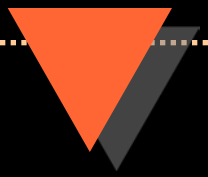


XML - kurs

XML-kurs for nybegynnere

XML: kursoversikt

- XML syntaks, elementer, attributter, entiteter
- Visningsmåter: CSS, XSL …
- DTD: Document Type Definition
- Namespace
- XML Schema
- XSLT, Xpath …
- Hands-on koding!
- Kilder til videre studier

Hvorfor lære XML?

- **Microsoft:**
  - ”XML vil revolusjonere web og være fundamentet for utvikling av neste generasjon web-løsninger”
  - ”XML vil være grunnleggende komponenter i alle fremtidige Microsoft produkter”
- **Oracle:**
  - ”XML har det som trengs for å bli det allmenne utvekslingsformat på web for både applikasjoner og dokumenter”

Vi får henge med å se hva dette har å by på for oss!

XML standarden

- World Wide Web Consortium utvikler XML
- XML 1.0 Recommendation
  - XML Working Group februar 1998
    5th ed 26. november 2008
    http://www.w3.org/TR/REC-xml

Hva er XML?

- XML=eXtensible Markup Language
  - ⇨ Markeringspråk, kodespråk, mye likt HTML
  - ⇨ Utviklet for å beskrive data, ikke for å vise data
  - ⇨ Taggene er ikke predefinerte, du må definere dine egne tagger
  - ⇨ XML er:
    platform uavhengig, software uavhengig, hardware uavhengig, utvekslingsbart
- XML er ren tekst

XML introduksjon

- XML=eXtensible Markup Language
  - ⇨ XML er et “subset” av SGML (Standardized General Markup Language)
  - ⇨ XML har som mål å kunne sende, motta og behandle data over web mere effektivt, men like enkelt som med HTML
  - ⇨ XML må være lett tilgjengelig og kompatibelt med SGML og HTML
- XML er SGML light for web

SGML, HTML, XML, XHTML

- **SGML**
  har røtter tilbake til slutten av 1960 omfattende layout og utvekslingsformat
- **HTML**
  applikasjon av SGML med predefinert markup 1989 -
- **XML**
  subset, forenklet utgave av SGML 1998 -
- **XHTML**
  overgang fra HTML til XML. Man kan blande tagger i samme dokument. Strict HTML - Strengere regler for HTML koding.

HTML versus XML koding

- **HTML**
  The best picture in 1998 went to the film <I>Titanic</I>
- **XML**
  The best picture in 1998 went to the film <FILM>Titanic</FILM >

- **XML**
  The <ACADEMY-AWARD-CATEGORY>best picture </ACADEMY-AWARD-CATEGORY> in <YEAR>1998 </YEAR> went to the film<TITLE MEDIA=”film”> Titanic</TITLE >

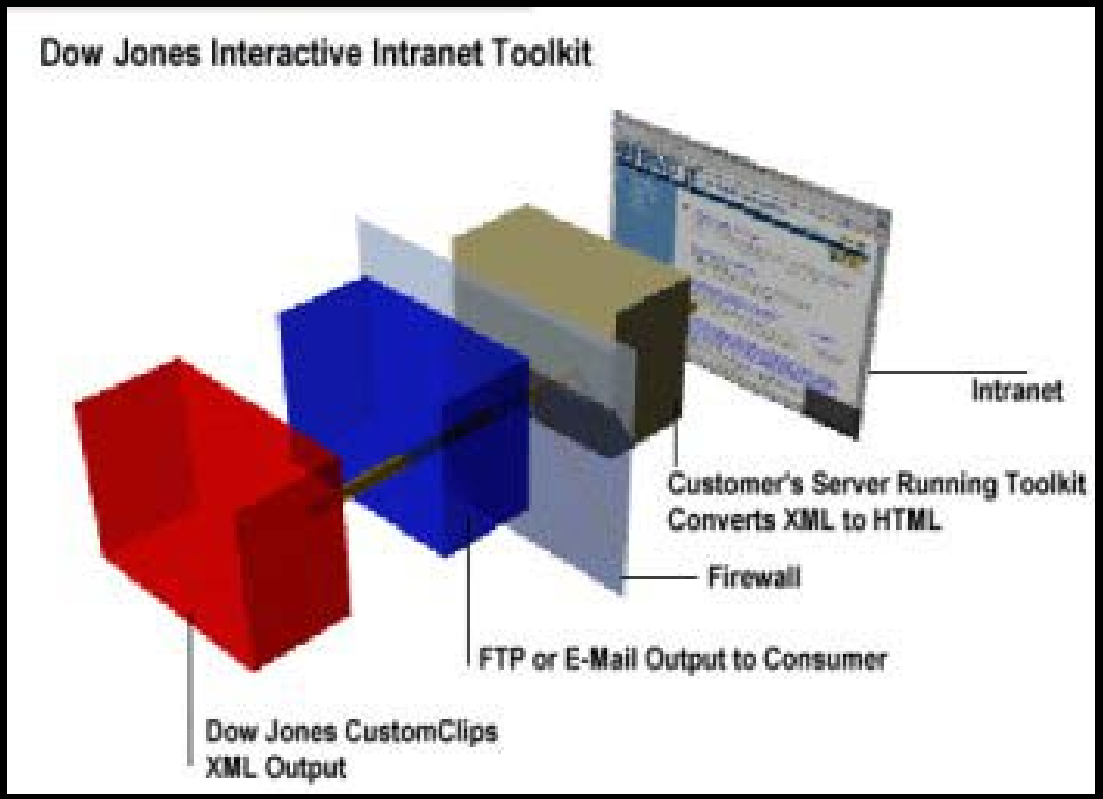
Hva kan XML brukes til?

