man ikke have vist rod-noden fordi den ligesom ligger et niveau før det man betragter som sine links i ens website.

Dette kan man undgå ved at man i SiteMapDataSourcens egenskaber sætter ShowStartingNode = "false". Så vises rodnoden ikke.

Web.sitemap-filen skal have følgende indhold

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
3   <siteMapNode url="" title="" description="">
4     <siteMapNode url="~/default.aspx" title="Home" description="Velkommen" />
5     <siteMapNode url="~/produkt.aspx" title="Produkter" description="Her kan De se vore
-   produkter" />
6   </siteMapNode>
7 </siteMap>
```

Og resultater er vist her



En anden mulighed er at vise en delmængde af sitemappet, hvor man viser sitemappet fra en aktuel node.

Default for SiteMapDataSource er at vise et fuldstændigt træ startende med rodnoden.

SiteMapDataSourcen har imidlertid mange egenskaber, der gør at man kan styre og style sin navigation.

Egenskab	Beskrivelse
ShowStartingNode	Hvis denne egenskab er false, så skjules den første node, dvs. rodnoden. Default er true
StartingNodeURL	Denne egenskab bruges til at ændre startnoden. Sæt denne værdi til URL'en på den node, der skal være den første node i navigationstræet. Denne værdi skal matche URL-attributten i sitemappet præcist. Hvis man f. eks. specificerer startnoden til at være ~/Home.aspx, så er den første node i træet Home og man vil kun se noder, der er under denne node.
StartFromCurrentNode	Hvis denne egenskab er true, så vil startsiden være den aktuelle node. Navigationstræet vil kun vise noder, der er efter denne. Brugeren vil kunne bevæge sig nedad i hierarkiet. Siden skal eksistere for at dette virker.
StartingNodeOffset	Denne egenskab bruges til at flytte startnoden op eller ned i hierarkiet. Der skal bruges en integer til at fortælle sitemappet, at det skal flytte sig fra startnoden og ned i træet (hvis tallet er positivt) eller op (hvis tallet er negativt). Det aktuelle resultat afhænger af hvordan man kombinerer denne egenskab med andre af sitemappets egenskaber. F. eks. hvis StartFromCurrentNode er false, skal man bruge et positivt tal for at bevæge sig fra startnoden ned mod den aktuelle node. Hvis StartFromcurrentNode er true, skal man bruge et negativt tal for at bevæge sig op i træet, væk fra den aktuelle node og op mod startnoden.

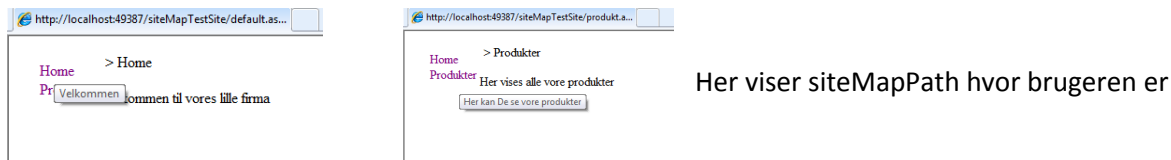
Egenskaber for en SiteMapDataSource

For at finde ud af, hvordan disse egenskaber virker, er det en god ide at afprøve dem

For at gøre tingene mere avancerede, er man ikke begrænset til kun at have én SiteMapDataSource på sin side. På den måde kan man bruge de to navigationskontroller til at vise forskellige dele af sitemap-hierarkiet (se eksempel i bilag 1).

Med SiteMapPathkontrollen giver man brugeren mulighed for at se, hvor vedkommende er.

SiteMapPathkontrollen laver brødkrumme-navigation. Det vil sige, at den viser brugeren hvor han er og brugeren har mulighed for at komme op i hierarkiet.



Man kan tilpasse SiteMapPath'en. En SiteMapPath behøver ikke have sine data fra en SiteMapDataSource. Der er nogle få egenskaber tilknyttet SiteMapPath

Egenskab	Beskrivelse
ShowToolTips	Denne egenskab skal sættes til false, hvis man ikke vil se beskrivelsesteksten, når brugeren holder musen hen over linket.
ParentLevelsDisplayed	Denne egenskab angiver det maksimale antal niveauer over den aktuelle node, som vil blive vist fra start. Som default er værdien -1. Så bliver alle niveauer vist.
RenderCurrentNodeAsLink	Hvis denne egenskab er sat til true, så bliver den del af siden, som indikerer den aktuelle side, lavet til et link. Som default er egenskaben sat til false, da brugeren jo allerede på den aktuelle side
PathDirection	Her har man to valgmuligheder. RootToCurrent, som er default og CurrentToRoot, som vender rækkefølgen af niveauer i stien
PathSeperator	Denne egenskab bruges til at vise, hvilken karakter, der skal bruges som adskillelsetegn mellem de enkelte niveauer i stien. Default er større end-tegnet (>). Et andet brugt tegn er kolon (:).

Egenskaber tilknyttet SiteMapPath

Man behøver ikke bruge en web.sitemap-fil for at få vist data via navigationskontrollerne. Man kan lave sin egen SiteMapDataSource eller man kan trække data ud direkte fra en database.

Eksempler

Neden for er to eksempler på anvendelse af XML sammen med .NET. Det første eksempel er et RSS feed, der henter data ud fra en database. Det andet eksempel er oprettelse, redigering og sletning i et XML-dokument.

RSS-feed

Opret et nyt website. Kald det RSSGenerator.

I sitet oprettes en xml-fil, rss.xml.

Den automatisk oprettede default beholdes.

Dernæst oprettes en database i Access. Kald den db.mdb. Den skal ligge i App_Data-mappen. I databasen oprettes en tabel, tblNews. Denne tabel skal have følgende felter:

Feltnavn	Datatype	Beskrivelse
fldID	autonummerering	
fldHeading	tekst	Overskrift
fldText	notat	Brødtekst
fldImage	tekst	Eventuelt billede
fldLinks	tekst	Eventuelle links (tekst-array)
fldDate	dato og klokkeslæt	Oprettelsestidspunkt

Dernæst oprettes dataAccess.vb i mappen App_Code

dataAccess.cs

