IV1023 vt2022
Avancerad Datahantering med XML

XSLT

nikos dimitrakas
nikosd@kth.se
08-161295
Rum 2423

Läsanvisningar
- Kapitel 7.2.1 i kursboken
- Kapitel 15 i XML 1.1 Bible
- Kompendiet "Introduction to XSLT"

XSLT

- **XSL Transformations**
  - **XSL: eXtensible Stylesheet Language**
- **Transformationer**
  - **Från XML**
  - **Till XML, HTML, text, etc.**
- **XSLT är ett XML-baserat språk**
  - **Ett XSLT-dokument är ett XML-dokument**
  - **Har ofta filändelsen .xsl**
  - **Namespace och definierad semantik**
- **Liknar programmeringsspråk**
  - **Rekursion**
  - **Iteration**
  - **Flödeskontroll**
  - **Variabler**

XSLT-versioner

- XSLT 1
  - Bygger på XPath 1
  - Stöds i webbläsare som Internet Explorer, Safari, Chrome, FireFox, Netscape, Opera, Vivaldi, Edge
- XSLT 2
  - Bygger på XPath 2
  - Utökade konstruktioner för bl a grupperingar
  - Flera output-format
  - Stöds inte i någon webbläsare ännu, men det finns server-side moduler, t ex Saxon (som kan användas via xsltransform.net, xslttest.appspot.com och i BaseX)
- XSLT 3
  - Tillsammans med XPath 3 och XQuery 3
  - Hantering av strömmande data
  - JSON (input och output)

XSLT-dokument

- Rotelement
  - xsl:transform eller xsl:stylesheet (synonymer)
  - attributet "version" i rotelementet styr XSLT-version
- Namespace
  - http://www.w3.org/1999/XSL/Transform
  - Rekommenderat prefix: xsl
- Länkning till/från XML-dokument
  - <?xml-stylesheet type="text/xsl" href="???.xsl"?>
  - eller dynamiskt i applikationen

Top Level-element

- Element direkt under rotelementet
  - "Deklarationer"
- XSLT 1
  - import, include, strip-space, preserve-space, output, key, decimal-format, namespace-alias, attribute-set, variable, param, template
  - Elementet template är där jobbet görs
  - Resten är konfigurationer
- XSLT 2 (utöver de ovan)
  - character-map, function, import-schema
- XSLT 3 (utöver de ovan)
  - mode, accumulator

XSLT-instruktioner

- Element inuti elementet template
- XSLT 1
  - Skapa noder: element, attribute, comment, processing-instruction, value-of, text, copy, copy-of
  - Flödeskontroll, iteration: if, choose (och when, otherwise), for-each
  - Variabler: variable, param
  - Template-anrop: apply-templates, call-template, apply-imports
  - Andra specialiserade instruktioner som t ex message och number
- XSLT 2 (utöver de ovan)
  - for-each-group, next-match, sequence, namespace, document, result-document, analyze-string
- XSLT 3 (utöver de ovan)
  - source-document, iterate, merge, fork, where-populated, on-empty, on-non-empty, try, evaluate, assert

Exempeldata

Enligt följande DTD:

```
<!ELEMENT Books (Book+)>
<!ELEMENT Book (Author+)>
<!ATTLIST Book Title CDATA #REQUIRED
        Language CDATA #REQUIRED
        Year CDATA #REQUIRED
        Publisher CDATA "N/A"
        Genre CDATA "N/A">
<!ELEMENT Author EMPTY>
<!ATTLIST Author Name CDATA #REQUIRED
        YearOfBirth CDATA #REQUIRED
        Country CDATA #REQUIRED>
```


IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

Exempeldata

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Books SYSTEM "books.dtd">
<Books>
        <Book Title="Misty Nights" Year="1997" Language="English" Publisher="Kingsly" Genre="Thriller">
                <Author Name="John Craft" YearOfBirth="1948" Country="England"></Author>
        </Book>
        <Book Title="Archeology in Egypt" Year="1992" Publisher="KLC" Language="English" Genre="Educational">
                <Author Name="Arnie Bastoft" YearOfBirth="1971" Country="Austria"></Author>
                <Author Name="Meg Gilmand" YearOfBirth="1968" Country="Australia"></Author>
                <Author Name="Chris Ryan" YearOfBirth="1944" Country="France"></Author>
        </Book>
        <Book Title="Database Systems in Practice" Year="2000" Language="English" Genre="Educational">
                <Author Name="Alan Griff" YearOfBirth="1972" Country="USA"></Author>
                <Author Name="Marty Faust" YearOfBirth="1970" Country="USA"></Author>
                <Author Name="Celine Biceau" YearOfBirth="1969" Country="Canada"></Author>
        </Book>
        <Book Title="Contact" Language="English" Year="1988" Genre="Science Fiction">
                <Author Name="Carl Sagan" YearOfBirth="1913" Country="USA"></Author>
        </Book>
        <Book Title="The Fourth Star" Year="2001" Language="English" Publisher="Bästa Bok" Genre="Science Fiction">
                <Author Name="Leslie Brenner" YearOfBirth="1945" Country="USA"></Author>
        </Book>
        <Book Title="Våren vid sjön" Year="1982" Language="Swedish" Genre="Novel">
                <Author Name="Marie Franksson" YearOfBirth="1937" Country="Sweden"></Author>
        </Book>
        <Book Title="Dödliga Data" Year="1993" Language="Swedish" Genre="Thriller">
                <Author Name="Jakob Hanson" YearOfBirth="1946" Country="Sweden"></Author>
        </Book>
        <!-- flera Book-element -->
</Books>
```

Deklaration output

- Används för att specificera resultatets format
- Har flera attribut
  - method: xml, html, text (och xhtml i XSLT 2, json i XSLT 3)
  - encoding
  - flera attribut för XML-deklarationer och konfigurationer
    - » t ex indent, media-type, omit-xml-declaration
- Exempel