- Dokumenter som ikke består av typiske komponenter, f eks noter, matematikk …
- Databaser: http://pubmed.gov
- Dokumenter som du ønsker å organisere i en trestruktur
- Dokumenter som du ønsker å utveksle med andre applikasjoner/systemer !!

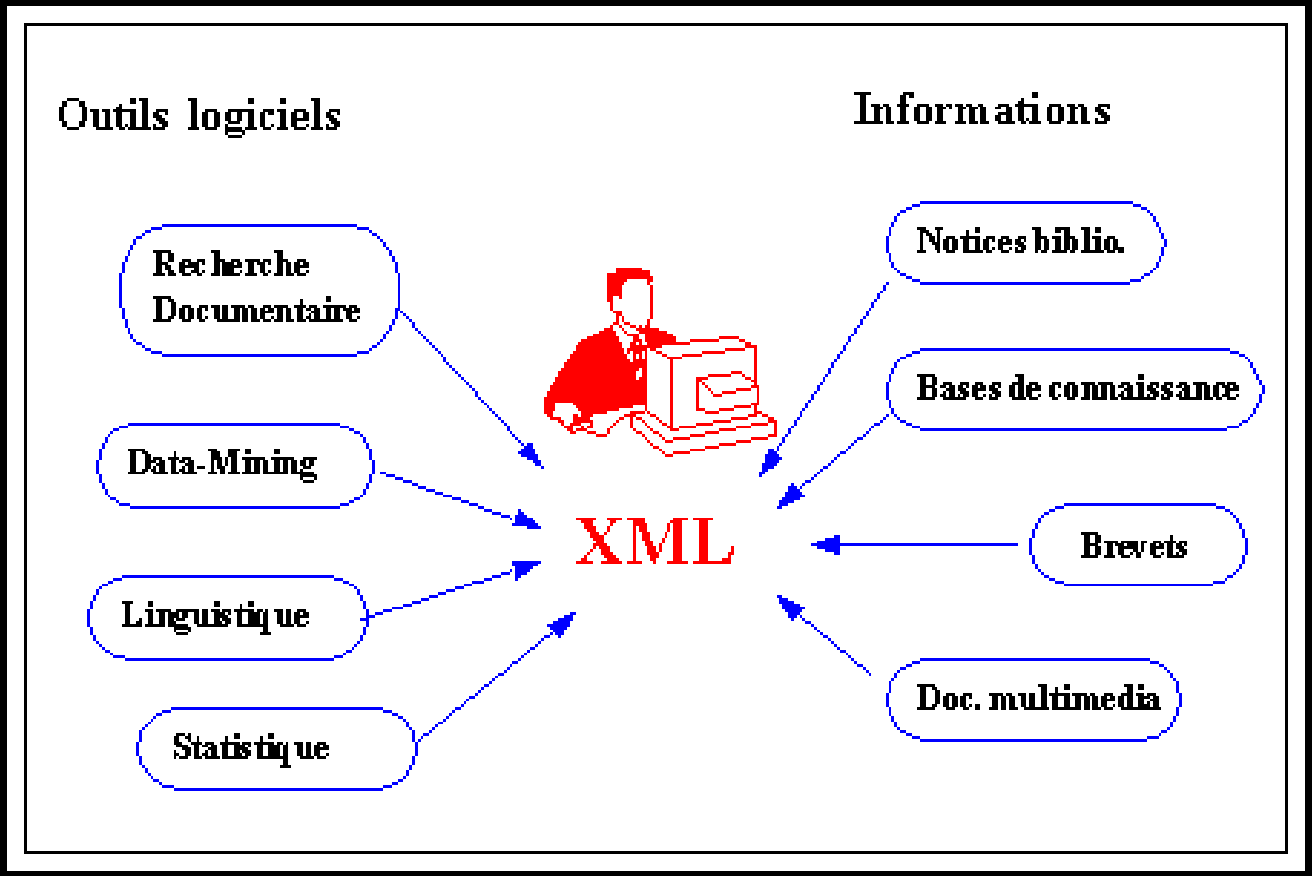
XML brukes til datautveksling

- Med XML kan data bli utvekslet mellom inkompatible systemer
  - I den virkelige verden inneholder datasystemer og databaser **inkompatible formater**. Tidkrevende utfordringer for utviklere å utveksle data mellom slike systemer over Internett.
  - **Konvertering** av dataene til XML kan redusere kompleksiteten og lage data som kan leses av forskjellige typer applikasjoner.

XML og B2B

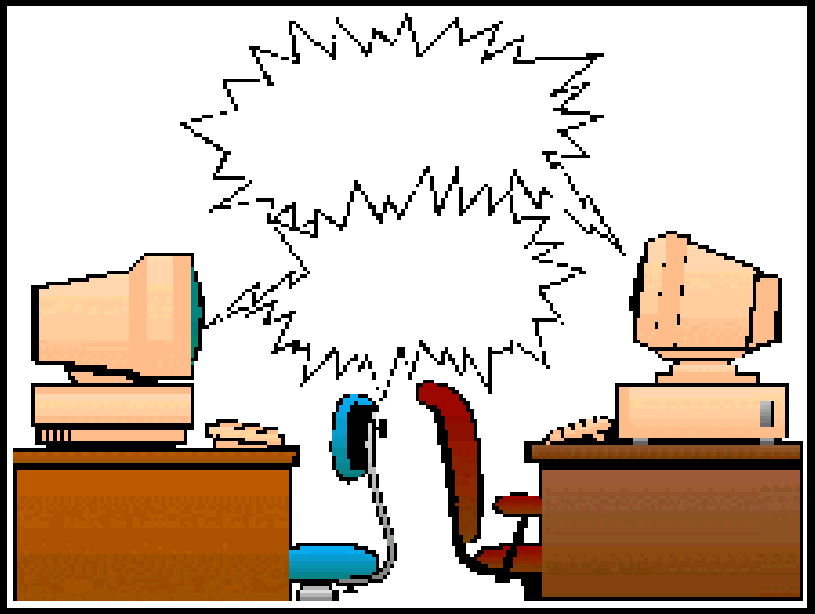


XML til litt av hvert - på fransk!



XML gjør dine data nyttigere

- XML filer kan gjøres tilgjengelig for annet enn standard browsere
- Andre klienter, applikajoner og ”lesemaskiner” (agenter) kan få tilgang til dine XML filer som datakilder på samme måte som de har tilgang til databaser
- XML er laget for maskiner, men kan leses av folk også



XML: grunnlag for andre kodespråk

- XML er mor-språket til WAP / WML
  - Wireless Application Protocol
  - Wireless Markup Language er skrevet i XML og brukt til Internett-applikasjoner for hånd-holdte PC'er og mobiltelefoner

XML syntaks

- Syntaksreglene for XML:
  - **er veldig enkle, men strenge**
  - **er enkle å lære og enkle å bruke (?! …)**
- Resultat:
  - Det er enkelt å utvikle programvare som kan lese og behandle XML

Oppgave: xml dokument

XML syntaks forts...

- XML dokumentet består av:
  - **Prolog:**
    xml declaration, ”white space”, kommentarer, DTD eller XML Schema, processing instructions
    gir informasjon til en bestemt applikasjon som skal lese XML-dokumentet
  - **Document elements**
    rot-element og nøstede elementer

XML regler

- Et ”root element” omslutter dokumentet
- Barn (child) elementer innenfor rotelementet
- Ingen tagger kan være ”empty”
- Attributters verdier i anførselstegn
- Tagger må være nøstet korrekt
- Tagger er ”case sensitive” og må ”matche”

”Bare bones” for et well-formed XML dokument!