```
1 using System.Web;
2 using System.Data;
3 using System.Data.OleDb;
4
5 /// <summary>
6 /// Summary description for dataAccess
7 /// </summary>
8 public class dataAccess
9 {
10
11     string DatabasePath = HttpContext.Current.Server.MapPath("App_Data/db.mdb");
12
13     public DataTable getData(string SQL)
14     {
15         DataSet objDS=new DataSet();
16         OleDbConnection objConn =new OleDbConnection("provider=Microsoft.Jet.OLEDB.4.0;Data
- Source=" + DatabasePath);
17
18         OleDbDataAdapter objDA = new OleDbDataAdapter(SQL, objConn);
19         objDA.Fill(objDS);
20         objConn.Close();
21
22         return objDS.Tables[0];
23     }
24
25     public void modifyData(string SQL)
26     {
27         OleDbConnection objConn =new OleDbConnection("provider=Microsoft.Jet.OLEDB.4.0;Data
- Source=" + DatabasePath);
28         OleDbCommand objCMD =new OleDbCommand(SQL, objConn);
29
30         objConn.Open();
31         objCMD.ExecuteNonQuery();
32         objConn.Close();
33     }
34 }
```

Klassefilen NewsFactory.cs oprettes også i mappen App_Code.

I klassen hentes namespace System.Data.

Dernæst oprettes der en funktion, som henter alle data ud fra tabellen tblNews.

NewsFactory.cs

```
1 using System.Data;
2
3 /// <summary>
4 /// Summary description for NewsFactory
5 /// </summary>
6 public class NewsFactory
7 {
8     dataAccess objData=new dataAccess();
9
10    public DataTable getLatest(int range)
11    {
12        return objData.getData("SELECT * FROM tblNews ORDER BY fldDate");
13    }
14 }
```

Default-sidens codebehind-del fylder data ind i xml-filen.

Default.aspx.cs

```
1 using System;
2 using System.Xml;
3 using System.Data;
4
5 public partial class _Default : System.Web.UI.Page
6 {
7
8     NewsFactory objNews = new NewsFactory();
9
10    protected void Page_Load(object sender, EventArgs e)
11    {
12        string doc = Request.PhysicalApplicationPath + "rss.xml";
13        XmlTextWriter rssWriter = new XmlTextWriter(doc, System.Text.Encoding.UTF8);
14
15        rssWriter.Formatting = Formatting.Indented;
16        rssWriter.WriteStartDocument(false);
17
18        rssWriter.WriteStartElement("rss");
19        rssWriter.WriteAttributeString("version", "2.0");
20
21        //start channel
22        rssWriter.WriteStartElement("channel");
23        rssWriter.WriteElementString("title", "RSS-feed");
24        rssWriter.WriteElementString("link", "http://www.etlink.dk");
25
26        foreach (DataRow item in objNews.getLatest(10).Rows)
27        {
28            //start item
29            rssWriter.WriteStartElement("item", "");
30            rssWriter.WriteElementString("title", item[1].ToString());
31            rssWriter.WriteElementString("link", item[4].ToString());
32            rssWriter.WriteElementString("description", item[2].ToString());
33            rssWriter.WriteElementString("datePub", item[5].ToString());
34            //slut item
35            rssWriter.WriteEndElement();
36        }
37
38        //Slut channel
39        rssWriter.WriteEndElement();
40
41        rssWriter.WriteEndElement();
42
43        rssWriter.Flush();
44        rssWriter.Close();
45    }
46 }
```

Linie 1 – 3: Her importeres namespacesne System.Xml, System.Xml.Schema, System.Data

Linie 8: Her laves en instans af NewsFactory.

Linie 10: Page_Load-rutinen er eventen, der lægger data ind i xml-filen.

Linie 12: Her erklæres en variabel doc som en streng og den tildeles stien til xml-filen.

Linie 13: Her erklæres en variabel rssWriter af typen XmlTextWriter. Der laves en instans af den, som tager to parametre, nemlig stien og tegnsættet

Linie 15: Formatting fortæller hvordan outputtet skal se ud. Her er valgt indrykning.

Linie 16: WriteStartDocument(**False**) er false, fordi xml-erklæringen ikke skal laves.

Linie 18 skriver det første element <rss>

Linie 19 skriver attributten "Version" med værdien "2.0" til elementet rss

Linie 22 skriver channel-elementet

Linie 23 skriver titlen på rss med to parametre, nemlig elementet og teksten

Linie 24 skriver linket til rss med to parametre, nemlig elementet og linket

Linie 26 – til 36 er en for-each løkke, der bruger de data, der er hentet ud fra tabellen tblNews.

Inde i selve løkken laves item-elementet med child-elementerne title, link, description, pubData.

Tallet i parentes efter item angiver feltet fra tabellen i databasen. Husk at den starter med 0. 1 er feltet fldOverskrift.

Linie 39 og 41 afslutter henholdsvis channel-elementet og rss-elementet.

Linie 43 flusher

Linie 44 closer streamen

Prøv at køre default-filen i browseren og vis derefter xml-filen i browseren. Du vil nu se dine nyheder som et rss-feed.

Oprette, redigere og slette i et XML-dokument

I det næste eksempel vil vi se på, hvordan man tilføjer noder, redigerer i noder og sletter noder i et XML-dokument.

Opret et nyt website. Kald det XML_CMS

Opret xml-filen boeger.xml i mappen App_Data

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <boegerRoot>
3   <boeger>
4     <author>ny test</author>
5     <title>gæt en ny bog3</title>
6     <content>diverse diverse</content>
7   </boeger>
8   <boeger>
9     <author>test testesen</author>
10    <title>en ny bog</title>
11    <content>Lidt af hvert</content>
12  </boeger>
13  <boeger>
14    <author>ny test</author>
15    <title>gæt en ny bog 4</title>
16    <content>diverse diverse</content>
17  </boeger>
18  <boeger>
19    <author>H. Jensen</author>
20    <title>En ny bog</title>
21    <content>Om lidt af hvert</content>
22  </boeger>
23  <boeger>
24    <author>hr. Jensen</author>
25    <title>En god bog</title>
26    <content>handler om lidt af hvert</content>
27  </boeger>
28  <boeger>
29    <author>Henrik</author>
30    <title>XML start</title>
31    <content>Om xml</content>
32  </boeger>
33 </boegerRoot>
```

XML-funktionaliteter

Forfatter:

Titel:

Indhold:

XML-data

6

	Forfatter	Titel	Indhold
Select	test testesen	en ny bog	Lidt af hvert
Select	hr. Jensen	En god bog	handler om lidt af hvert
Select	Henrik	XML start	Om xml
Select	torsdag	en titel	diverse diverse
Select	august	ny bog	lidt nyt
Select	Palle Bob	Bobs Erindringer	Mine erindringer må ikke glemmes

På default.aspx indsættes nedenstående kode. Der er tekstbokse til indtastning og et gridview til visning af data.