```
<xsl:output method="xml" />
```


Deklaration variable

- Används för att deklarera variabler och tilldela dem värden
- Attribut
  - name: variabelns namn
  - select: variabelns värde (kan vara ett XPath uttryck vars resultat blir variabelns värde)
- Exempel

```
<xsl:variable name="x" select="5" />
```

- Variabler kan sedan användas i XPath-uttryck med prefix $

Deklaration template

- Används för att bearbeta input och konstruera output
- Namngivna template har attributet name
- Regel-template har attributet match som innehåller ett mönsteruttryck (en specifik typ av XPath-uttryck) som styr när det exekveras

- Exempel

```
<xsl:template name="abc">
</xsl:template>

<xsl:template match="/">
</xsl:template>
```


Anropa templates

- Default template
  - match="/"
  - drar igång exekveringen
- Anropa namngivna templates
  - <xsl:call-template name="abc" />
- Anropa regel-templates
  - <xsl:apply-templates />
  - <xsl:apply-templates select="Book" />
  - attributet select innehåller ett XPath-uttryck som styr vilka noder som templates skall appliceras på
  - om attributet select saknas är det alla barnnoder som gäller
  - Om ingen template matchar används default-beteenden (built-in templates)
    - » Returnerar nodens värde
    - » Flera varianter av built-in templates i XSLT3 med hjälp av xsl:mode och on-no-match

Skapande instruktioner

- element
  - Skapar element
  - Elementets namn anges i attributet name
  - Elementets innehåll konstrueras i innehållet
  - <xsl:element name="Böcker">innehållet</xsl:element>
- attribute
  - Skapar attribut för elementet
  - Attributets namn anges i attributet name
  - Attributets värde anges i attributet select (XSLT 2) eller som innehållet
  - <xsl:attribute name="Titel">värdet</xsl:attribute>
- comment
  - Skapar kommentarer
  - Kommentaren anges i attributet select (XSLT 2) eller som innehållet
  - <xsl:comment>kommentartexten</xsl:comment>

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

Skapande instruktioner

- processing-instruction
  - Skapar XML processing instructions
  - Dock inte XML-deklarationen som skapas med xsl:output
  - PI-namnet anges i attributet name
  - PI-värdet anges i attributet select (XSLT 2) eller som elementets innehåll
  - <xsl:processing-instruction name="Hälsning" select="'Hej'"/>
- namespace (XSLT 2)
  - Skapar namespace-noder (xmlns-attribut)
  - Namespace-namnet anges i attributet name
  - Namespace-värdet anges i attributet select eller som innehållet
  - <xsl:namespace name="kth" select="'http://ns.kth.se/'" />
- text
  - Skapar textnoder
  - <xsl:text>hej</xsl:text>

Skapande instruktioner

- value-of
  - Skapar en textnod från ett XPath-uttryck
    - » Endast första värdet om uttrycket ger en sekvens (XSLT 1)
    - » Alla värden separerade enligt attributet separator (XSLT 2)
  - XPath-uttrycket anges i attributet select
  - <xsl:value-of select="Book/@Title" />
- copy
  - Skapar en kopia av den aktuella noden
  - <xsl:copy />
- copy-of
  - Skapar en djup kopia av noden/noderna som är resultatet av uttrycket i attributet select
  - <xsl:copy-of select="Book" />

Skapa noder utan instruktioner

- Skriv xml-kod direkt
  - <Person />
  - Samma som <xsl:element name="Person"/>
- Attribut
  - <Person namn="Kalle" />
  - Samma som

```
<xsl:element name="Person">
        <xsl:attribute name="namn">Kalle</xsl:attribute>
</xsl:element>
```

- Dynamiska attributvärden?
  -

```
<xsl:element name="Person">
        <xsl:attribute name="namn">
                <xsl:value-of select="@Pname" />
        </xsl:attribute>
</xsl:element>
```

  - <Person namn="??" />

Attribute value templates

- För attributvärden som hämtas dynamiskt från andra noder
  - <xsl:element name="Person">
    <xsl:attribute name="namn">
    <xsl:value-of select="@Pname" />
    </xsl:attribute>
    </xsl:element>
  - <Person namn="{@Pname}" />
- Obs! Fungerar endast för attributvärden.
  Följande är alltså ogiltigt (i XSLT 1 och 2):
  - <Person>{@Pname}</Person>

Text value templates (XSLT 3)

- Textnoder skapas dynamiskt från uttryck inuti {}
  - Kräver aktivering med hjälp av attributet expand-text
    - » yes, 1 eller true (no, 0 eller false är default)
  - Attributet expand-text måste finnas hos en ancestor.