XML elementer

Elementer har innhold:

- Element content– andre elementer
- Mixed content – tekst og andre elementer
- Simple content – bare tekst
- Empty content– ingen informasjon
  <prod id=”33-657”> </prod>
- Attribute content – attributt-verdipar i starttaggen
- Mixed content – blandet innhold - eks:
  <title>
  Moby-Dick
  <subtitle>Or, the Whale </subtitle>
  </title>

Navngiving av elementer

- Gi beskrivende navn, bruk eventuelt understrek: <bok_tittel>
- Ingen mellomrom er tillat
- Ikke bruk : som er del av syntaksen
- Ikke start navnet med et tall eller tegn unntatt _
- Det anbefales å ikke bruke xml eller XML i navnet
- Det meste er tillat i elementnavn, norske bokstaver..., men tenk på eventuelle problemer med software som skal behandle dataene
- XML dokumenter har ofte en parallell database, hensiktsmessig at elementnavn samsvarer med feltnavn

XML attributter

- XML elementer kan ha attributter akkurat som HTML
- Attributter tilfører tilleggsinformasjon til elementene
  i HTML:<IMG SRC=”computer.gif”>
  i XML: <FILE TYPE=”gif”>computer.gif</FILE>
- Attributter tilfører ofte informasjon som ikke er en del av dataene, men viktig for applikasjonen som skal behandle...
- Attributtverdier må være omsluttet av anførselstegn
- En avveining om man skal bruke attributter eller lage nye elementer – ingen klare regler ...

Eksemplet fra Medline databasen!

Attributter eller ikke? Noen problemer:

Attributter:

- kan ikke ha flere verdier (barnelementer kan)
- er ikke så lette å forandre i fremtidige revideringer
- kan ikke beskrive struktur (barnelementer kan)
- er vanskelige å manipulere i programkode
- lar seg ikke lett teste mot en DTD

En konklusjon kan være at metadata (data om data) kan lagres som attributter og at selve dataene skal lagres som elementer.

XML-attributter

```
<note>
<date>
  <day>12</day>
  <month>11</month>
  <year>2002</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>

<note day="12" month="11" year="2002"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

Entiteter

- Innebygde tegn-entiteter erstatter tegn som har spesiell betydning i XML.
  Eks:
  < - > - & - " - '

- Generelle entiteter defineres i en DTD og erstattes når de benyttes. Kan brukes i elementer og attributter:
  Eks:
  &ICONS; kurs gir…
  Etter prosessering: ICONS Managment kurs gir…

CDATA sections

- CDATA – tegndata – character data
- Blir ikke ”parsed”, tolket, av XML prosessoren
- F eks en kodesnutt som ikke skal XML-behandles:

```
<![CDATA[
innhold som ikke skal tolkes ….
]]>
```

Visning av XML dokumenter

- Hvordan skal nettleseren vite hvordan <BOOK>, BINDING>... skal vises?
  Det er hovedsakelig 3 metoder (i tillegg til å vise rå XML):

  1) Style Sheet Linking
  2) Data Binding & Data Islands
  3) Scripting

Visning i Internet Explorer

- IE 5.0 og høyere støtter visning av XML dokumenter på følgende måter:
  - Visning av ”rå” XML dokumenter
  - Visning med CSS
  - Visning med XSL
  - XML knyttet til HTML elementer, Data Binding
  - XML inne i HTML som XML Data Island
  - Javascripts - tilgang til DOM (Document Object Model)

Visning med CSS

- XML er utviklet for å lagre, bære og utveksle data – ikke for å vise data !
  - XML gjør ingenting!

Visning med CSS forts…

- Cascading StyleSheet koding for XML er den samme som for HTML

Eks:

```
book {
display: block;
font-size: 10pt;
}
title {
color: #006666;
}
```

Visning med CSS forts…

- Stilarkfilen er en helt vanlig tekstfil som lagres med extension .css
- Stilarfilen lenkes til XML-dokumentet med en processing instruction plassert rett over rotelementet:

<?xml-stylesheet type=”text/css” href=”filnavn.css”?>

Oppgave: visning med css

XML i Data Islands

- XML kan hentes inn i HTML sider i Data Islands med den uoffisielle <xml> taggen:
  <xml id="book">
  <book>……(hele XML'en her altså)</book> </xml>
- Eller en separat XML fil kan legges inn:
  <xml id="book" src="book.xml"></xml>

<xml> taggen er en HTML tagg – ikke en XML tagg

I begge tilfeller legges koden i html <body>

Data Binding

- Data Islands bindes til HTML elementer: og brukes mye

```
<html>
<body>
<xml id="cdcat" src="cd_catalog.xml"></xml>
 <table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>
</body>
</html>
```

Oppgave Data Island & Binding

Visning med scripts

- Javascript og andre ”språk” kan vise XML i tolkeren - Internet Explorer i vårt tilfelle:

```
<script type="text/javascript">
var xmlDoc = new
  ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")
// ..... processing the document goes here
</script>
```

Visning med scripts forts.

- Lasting av en tekst-streng:

```
<script type="text/javascript">
var text="<note>"
text=text+"<to>Tove</to><from>Jani</from>"
text=text+"<heading>Reminder</heading>"
text=text+"<body>Don't forget me this weekend!</body>"
text=text+"</note>”

var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false”
xmlDoc.loadXML(text)
// ..... processing the document goes here
</script>
```

Valide XML dokumenter

- XML med riktig syntaks er Well-formed
  (Hittil har våre dokumenter vært velformede.)
- XML som i tillegg er i henhold til en DTD (Document Type Definition) eller et XML Schema er Valid XML
  - Dokumentets prolog må inkludere en: Document Type Definition (DTD) eller et Schema
  - Resten av dokumentet må rette seg etter strukturen definert i DTD'en eller XML-skjemaet

Valide dokumenter forts.

- Det er nyttig å gjøre XML-dokumentene valide for å skape enhet i grupper av liknende dokumenter
- Det blir grammatikken for en klasse dokumenter
- Det finnes ferdig utviklede DTDer og skjema for forskjellige fagområder. Man behøver ikke lage sine egne DTDer og skjema
- Gratis DTDer og skjema kan lastes ned fra Internet etc.

Hva er en DTD?

- **En DTD definerer de lovlige elementene i et XML dokument**
- Hensikten med en DTD er å definere en stram struktur som passer på at man bruker gyldige elementer og at den logiske strukturen i dokumentet følges
- DTDen lister opp hvilke navn som kan brukes for forskjellige elementer, hvor de kan forekomme og hvordan det hele skal settes sammen

Hvordan ser en DTD ut?

- Den relevante delen av en DTD for et dokument som innholder en liste med listepunkter kan se slik ut:

```
<!ELEMENT Liste (Listepunkt)+>
<!ELEMENT Listepunkt (#PCDATA)>
```

- Programmer leser DTDen før de leser dokumentet slik at applikasjonene (editorer, nettlesere, søkemaskiner, databaser etc) vet hva de trenger å vite på forhånd og kan innstille seg korrekt for å behandle data i dokumentet. Lagres som .dtd hvis ekstern DTD.

DTD forts

```
<Liste>
<Listepunkt>Langrenn</Listepunkt>
<Listepunkt>Slalom</Listepunkt>
<Listepunkt>Utfor</Listepunkt>
</Liste>
```

```
<!DOCTYPE INVENTORY
  [
<!ELEMENT INVENTORY (BOOK)*>
<!ELEMENT BOOK (TITLE, AUTHOR, BINDING, PAGES, PRICE)>
<!ATTLIST BOOK   InStock (yes|no) #REQUIRED>
<!ELEMENT TITLE (#PCDATA | SUBTITLE)*>
<!ELEMENT SUBTITLE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ATTLIST AUTHOR   Born CDATA #IMPLIED>
<!ELEMENT BINDING (#PCDATA)>
<!ELEMENT PAGES (#PCDATA)>
<!ELEMENT PRICE (#PCDATA)>
  ]
>
```

Elementdeklarasjoner i DTD:

? 0 eller én forekomst (hvis utelatt, så én og bare én forekomst)

***** 0 eller flere forekomster

+ 1 eller flere forekomster

^ Obligatorisk og én forekomst

, Elementene skal komme i den rekkefølge de er listet opp

| Et eller flere av disse elementene kan komme i tilfeldig rekkefølge

Oppgave: valid dokument. Oppgave: lage en dtd

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER "Vervet Logic Times">
<!ENTITY PUBLISHER "Vervet Logic Press">
<!ENTITY COPYRIGHT "Copyright 1998 Vervet Logic Press">

]>
```