Default.aspx

```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
2
3 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
4 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
5
6 <html xmlns="http://www.w3.org/1999/xhtml">
7   <head runat="server">
8     <title></title>
9     <style type="text/css">
10     body {
11         margin-left: 20px;
12         margin-top: 20px;
13         margin-right: 20px;
14         margin-bottom: 0px;
15     }
16     a:link {
17         color: #0000FF;
18     }
19     a:visited {
```

```

19         color: #0000FF;
20     }
21     a:hover {
22         color: #0000FF;
23         text-decoration: none;
24     }
25     a:active {
26         color: #0000FF;
27     }
28     </style>
29 </head>
30 <body>
31     <form id="form1" runat="server">
32         <div>
33             <fieldset style="width:500px;">
34                 <legend>XML-funtionaliteter&nbsp;&nbsp;&nbsp;</legend>
35                 <table>
36                     <tr>
37                         <td colspan="2">Forfatter:</td><td colspan="2"><asp:TextBox ID="txtForfatter"
- runat="server" Width="231px"></asp:TextBox></td>
38                     </tr>
39                     <tr>
40                         <td colspan="2">Titel:</td><td colspan="2"><asp:TextBox ID="txtTitel" runat="server"
- Width="231px"></asp:TextBox></td>
41                     </tr>
42                     <tr>
43                         <td colspan="2">Indhold:</td><td colspan="2"><asp:TextBox ID="txtIndhold" runat="server"
- Width="231px"></asp:TextBox></td>
44                     </tr>
45                     <tr><td><asp:Button ID="btnOpret" runat="server" Text="Tilføj"
- CausesValidation="false" onclick="btnOpret_Click" /></td>
46                     <td><asp:Button ID="btnOpdater" runat="server" Text="Opdater"
- CausesValidation="false" onclick="btnOpdater_Click" /></td>
47                     <td><asp:Button ID="btnSlet" runat="server" Text="Slet" CausesValidation="false"
- onclick="btnSlet_Click" /></td>
48                     <td><asp:Button ID="btnClear" runat="server" Text="Clear" CausesValidation="false"
- onclick="btnClear_Click" /></td></tr></table>
49                 <br />
50             </fieldset>
51         <br />
52         <fieldset style="width:500px">
53             <legend>XML-data&nbsp;&nbsp;&nbsp;</legend>
54             <asp:Label ID="lblSelectedIndex" runat="server" />
55             <br />
56             <br />
57             <asp:GridView ID="GridView1" runat="server" CellPadding="4" ForeColor="#333333"
- GridLines="None" AutoGenerateColumns="false"
- onselectedindexchanged="GridView1_SelectedIndexChanged">
60                 <FooterStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" />
61                 <Columns>
62                     <asp:CommandField CancelText="Cancel" DeleteText="Delete" EditText="Modify"
- InsertText="Insert" NewText="New" SelectText="Select" ShowSelectButton="True"
- UpdateText="Update" />
63                     <asp:BoundField DataField="author" HeaderText="Forfatter" />
64                     <asp:BoundField DataField="title" HeaderText="Titel" />
65                     <asp:BoundField DataField="content" HeaderText="Indhold" />
66                 </Columns>
67                 <RowStyle BackColor="#E3EAE3" />
68                 <EditRowStyle BackColor="#7C6F57" />
69                 <SelectedRowStyle BackColor="#C5BBAF" Font-Bold="True" ForeColor="#333333" />
70                 <PagerStyle BackColor="#666666" ForeColor="White" HorizontalAlign="Center" />
71                 <HeaderStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" />
72                 <AlternatingRowStyle BackColor="White" />
73             </asp:GridView>
74             <br />
75         </fieldset>
76     </div>
77 </form>
78 </body>
79 </html>

```

Default.aspx.cs

```

1  using System;
2  using System.Data;
3  using System.Xml;
4
5  public partial class _Default : System.Web.UI.Page
6  {
7      protected void Page_Load(object sender, EventArgs e)
8      {
9          if (this.IsPostBack == false)
10         {
11             loadXMLdata();
12             this.lblSelectedIndex.Text = Convert.ToString(-1);
13             lblSelectedIndex.Visible = false;
14         }
15     }
16
17     private void loadXMLdata()
18     {
19         DataSet myDS = new DataSet();
20         myDS.ReadXml(Server.MapPath("App_Data/boeger.xml"));
21
22         if (myDS.Tables.Count != 0)
23         {
24             this.GridView1.DataSource = myDS;
25             this.GridView1.DataBind();
26         }
27     }
28
29     private void FindXmlData(int selectedIndex)
30     {
31         XmlDocument xmlDoc = new XmlDocument();
32
33         xmlDoc.Load(Server.MapPath("App_Data/boeger.xml"));
34         XmlNodeList xmlNodeList = xmlDoc.DocumentElement.ChildNodes;
35         XmlNode xmlNode = xmlNodeList.Item(selectedIndex);
36         this.txtForfatter.Text = xmlNode["author"].InnerText;
37         this.txtTitel.Text = xmlNode["title"].InnerText;
38         this.txtIndhold.Text = xmlNode["content"].InnerText;
39     }
40
41
42     protected void btnOpret_Click(object sender, EventArgs e)
43     {
44         XmlDocument xmlDoc = new XmlDocument();
45         xmlDoc.Load(Server.MapPath("App_Data/boeger.xml"));
46
47         XmlElement newElement = xmlDoc.CreateElement("boeger");
48         XmlElement xmlAuthor = xmlDoc.CreateElement("author");
49         XmlElement xmlTitle = xmlDoc.CreateElement("title");
50         XmlElement xmlContent = xmlDoc.CreateElement("content");
51
52         xmlAuthor.InnerText = this.txtForfatter.Text;
53         xmlTitle.InnerText = this.txtTitel.Text;
54         xmlContent.InnerText = this.txtIndhold.Text;
55
56         newElement.AppendChild(xmlAuthor);
57         newElement.AppendChild(xmlTitle);
58         newElement.AppendChild(xmlContent);
59
60         xmlDoc.DocumentElement.AppendChild(newElement);
61         xmlDoc.Save(Server.MapPath("App_data/boeger.xml"));
62
63         loadXMLdata();
64     }
65
66     protected void btnOpdater_Click(object sender, EventArgs e)
67     {
68         XmlDocument xmlDoc = new XmlDocument();
69         xmlDoc.Load(Server.MapPath("App_Data/boeger.xml"));
70
71         if (Int32.Parse(this.lblSelectedIndex.Text) < 0)
72     {