```
<Person xsl:expand-text="yes">{@Pname}</Person>

<xsl:element expand-text="yes" name="Personer">
   <Person>{@Pname}</Person>
</xsl:element>
```

Flödeskontroll

- if
  - Innehållet utförs endast om villkoret är sant
  - villkoret anges i attributet test
  - <xsl:if test="@Title='Contact' ">…</xsl:if>
- choose
  - Har ett eller flera when och eventuellt ett otherwise
  - Varje when har ett villkor som anges i attributet test
  - Endast det första matchande when exekveras, om inget when matchar, exekveras otherwise
  - <xsl:choose>
    <xsl:when test="$n=1">En</xsl:when>
    <xsl:when test="$n=0">Ingen</xsl:when>
    <xsl:otherwise>Många</xsl:otherwise>
    </xsl:choose>

Iteration

- for-each
  - Loopar igenom noderna i sekvensen som är resultatet av XPath-uttrycket i attributet select
  - <xsl:for-each select="Author">…</xsl:for-each>
- for-each-group (XSLT 2)
  - Grupperar noderna i resultatet av XPath-uttrycket i attributet select enligt uttrycket i attributet group-by (eller group-adjacent, eller group-starting-with, eller group-ending-with) och loopar igenom grupperna
  - Funktionen current-group() kan användas för att komma åt sekvensen med alla noder som tillhör den aktuella gruppen
  - Funktionen current-grouping-key() kan användas för att komma åt grupperingsvärdet i den aktuella gruppen
  - <xsl:for-each-group select="Author" group-by="@Country">
    …
    </xsl:for-each-group>
  - Stöd för komplex group-by i XSLT 3 (med attributet "composite")

Sortering

- sort
  - Kan användas i alla sorters loopar (for-each, apply-templates, for-each-group)
  - Sorterar loopens varv enligt uttrycket i attributet select
  - Sorterar på en sak, men man kan ha flera xsl:sort
  - Ordningen kan vara ascending (default) eller descending och anges i attributet order
  - Attributet data-type styr sorteringen
    Giltiga värden: text (default), number, eller annat som xs:date
  - <xsl:sort select="@Year" order="descending" data-type="number" />

Diverse

- source-document (XSLT 3)
  - Dynamiskt öppna andra XML-dokument som input
  - <xsl:source-document href="{$filnamn}"/>
- evaluate (XSLT 3)
  - Evaluera ett dynamiskt XPath-uttryck
  - <xsl:evaluate xpath="$xpathstring"/>
  - Stödjer parametrar med xsl:with-param

Viktiga funktioner

- current()
  - returnerar den aktuella noden
- position()
  - aktuell plats i sekvensen
- last()
  - antalet noder i sekvensen
- doc(uri) document(uri)
  - öppnar en XML-fil
- not(uttryck)
  - negerar parameterns booleska värde

Exempel - iteration med for each

- Alla titlar

```
<xsl:template match="/">
   <xsl:element name="Titlar">
      <xsl:for-each select="Books/Book">
         <xsl:element name="Titel">
            <xsl:value-of select="@Title" />
         </xsl:element>
      </xsl:for-each>
   </xsl:element>
</xsl:template>
```

Exempel - templates

- Alla titlar

```
<xsl:template match="/">
   <xsl:element name="Titlar">
      <xsl:apply-templates select="Books/Book" />
   </xsl:element>
</xsl:template>
<xsl:template match="Book">
   <xsl:element name="Titel">
      <xsl:value-of select="@Title" />
   </xsl:element>
</xsl:template>
```


Mer information

- XSLT 1-, XSLT 2- och XSLT 3-specifikationerna
- Kompendiet "Introduction to XSLT"
- Kapitel 15 i XML 1.1 Bible

XSLT kan exekveras med hjälp av

- Webbläsare (endast XSLT 1)
- http://xsltransform.net
- http://xslttest.appspot.com
- Oracle, DB2, SQL Server (endast XSLT 1 med hjälp av SQL)
- BaseX (med hjälp av XQuery)
- Eget program med XSLT-bibliotek (t ex Saxon)

Fortsättning

- Quiz om XSLT
- Labb om XSLT (kompendiet "Introduction to XSLT")
  - Innehåller flera exempel
- Lektionsuppgifter
- Seminarieuppgifter (Inlupp 1)
- Inlupp 3