Ekstern DTD

Vi har hittil brukt interne DTDer. Vi kan også ha DTDen som ekstern fil og lenke den inn i XML dokumentet:

```
<?xml version="1.0" encoding="utf-8"?>
<!--File Name: Inventory02.xml-->
<?xml-stylesheet type="text/css" href="Inventory02.css"?>
<!DOCTYPE inventory SYSTEM "valid.dtd">
<inventory>
  etc etc
</inventory>
```

XML Namspaces (navnerom)

```
<table>
    <tr>
    <td>Apples</td>
    <td>Bananas</td>
    </tr>
</table>
```

```
<table>
   <name>African
   Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

XML Namespaces forts.

1. Sette prefiks til elementet:

```
<f:table>
 <f:name>African Coffee Table</f:name>
 <f:width>80</f:width>
 <f:length>120</f:length>
 </f: table>
```

2. xmlns attribute for å gi element-prefikset et ”qualified name” assosiert med et namespace:

```
<f:table xmlns:f=“http://www.w3schools.com/furniture”>
```

Namespaces forts.

- <elementnavn xmlns:navn=”uri”>
  Eks:
  <kurs xmlns:kurs=”http://www.icons.no/foreleser>
  - Kvalifisert elementnavn angis slik: <kurs:foreleser>

- Angir at elementet og dets barn tilhører det samme navnerommet
- For å unngå navnekonflikter når flere XML-dokumenter slås sammen til ett
- URLen som oppgis er ikke henvisning til en DTD, men beskriver navnerommet

Hva er XML Schema?

XML schema definerer:

- Hvilke elementer, attributter og barnelementer som kan forekomme i et dokument
- Rekkefølgen og antall barnelementer
- Om et element er ”empty” eller ikke
- Hvilke datatyper for elementer og attributter
- Standard (default) og faste (fixed) verdier for elementer og attributter

Hva er XML Schema?

- DTD tilhører SGML verden, men brukes også for XML og HTML dokumenter
- XML Schema er et XML basert alternativ til DTD sannsynligvis arvtakeren til DTD
- XML Schema er XML-dokumenter
- XML Schema støtter data typer
- XML Schema støtter navnerom
- XML Schema kan transformeres med XSLT

XSD elements

- XML Schema Language er også kalt XML Schema Definition (XSD)
- Syntaks for å deklarere et simple element:
  <xs:element name=”nnn” type=”nnn” />

```
<lastname>Buntz</lastname>
<age>67</age>
<dateborn>1943-03-13</dateborn>
```

```
<xs:element name="lastname" type="xs:string" />
<xs:element name="age" type="xs:integer" />
<xs:element name="dateborn" type="xs:date" />
```

XML skjema: Datatypes

- xs:string: en streng med tekst
- xs:boolean: verdien true eller false
- xs:decimal: desimaltall
- xs:integer: heltall
- xs:date: dato på formatet yyyy-mm-dd

XML Schema: deklarere elementer

- Deklarere elementer som bruker datatypene gjøres med: xs:element

Deklarerer et navn-element:

```
<xs:element name=”navn” type=”xs:string”>
```

Deklarere et person-element med utgangspunkt i XML-dokumentet:

```
 <person>
      <navn>Nina T. Svendsen</navn>
      <tlf>34343434</tlf>
</person>
```

Deklarere elementer forts…

- Person-elementet er av **complexType**. Vi bruker datatypen xs:string på navn-elementet og xs:integer på tlf-elementet:

```
<xs:element name=”person”>
 <xs:complexType>
  <xs:sequence>
     <xs:element name=”navn” type=”xs:string” />
     <xs:element name=”tlf” type=”xs:integer” />
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Deklarere attributter

- Vi kan deklarere attributter til elementene i XML-dokumentet. I følgende eksempel har elementet person et attributt personnr:

```
<person personnr=”12345678912”>
   <navn>Nina T. Svendsen</navn>
  <tlf>34343434</tlf>
</person>
```