```

```

73         ClientScript.RegisterClientScriptBlock(this.GetType(), "alertmessage",
-         "<script>alert('Vælg venligst en bog')</script>");
74     }
75     else
76     {
77         XmlNode xmlNode =
-         xmlDoc.DocumentElement.ChildNodes.Item(Int32.Parse(this.lblSelectedIndex.Text));
78
79         xmlNode["author"].InnerText = this.txtForfatter.Text.Trim();
80         xmlNode["title"].InnerText = this.txtTitel.Text.Trim();
81         xmlNode["content"].InnerText = this.txtIndhold.Text.Trim();
82
83         xmlDoc.Save(Server.MapPath("App_Data/boeger.xml"));
84     }
85
86     loadXMLdata();
87 }
88
89 protected void btnSlet_Click(object sender, EventArgs e)
90 {
91     XmlDocument xmlDoc= new XmlDocument();
92     xmlDoc.Load(Server.MapPath("App_Data/boeger.xml"));
93
94     XmlNode xmlNode =
-     xmlDoc.DocumentElement.ChildNodes.Item(Int32.Parse(this.lblSelectedIndex.Text));
95     xmlNode.ParentNode.RemoveChild(xmlNode);
96     xmlDoc.Save(Server.MapPath("App_Data/boeger.xml"));
97
98     loadXMLdata();
99     this.txtForfatter.Text = "";
100    this.txtTitel.Text = "";
101    this.txtIndhold.Text = "";
102 }
103
104 protected void btnClear_Click(object sender, EventArgs e)
105 {
106     this.txtForfatter.Text = "";
107     this.txtIndhold.Text = "";
108     this.txtTitel.Text = "";
109     lblSelectedIndex.Text =Convert.ToString(-1);
110 }
111
112 protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
113 {
114     this.lblSelectedIndex.Visible = true;
115     this.lblSelectedIndex.Text = this.GridView1.SelectedIndex.ToString();
116     FindXmlData(Int32.Parse(this.lblSelectedIndex.Text));
117 }
118 }

```

Linie 2 – linie 3: Her importeres namespacesne System.xml og System.Data

Linie 7 – linie 15 er Page_load rutinen.

Linie 7 -14 loader xml-filen via rutinen loadXmlData(), hvis siden ikke tidligere har kørt

Linie 17 – 27 er rutinen loadXmlData().

Linie 19: Her oprettes et Dataset, som i linie 20 kommer til at indeholde data fra xml-filen

Linie 22 – 26: Hvis der er data i datasættes skal det vises i griedviewet

Linie 29 – 40: Her skrives rutinen FindXmlData(int selectedIndex), som finder de data, der passer til det valgte index

Linie 42 – 64 er knappen, der opretter en ny bog

Linie 66 – 87 er knappen, der opdaterer ændringerne

Linie 89 – 102 er knappen, der sletter en bog

Linie 104 – 110 er knappen, der tømmer tekstboksene

Linie 112 – 117: Når man vælger en bog i griedviewet via select, så vises dets index i en label og rutinen FindXmlData([int](#) selectedIndex) køres

Eksempel med SiteMap

Dette eksempel laver en web.sitemap-fil ud fra data i en database. Denne fil bruges til at vise menuen vis en treeviewkontrol og breadcrumbs .

Opret en ny database. Kald den dbPizza.

I databasen skal der være følgende tabeller

tblMenu1

Feltnavn	Datatype
fIdID	Integer, autonummering
fIdUnderID	Integer
fIdNavn	VarChar
fIdLink	VarChar
fIdBeskrivelse	Text

tblMenu1 - Data

fIdID	fIdUnderMenu	fIdNavn	fIdLink	fIdBeskrivelse
1	0	Pizzaer	pizza.aspx	Se pizzaer her
2	0	Kontakt	kontakt.aspx	Kontakt os
3	0	Retningslinier	retning.aspx	retningslinier
4	1	Børnepizza	pizzaEmne.aspx?id=1	Børnepizzaer
5	1	Voksenpizzaer	pizzaEmne.aspx?id=2	Voksenpizzaer
6	2	Kontakt1	kontakt1.aspx	Kontakt1
7	0	Forside	Default.aspx	Velkommen

tblPizza

Feltnavn	Datatype
fIdID	Integer, autonummering
fIdKatId	Integer
fIdOverskrift	VarChar
fIdBeskrivelse	Text
fIdPris	Float
fIdIngredienser	MediumText
fIdBillede	VarChar

tblPizza - data

fIdID	fIdKatId	fIdOverskrift	fIdBeskrivelse	fIdPris	fIdIngredienser	fIdBillede
1	2	Pizza Quattro	En pizza med meget fyld. En solid bund, hvorpå der er tomat og meget andet sp...	60	Oksekød, løg, rød peber, hvidløg, oregano og ost	pizza1.jpg
2	2	Pizza Tomato	En anderledes pizza med soltørrede tomater og lufttørret skinke	58	lufttørret skinke, soltørrede tomater, hvidløg, oregano og ost	pizza2.jpg
3	1	Pizza Olfert	En rigtig god børnepizza	38	Bacon, skinke, pølser, tomater, ost	pizza3.jpg
4	1	Pizza Cirkeline	En spændende børnepizza med forskelligt fyld	35	Cocktailpølse, tomater, spegepølse, ost	pizza4.jpg
5	3	Pizza Vegetarianeo	En god pizza for vegetaren	55	Tomat, løg, rød peber, ost	pizza5.jpg
6	3	Pizza al dente	Endnu en god pizza for vegetaren og de, der vil prøve noget andet	49	Tomat, oliven, aubergine, ost	pizza6.jpg
7	4	Pizza pescare	En spændende pizza med alt godt fra havet	65	Tomat, rejer, muslinger, ost, ananas, oregano	pizza7.jpg
8	4	Pizza Grenaa	En pizza med havets goder	65	Tomat, blæksprutte, krabbe, hummer, ost, oregano, hvidløg	pizza8.jpg

Opret et nyt website. Kald det SiteMapPizza. Husk at hente referencen til MySQL, hvis du arbejder med en MySQLServer.