- <xs:attribute name=”personnr” type=”xs:integer” />

Deklarere elementer og attributter

- Atributter deklareres til slutt i skjemaet

```
<?xml version=”1.0”?>
<xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”>
 <xs:element name=”person”>
   <xs:complexType>
    <xs:sequence>
      <xs:element name=”navn” type=”xs:string” />
      <xs:element name=”tlf” type=”xs:integer” />
    </xs:sequence>
   <xs:attribute name=”personnr” type=”xs:integer” />
  </xs:complexType>
 </xs:element>
 </xs:schema>
```

Bruk av attributter

- Attributtet, use, angir hvordan det deklarerte attributtet skal brukes. Use har følgende verdier:
  - required: attributtet må være med
  - fixed: verdien er satt i XML-skjemaet. Brukes sammen med value-attributtet som angir verdien
  - default: setter en standardverdi hvis ikke annet er oppgitt. Brukes sammen med value-attributtet som angir verdien
  - optional: valgfritt om attributtet er med
  - prohibited: attributtet kan ikke brukes

```
<xs:attribute name=”valuta” type=”xs:string” use=”required” />
```

XSD Restrictions/Facets

- Elementet ”age” kan ikke være lavere enn 0 eller større enn 100:

```
<xs:element name="age">
<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

XSD Restrictions/Facets

- Elementet ”gender” kan bare ha verdien male eller female

```
<xs:element name="gender">
<xs:simpleType>
  <xs:restriction base="xs:string">
  <xs:pattern value="male|female"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Hvordan ser et XML Schema ut?

```
<?xml version=”1.0”?>
 <xs:schema xmlns:xs=”http:www-w3.org/2001/XMLSchema”>
   <xs:element name=”book”>
    <xs:complexType>
     <xs:sequence>
      <xs:element name=”title” type=”xs:string”/>
      <xs:element name=”author” type=”xs:string”/>
     <xs:sequence>
    <xs:complexType>
  </xs:element>
</xs:schema>
```

Lagres som .xsd

Oppgave: schema

Hvordan ser et XML skjema ut?

```
<?xml version="1.0" encoding="utf-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <xs:element name="book">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element name="title" type="xs:string" />
          <xs:element name="author" type="xs:string" />
        - <xs:element name="character" minOccurs="0" maxOccurs="unbounded">
          - <xs:complexType>
            - <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="friend-of" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
                <xs:element name="since" type="xs:date" />
                <xs:element name="qualification" type="xs:string" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="isbn" type="xs:string" />
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

XSL stilark

- **eXtensible Stylesheet Language** er en mer effektiv måte å få frem alle deler av dokumentet i nettleser på enn CSS
- XSL er den foretrukne stilark-visning for XML dokumenter
- XSL kan **sortere** og filtrere data .... søkefunksjoner ...
- XSL stilark transformerer XML til HTML etc for visning
  - XML stilark skrives som et velformet dokument etter XML syntaks og lagres med extension **.xsl** i samme mappe som XML dokumentet
  - XSL stilarket lenkes til XML dokumentet i form av en ”processing instruction” i prologen:

```
<?xml-stylesheet type=”text/xsl” href=”Inventory.xsl”?>
```

XSL stilark forts…

- Det startet med XSL og endte opp med XSLT, Xpath og XSL-FO

- XSLT er språket for å transformere XML dokumenter
- XPath er språket for å definere deler av et XML dokument
- XSL-FO er språket for å formatere XML dokumenter

  Formelt nå bare kalt XSL

XSLT som språk

- En applikasjon av XML
  - Et XSLT-dokument må derfor være velformet
  - XSLT-instrukser gjenkjennes på et eget navnerom som angis med prefikset **xsl:**
- Baserer seg på maler (templates) for transformering
  - Malene gjelder for noder som adresseres med Xpath uttrykk. Xpath uttrykk likner på katalogangivelser i Unix

XPath

- XSLT bruker XPath til å definere ”matching patterns” for transformering

- XSLT bruker altså XPath til å definere deler av kildedokumentet ”**source**” som ”**match**” en eller flere maler ”**templates**”. Når et samsvar er funnet, transformerer XSLT de ”matching” deler av source-dokumentet til resultat-dokumentet.