I App_Code-mappen oprettes en dataAccess-fil

dbDataAccess.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Configuration;
6 using MySql.Data.MySqlClient;
7 using System.Data;
8
9 /// <summary>
10 /// Summary description for dbDataAccess
11 /// </summary>
12 public class dbDataAccess
13 {
14     string strDataConn = ConfigurationManager.ConnectionStrings["MySQLConn"].ConnectionString;
15
16     public DataTable GetData(MySqlCommand CMD)
17     {
18         //Ny forbindelse fra Connectionstring'en
19         MySqlConnection objConn = new MySqlConnection(strDataConn);
20
21         //Ny MySqlDataAdapter
22         MySqlDataAdapter objDA = new MySqlDataAdapter();
23
24         //Nyt Dataset
25         DataSet objDS = new DataSet();
26
27         //Angivelse af MySqlCommando'ens forbindelse
28         CMD.Connection = objConn;
29
30         //Angivelse af DataAdapterens Select-kommando (da GetData kun bruges ved SELECT-
31         - statements)
32         objDA.SelectCommand = CMD;
33         objDA.Fill(objDS);
34
35         return objDS.Tables[0];
36     }
37
38     public void ModifyData(MySqlCommand CMD)
39     {
40         //Ny forbindelse fra Connectionstring'en
41         MySqlConnection objConn = new MySqlConnection(strDataConn);
42
43         //Kommandoens forbindelse
44         CMD.Connection = objConn;
45
46         //Åbn forbindelsen
47         objConn.Open();
48
49         //Udfør SQL-kommandoen (en UPDATE/INSERT/DELETE) der ikke giver et resultatsæt men
50         - derimod en tal, nemlig antallet af påvirkede rækker
51         CMD.ExecuteNonQuery();
52         objConn.Close();
53     }
54 }
```

Her kommer ingen forklaring for det er den samme fil, som man ellers bruger i sine andre projekter.

Opret en ny klassefil, sitemapFac.cs

sitemapFac.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Data;
6 using MySql.Data.MySqlClient;
7
8 /// <summary>
9 /// Summary description for sitemapFac
10 /// </summary>
11 public class sitemapFac
12 {
13     dbDataAccess objData = new dbDataAccess();
14     public DataTable getMenuMaster()
15     {
16         string strSQL = "select * from tblMenu1 where fldUndermenu=0 order by fldNavn asc";
17         MySqlCommand objCMD = new MySqlCommand(strSQL);
18
19         return objData.GetData(objCMD);
20     }
21
22
23     public DataTable getChildMenu(int ID)
24     {
25         string strSQL = "select * from tblMenu1 where fldUndermenu=?id";
26         MySqlCommand objCMD = new MySqlCommand(strSQL);
27         objCMD.Parameters.Add("?id", MySqlDbType.Int32).Value = ID;
28         return objData.GetData(objCMD);
29     }
30 }
```

Linie 14: Her oprettes en instans af dbDataAccess-filen

Linie 15 - 20: Der oprettes en funktion, der trækker data ud fra tabellen tblMenu1. Disse data skal danne overmenupunkterne i navigationen

Linie 23 - 29: Her oprettes en funktion, der henter undermenupunkterne til de respektive overmenupunkter

Opret endnu en ny klassefil, PizzaFactory.cs

PizzaFactory.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Data;
6 using MySql.Data.MySqlClient;
7
8 /// <summary>
9 /// Summary description for PizzaFactory
10 /// </summary>
11 public class PizzaFactory
12 {
13
14     dbDataAccess objData = new dbDataAccess();
15
16     public DataTable getTre()
17     {
```

```

18         string strSQL = "Select fldID, fldOverskrift, fldPris, fldBillede from tblPizza order
- by fldID desc limit 3";
19         MySqlCommand objCMD = new MySqlCommand(strSQL);
20         return objData.GetData(objCMD);
21     }
22
23     public DataTable getAllePizzaer()
24     {
25         string strSQL = "Select fldID, fldKatID, fldOverskrift, fldBeskrivelse, fldPris,
- fldBillede From tblPizza order by fldOverskrift desc";
26         MySqlCommand objCMD = new MySqlCommand(strSQL);
27
28         return objData.GetData(objCMD);
29     }
30
31     public DataTable getPizzaerEfterEmne(int id)
32     {
-         string strSQL = "Select fldID, fldKatID, fldOverskrift, fldBeskrivelse, fldPris,
33 fldBillede From tblPizza where fldKatID=?ID order by fldOverskrift desc";
34         MySqlCommand objCMD = new MySqlCommand(strSQL);
35         objCMD.Parameters.Add("ID", MySqlDbType.Int32).Value = id;
36
37         return objData.GetData(objCMD);
38     }
39
40     public DataRow getUdvPizza(int id)
41     {
-         string strSQL = "Select fldID, fldOverskrift, fldBeskrivelse, fldPris, fldBillede From
42 tblPizza where fldID=?ID";
43         MySqlCommand objCMD = new MySqlCommand(strSQL);
44         objCMD.Parameters.Add("ID", MySqlDbType.Int32).Value = id;
45
46         return objData.GetData(objCMD).Rows[0];
47     }
48 }

```

Linie 14 opretter en instans af dbDataAccess

Linie 16 – 21: Der oprettes en funktion, getTre() som DataTable, der henter data ud til de tre nyeste pizzaer

Linie 23 – 29: Her oprettes en funktion, getAllePizzaer() som DataTable, der data ud til alle pizzaer

Linie 31 – 38: Her oprettes en funktion, getPizzaerEfterEmne(int id) som DataTable, der henter pizzaer ud efter valgt emne

Linie 40 – 48: Her oprettes en funtion, getUdvPizza(int id) som DataRow, der henter den valgte pizza

Slet defaultsiden og opret en masterpage.

masterPage.Master

```

1  <%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
2  Inherits="MasterPage" %>
3
4  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
- "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
5
6  <html xmlns="http://www.w3.org/1999/xhtml">
7  <head runat="server">
8      <title>SiteMapPath</title>
9      <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
10     <asp:ContentPlaceHolder id="head" runat="server">
11     </asp:ContentPlaceHolder>
12 </head>
13 <body>
14     <form id="form1" runat="server">

```

```

15 <div id="main">
16 <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" ShowStartingNode="false" />
17 <table id="tblMain">
18 <tr>
19 <td id="banner" colspan="3">Et banner</td>
20 </tr>
21 <tr>
22 <td id="menu">Menu<br /><br />
23
24 <asp:TreeView ID="TreeView1" runat="server" DataSourceID="SiteMapDataSource1"
- NodeIndent="0" ShowExpandCollapse="false">
25 <HoverNodeStyle Font-Underline="False" ForeColor="#5555DD" />
26 <NodeStyle NodeSpacing="0px" VerticalPadding="0px" Font-Names="Verdana"
- Font-Size="10px" ForeColor="Navy" HorizontalPadding="0px"
- Width="150px" />
27 <ParentNodeStyle Font-Bold="False" />
28 <SelectedNodeStyle Font-Underline="True" ForeColor="#5555DD"
- HorizontalPadding="0px" VerticalPadding="0px" Font-Size="10px" />
29 </asp:TreeView>
30 </td>
31 <td id="indhold">
32 <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
33
34 </asp:ContentPlaceHolder></td><td id="hoejre">
35 <asp:Literal ID="litTilfaeldig" runat="server"></asp:Literal></td>
36 </tr>
37 <tr>
38 <td id="bund" colspan="3">Bundoplysninger</td>
39 </tr>
40 </table>
41
42 </div>
43
44 </form>
45 </body>
46 </html>

```

Linie 16: Her trækkes SiteMapDataSourcen ind. ShowStartingNode sættes til false

Linie 17 – 40: Sidens opbygning

Linie 24 – 29: Treeview med henvisning til den rigtige DataSource via DataSourceID. Der er sat forskellige egenskaber på treeviewet.

På codebehind-filen skrives følgende kode

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7 using System.Data;
8 using System.Xml;
9
10 public partial class MasterPage : System.Web.UI.MasterPage
11 {
12     sitemapFac objMenu = new sitemapFac();
13
14     protected void Page_Load(object sender, EventArgs e)
15     {
16         #region Sitemap
17
18         XmlTextWriter xmlMyWriter = new XmlTextWriter(Request.PhysicalApplicationPath +
- "Web.sitemap", System.Text.Encoding.UTF8);
19
20         xmlMyWriter.Formatting = Formatting.Indented;
21         xmlMyWriter.WriteStartDocument(false);
22

```

```

23         //Hovedelement
24
25         xmlMyWriter.WriteStartElement("siteMap",
- "http://schemas.microsoft.com/AspNet/SiteMap-File-1.0");
26
27         //Indsætter toplevelmenu
28         DataTable dt = null;
29         dt = objMenu.getMenuMaster();
30
31         //Anden undernode
32         xmlMyWriter.WriteStartElement("siteMapNode");
33         xmlMyWriter.WriteAttributeString("url", "");
34         xmlMyWriter.WriteAttributeString("title", "");
35         xmlMyWriter.WriteAttributeString("description", "");
36         //xmlMyWriter.WriteAttributeString("url", "~/Default.aspx");
37         xmlMyWriter.WriteAttributeString("title", "Forside");
38         xmlMyWriter.WriteAttributeString("description", "Velkommen");
39         //xmlMyWriter.WriteEndElement();
40         // mastermenu
41         foreach (DataRow row in dt.Rows)
42         {
43             xmlMyWriter.WriteStartElement("siteMapNode");
44             xmlMyWriter.WriteAttributeString("url", "" + row["fldLink"] + "");
45             xmlMyWriter.WriteAttributeString("title", "" + row["fldNavn"] + "");
46             xmlMyWriter.WriteAttributeString("description", "" + row["fldBeskrivelse"] +
- "");
47
48             int varID = Int32.Parse(row["fldID"].ToString());
49             DataTable dtChild1 = objMenu.getChildMenu(varID);
50
51             //Level1
52             if (dtChild1.Rows.Count > 0)
53             {
54
55                 foreach (DataRow RowChild1 in dtChild1.Rows)
56                 {
57                     xmlMyWriter.WriteStartElement("siteMapNode");
58                     xmlMyWriter.WriteAttributeString("url", "" + RowChild1["fldLink"] +
- "");
59                     xmlMyWriter.WriteAttributeString("title", "" + RowChild1["fldNavn"] +
- "");
60                     xmlMyWriter.WriteAttributeString("description", "" +
- RowChild1["fldBeskrivelse"] + "");
61
62                     //Level2
63                     int varID2 = Int32.Parse(RowChild1["fldID"].ToString());
64                     DataTable dtChild2 = objMenu.getChildMenu(varID2);
65                     if (dtChild2.Rows.Count > 0)
66                     {
67                         foreach (DataRow RowChild2 in dtChild2.Rows)
68                         {
69                             xmlMyWriter.WriteStartElement("siteMapNode");
70                             xmlMyWriter.WriteAttributeString("url", "" +
- RowChild2["fldLink"] + "");
71                             xmlMyWriter.WriteAttributeString("title", "" +
- RowChild2["fldNavn"] + "");
72                             xmlMyWriter.WriteAttributeString("description", "" +
- RowChild2["fldBeskrivelse"] + "");
73
74                             //Level3
75                             int varID3 = Int32.Parse(RowChild2["fldID"].ToString());
76                             DataTable dtChild3 = objMenu.getChildMenu(varID3);
77                             if (dtChild3.Rows.Count > 0)
78                             {
79                                 foreach (DataRow RowChild3 in dtChild3.Rows)
80                                 {
81                                     xmlMyWriter.WriteStartElement("siteMapNode");
82                                     xmlMyWriter.WriteAttributeString("url", "" +
- RowChild3["fldLink"] + "");
83                                     xmlMyWriter.WriteAttributeString("title", "" +
- RowChild3["fldNavn"] + "");
84                                     xmlMyWriter.WriteAttributeString("description", "" +
- RowChild3["fldBeskrivelse"] + "");
85