```
<td><xsl:value-of select="catalog/cd/title" /></td>
<td><xsl:value-of select="catalog/cd/artist" /></td>
```

XSL stilark forts…

- Stilarket må lenkes til XML dokumentet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
    <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
    </cd>
    ...
</catalog>
```

Oppgave: xsl

XML Linking Language

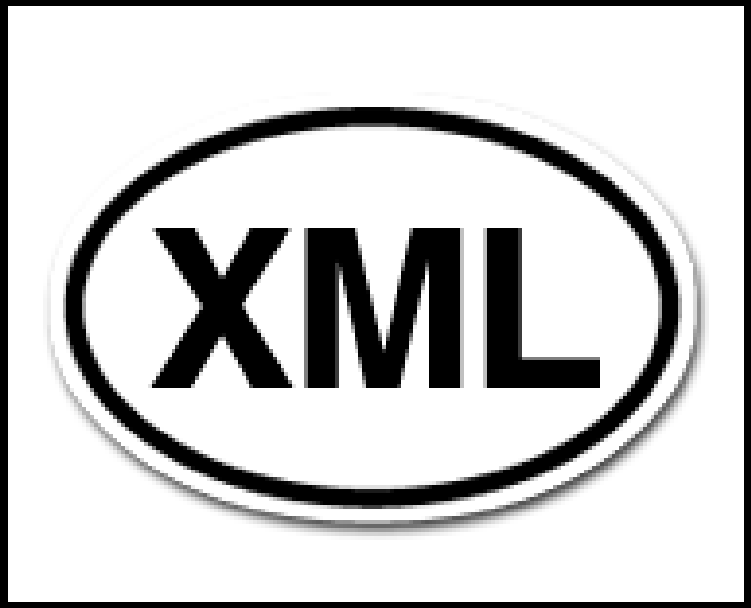
- XML er en hel familie av teknologier, XSL, Schema... Flere av teknologiene har med lenking i XML dokumenter å gjøre:
  - XLink (XML Linking Language)
  - XPointer (XML Pointer Language)
- XLink gir mange flere muligheter enn vanlig lenking i HTML og sammen men XPointer kan det lenkes til hvor som helst i et måldokument

Andre XML teknologier

- **SOAP:** Simple Object Access Protocol
  - kommunikasjonsprotokoll for utveksling av informasjon over http – protokoll for å aksessere en Web service
- **WSDL**: Web Services Description Language
  - beskriver Web services og hvordan aksessere dem
- **RDF**: Resource Description Framework
  - språk for å beskrive web ressurser, innhold, tittel, forfatter, copyright, tilgjengelighet med mer

Flere XML-forkortelser i kurskompendiet!

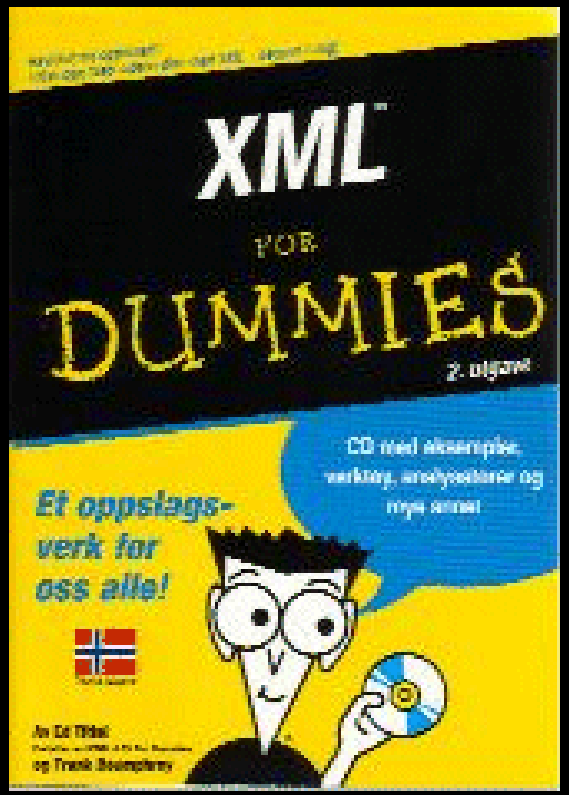
Lenker til videre studier



- World Wide Web Consortium
  www.w3.org/XML/
- World Wide Web School
  www.w3schools.com/xml/
- The XML FAQ
  www.ucc.ie/xml/
- Cover Pages (Hosted by Oasis)
  xml.coverpages.org/xml.html

Bøker til videre studier

- Castro, Elizabeth:
  XML for the World Wide Web
- Tittel, Ed:
  XML for Dummies: et oppslagsverk for oss alle
- Young, Michael:
  XML Step by Step



Search the web!

Chances are e**X**tre**M**e**L**y good that you find what you seek!