```

```

86         xmlMyWriter.WriteEndElement();
87     }
88 }
89     xmlMyWriter.WriteEndElement();
90 }
91 }
92     xmlMyWriter.WriteEndElement();
93 }
94 }
95 //Slutter
96 xmlMyWriter.WriteEndElement();
97
98 }
99
100 //Slutter toplevel
101 xmlMyWriter.WriteEndElement();
102
103 //Slutter dokument
104 xmlMyWriter.WriteEndDocument();
105
106 //Udfør alle "pending operations" og luk xmlMyWriterobjektet
107 xmlMyWriter.Flush();
108 xmlMyWriter.Close();
109
110 #endregion
111
112 //Udskrivning af de tre pizzaer i højre side
113 PizzaFactory objTre = new PizzaFactory();
114 foreach (DataRow row in objTre.getTre().Rows)
115 {
116     litTilfaeldig.Text += "<table class='tre'><tr><td><b>" +
117 row["fldOverskrift"] + "</b></td></tr><tr><td><a href='\"pizzaEmne.aspx?udvid=" + row["fldID"]
118 + "\"><img src='./Billeder/PizzaBilleder/' + row["fldBillede"] + "' border='0' width='25px'
119 /></a><br />Pris: " + row["fldPris"] + " kr.</td></tr></table><br /><br />";
120 }
121 }
122 }

```

Linie 12: Der laves en instans af siteMapFac()

Linie 16 – 110: Her dannes web.siteMap-filen

Linie 18: Her oprettes en XmlTextWriter, som skal have to parametre, nemlig filen der skal skrives i (Web.siteMap) og det tegnsæt der skal bruges

Linie 20: Fortæller at noderne skal skrives med indryk

Linie 21: Fortæller vi at man ikke skal skrive startnoden. Den blev skrevet da dokumentet blev oprettet.

Linie 25: Skriver den første siteMapnode med de rette attributter og værdier.

Linie 28: Her oprettes en datatable, der sættes lig null.

Linie 29: Her lægges de data, som man har hentet via funktionen getMenuMaster, over i datatablen.

Linie 32 – 35: Her begynder man at skrive den 2. undernode. Her er der ingen værdier i attributterne, da vi senere vælger at skjule den første node.

Linie 36 – 40: Koden her er udkommenteret, men er den kode, der skal bruges, hvis der skal være faste værdier til attributterne i koden på linie 32 – 35.

Linie 41 – 94: Er en for-each løkke, hvor mastermenuen laves først og dernæst eventuelle underpunkter.

Linie 42 – 46: Her skrives "over" siteMapNoder med tilhørende url, title og description. Værdierne er trukket fra databasen.

Linie 48: Her laves en variabel, varID, af typen integer. Den skal indeholde id-værdien fra "over"-punkterne.

Linie 49: Der laves en datatable, dtchild1, med de værdier, der er hentet via funktionen getChildMenu(varID).

Linie 52: Hvis der er værdier i datatablen er Rows.Count > 0 og derfor skal der oprettes menupunkter.

Linie 55 - 91: Her laves en for-each løkke, der kan lave menupunkter til niveau 1. Der laves en siteMapNode med de tilhørende attributter og værdier hentet fra databasen.

Linie 63: Her oprettes endnu en variabel, varID2, af typen integer. Den får værdien som ligger i Rowchild1["fldID"].

Linie 64: Der laves en datatable, dtchild2, med de værdier, der er hentet via funktionen getChildMenu(varID2).

Linie 65: Hvis der er værdier i datatablen er Rows.Count > 0 og derfor skal der oprettes menupunkter.

Linie 67 – 72: Her laves en for-each løkke, der kan lave menupunkter til niveau 2. Der laves en siteMapNode med de tilhørende attributter og værdier hentet fra databasen.

Linie 75: Her oprettes endnu en variabel, varID3, af typen integer. Den får værdien som ligger i Rowchild2["fldID"].

Linie 76: Der laves en datatable, dtchild3, med de værdier, der er hentet via funktionen getChildMenu(varID3).

Linie 77: Hvis der er værdier i datatablen er Rows.Count > 0 og derfor skal der oprettes menupunkter.

Linie 79 – 87: Her laves en for-each løkke, der kan lave menupunkter til niveau 3. Der laves en siteMapNode med de tilhørende attributter og værdier hentet fra databasen.

Linie 88: Her afsluttes den if-sætning, der blev begyndt i linie 77.

Linie 89: Her skrives </sitemapNode>-elementet for niveau 3.

Linie 90: Her slutter den for-each løkken, der blev begyndt i linie 68.

Linie 91: Her slutter den if-sætning, der blev startet i linie 65.

Linie 92: Her skrives </sitemapNode>-elementet for niveau 2.

Linie 93: Her slutter den for-each løkke, der blev begyndt i linie 55.

Linie 94: Her slutter den if-sætning, der blev startet i linie 52.

Linie 95: Her skrives </sitemapNode>-elementet for niveau 1.

Linie 98: Her slutter for-each løkken, der blev startet i linie 41.

Linie 101: Her skrives </siteMapNode> til top-elementet.

Linie 104: Her skrives slutnoden for dokumentet.

Linie 107: Der flushes.

Linie 108: Xmlwriteren lukkes.

Linie 113 – 120: Her skrives de 3 nyeste pizzaer i højre side

Sæt selv lidt kode på default.aspx, kontakt.aspx og kontakt1.aspx. Kontakt1.aspx er lavet for at demonstrere sitmapfunktionaliteten.

Opret en ny side, Pizza.aspx

```
1 <%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
- CodeFile="pizza.aspx.cs" Inherits="pizza" %>
2
3 <asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
4 </asp:Content>
5 <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
6 <asp:SiteMapPath ID="SiteMapPath1" runat="server">
7 <PathSeparatorTemplate>
8 <asp:Image ID="Image1" runat="server" ImageUrl="~/Grafik/Pil_lille.png" />
9 </PathSeparatorTemplate>
10 </asp:SiteMapPath>
11 <asp:Literal ID="litOverskrift" runat="server"></asp:Literal><hr /><br />
12 <asp:Literal ID="litVis" runat="server"></asp:Literal>
13 </asp:Content>
```

På codebehind skrives koden

Pizza.aspx.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7 using System.Data;
8
9 public partial class pizza : System.Web.UI.Page
10 {
11     PizzaFactory objPizza = new PizzaFactory();
12     protected void Page_Load(object sender, EventArgs e)
13     {
14         foreach (DataRow row in objPizza.getAllePizzaer().Rows)
15         {
16             litVis.Text += "<b>" + row["fldOverskrift"] + "</b><hr />";
17             litVis.Text += "<table><tr><td><img src='Billeder/PizzaBilleder/' +
18 row["fldBillede"] + "' style='float:left;padding-right:3px;padding-bottom:3px;' />" +
- row["fldBeskrivelse"] + "<br /><br />";
-             litVis.Text += "Pris: " + row["fldPris"] + ", - kr.<br /><br />";
19             litVis.Text += "<a href='pizzaEmne.aspx?udvid=" + row["fldID"] + "'><i>Se
20 yderligere oplysninger</i></a>";
-             litVis.Text += "</td></tr></table><br />";
21         }
22     }
23 }
24 }
```

Opret endnu en side, pizzaEmne.aspx

pizzaEmne.aspx

```
1 <%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
- CodeFile="pizzaEmne.aspx.cs" Inherits="pizzaEmne" %>
2
3 <asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
4 </asp:Content>
5 <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
6 <asp:SiteMapPath ID="SiteMapPath1" runat="server">
7 <PathSeparatorTemplate>
8 <asp:Image ID="Image1" runat="server" ImageUrl="~/Grafik/Pil_lille.png" />
9 </PathSeparatorTemplate>
10 </asp:SiteMapPath>
11 <br />
12 <asp:Literal ID="litVis" runat="server"></asp:Literal>
13 </asp:Content>
```

På codebehind skrives

pizzaEmne.aspx.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7 using System.Data;
8
9 public partial class pizzaEmne : System.Web.UI.Page
10 {
11     protected void Page_Load(object sender, EventArgs e)
12     {
13         PizzaFactory objPizza = new PizzaFactory();
14
15         if (string.IsNullOrEmpty(Request.QueryString["udvid"]))
16         {
17             foreach (DataRow row in
- objPizza.getPizzaerAfterEmne(int.Parse(Request.QueryString["id"])).Rows)
18             {
19                 litVis.Text += "<br /><hr />";
20                 litVis.Text += "<b>" + row["fldOverskrift"] + "</b><br />";
21                 litVis.Text += "<table><tr><td><img src='Billeder/PizzaBilleder/' +
- row["fldBillede"] + "' style='float:left;padding-right:3px;padding-bottom:3px;' /> +
- row["fldBeskrivelse"] + "<br /><br />";
22                 litVis.Text += "Pris: " + row["fldPris"] + ", - kr.<br /><br />";
23                 litVis.Text += "<a href='pizzaEmne.aspx?udvid=" + row["fldID"] + "'><i>Se
- yderligere oplysninger</i></a>";
24                 litVis.Text += "</td></tr></table><br />";
25             }
26         }
27         else
28         {
29             string strID = Request.QueryString["udvid"];
30
31             DataRow objUdvPizza = objPizza.getUdvPizza(int.Parse(strID));
32             litVis.Text += "<b>" + objUdvPizza["fldOverskrift"].ToString() + "</b><br />";
33             litVis.Text += "<table><tr><td><img src='Billeder/PizzaBilleder/' +
- objUdvPizza["fldBillede"].ToString() + "' style='float:left;padding-right:3px;padding-
- bottom:3px;' /> + objUdvPizza["fldBeskrivelse"].ToString() + "<br /><br />";
34             litVis.Text += "Pris: " + objUdvPizza["fldPris"].ToString() + ", - kr.<br /><br />";
35             litVis.Text += "</td></tr></table><br />";
36         }
37     }
38 }
```

Bilag

Eksempel med 2 SiteMapDataSourcer og 2 navigationskontroller.

Web.siteMap

```
1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2 <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
3   <siteMapNode url="~/Default.aspx" title="Forside" description="Velkommen">
4     <siteMapNode url="~/info.aspx" title="Information" description="Se information om firmaet">
5       <siteMapNode url="~/om.aspx" title="Om os" description="Hvordan firmaet blev grundlagt" />
6       <siteMapNode url="~/kontakt.aspx" title="Kontakt os" description="Her kan du kontakte os" />
7     </siteMapNode>
8     <siteMapNode url="~/produkt.aspx" title="Produkter" description="Se vore produkter">
9       <siteMapNode url="~/produktEmne.aspx?id=1" title="Børnepizza" description="Se alle vore
- lækre børnepizzaer" />
10      <siteMapNode url="~/produktEmne.aspx?id=2" title="Voksenpizzaer" description="Se alle vore
- lækre pizzaer til voksne" />
11    </siteMapNode>
12  </siteMapNode>
13 </siteMap>
```

MasterPage.master

```
1 <%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
- Inherits="MasterPage" %>
2
3 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
- "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7   <title></title>
8   <asp:ContentPlaceHolder id="head" runat="server">
9   </asp:ContentPlaceHolder>
10 </head>
11 <body>
12
13   <form id="form1" runat="server">
14     <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" StartFromCurrentNode="true" />
15     <asp:SiteMapDataSource ID="SiteMapDataSource2" runat="server" StartingNodeId="~/info.aspx" />
16
17     <div><table>
18       <tr>
19         <td> Sider under den aktuelle side<br /><asp:TreeView ID="TreeView1" runat="server"
- DataSourceID="SiteMapDataSource1" /><br /><br />
20         Informationsgruppe af sider:<br />
21         <br /><br /><asp:TreeView ID="TreeView2" runat="server" DataSourceID="SiteMapDataSource2">
22         </asp:TreeView>
23         </td>
24
25         <td><asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
26
27           </asp:ContentPlaceHolder>
28
29         </td>
30       </tr>
31     </table>
32
33   </div>
34 </form>
35 </body>
36 </html>
```

Default.aspx

```
1 <%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
- CodeFile="Default.aspx.cs" Inherits="_Default" %>
2
3 <asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
4 </asp:Content>
5 <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
6 <asp:SiteMapPath ID="SiteMapPath1" runat="server" /><br /><br />
```

7	
8	Velkommen til vores lille firma
9	</asp:Content>

Produkt.aspx

1	<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
-	CodeFile="produkt.aspx.cs" Inherits="produkt" %>
2	
3	<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
4	</asp:Content>
5	<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
6	<asp:SiteMapPath ID="SiteMapPath1" runat="server" />
7	
8	Her vises alle vore produkter
9	</asp:Content>

Info.aspx

1	<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
-	CodeFile="info.aspx.cs" Inherits="info" %>
2	
3	<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
4	</asp:Content>
5	<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
6	Du er nu på informationssiden
7	</asp:Content>