XSLT

Extensible Stylesheet
Language Transformations

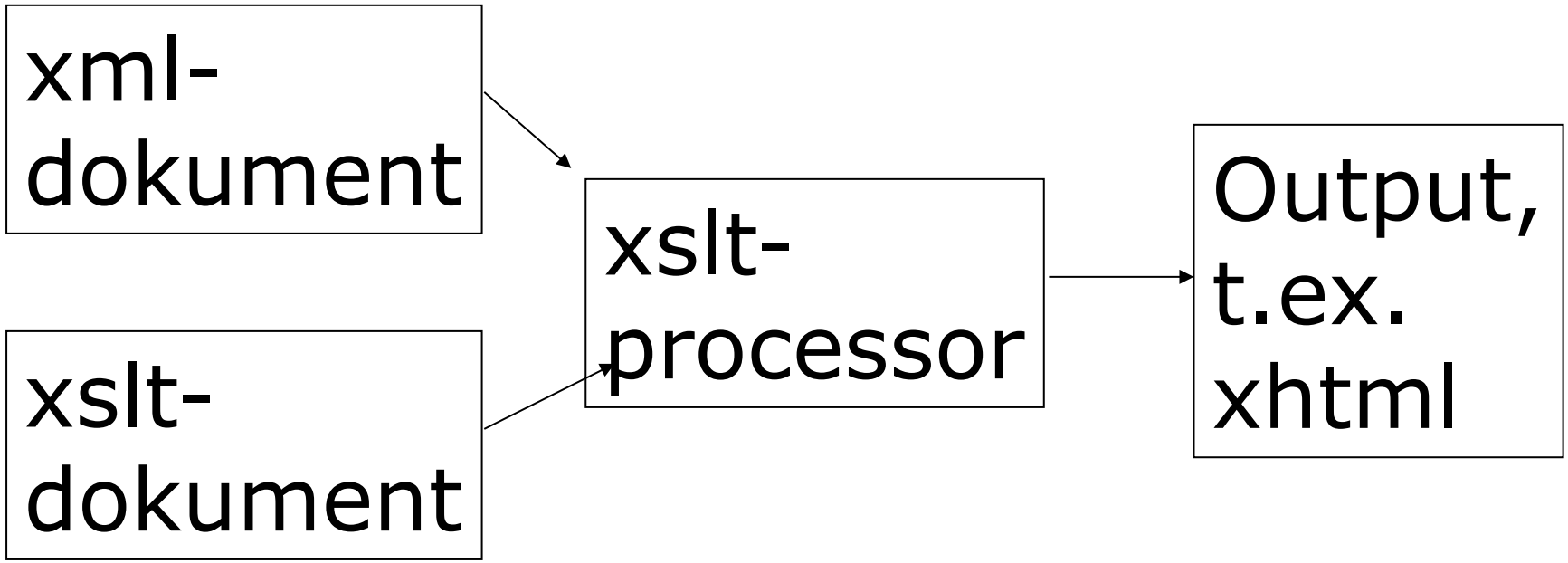
Vad är XSLT

- Ett transformationsspråk som transformerar ett XML-dokument till ett annat XML- eller textdokument.
- Kan t.ex. användas för att transformera en XML-struktur till XHTML och WML.
- En del av XSL som förutom XSLT inkluderar XSL-FO.
- Rekommendation från W3C.

XSLT-processor

- XSLT är en rekommendation, inte ett program.
- Det finns däremot program som kan utföra XSLT-transformationer, sk XSLT-processorer.
- Xalan och SAXON är vanliga XSLT-processorer.
- Inbyggt stöd i Firefox och Internet Explorer

XSLT-processor
Exempel



XSLT vs CSS

- Precis som för CSS letar man efter mönster (patterns) i en XML-struktur, och när dessa hittas utförs åtgärder (instruction elements)
- Till skillnad från CSS är syftet med XSLT att förändra ett XML-träds struktur, inte dess visuella representation.
- XSLT kan skapa helt nya element, byta ordning på element, välja ut delar av innehållet osv, vilket inte CSS kan.
- XML-strukturen som är indata kallas ”input-träd” och utdata kallas ”resultat-träd”. CSS har inga resultat-träd.

Namespace för XSLT

- Den namespace som används är http://www.w3.org/1999/XSL/Transform
- Den brukar bindas till prefixet xsl
- Root-elementet är <xsl:stylesheet> eller <xsl:transform> vilka är ekvivalenta.
- <xsl:stylesheet> används nästan alltid.

Exempel XSLT-dokument

```
<xsl:stylesheet version=”1.0”
  xmlns:xsl=”
  http://www.w3.org/1999/XSL/
  Transform”>
    <!-- innehåll -->
  </xsl:stylesheet>
```

Templates

- Mönstermatchningen görs i elementet <xsl:template>
- <xsl:template> tar ett obligatoriskt attribut ”match” vars värdedel är ett Xpath-uttryck som matchar delar av trädstrukturer.
- Elementinnehållet är ett eller flera ”instruction element”, s.k. ”action”

Exempel Templates

```
<xsl:stylesheet version=”1.0” xmlns:xsl=
  ”http://www.w3.org/1999/XSL/Transform”>
  <xsl:template match=”/”>
    [action]
  </xsl:template>
  <xsl:template match=”party”>
    [action]
  </xsl:template>
</xsl:stylesheet>
```

instruction elements (actions)

- En action är ett uttryck som bestämmer vad som ska sättas in i resultatträdet när ett visst mönster påträffas.
- Några vanliga actions är
  - Sätt in XML-element, attribut och textnoder.
  - Selektera en mängd noder för vidare template-matching.
  - Sätt inte in något alls.

Exempel Actions

```
<xsl:stylesheet version=”1.0” xmlns:xsl= ”http://
   www.w3.org/1999/XSL/Transform”>
  <xsl:template match=”/”>
    <html>1)
      <xsl:apply-templates/>2)
    </html>1)
  </xsl:template>
  <xsl:template match=”party”/>3)
</xsl:stylesheet>
```

1)Sätt in XML-element, attribut och textnoder.

2)Selektera noder för vidare template-matching.

3)Sätt inte in något alls.

<xsl:apply-templates>

- <xsl:apply-templates> används för att välja ut en mängd nya noder, och applicera eventuella templates på dessa.
- Det har ett frivilligt ”select”-attribut som innehåller en XPath relativt den aktuella noden/elementet, den så kallade ”kontextnoden”.

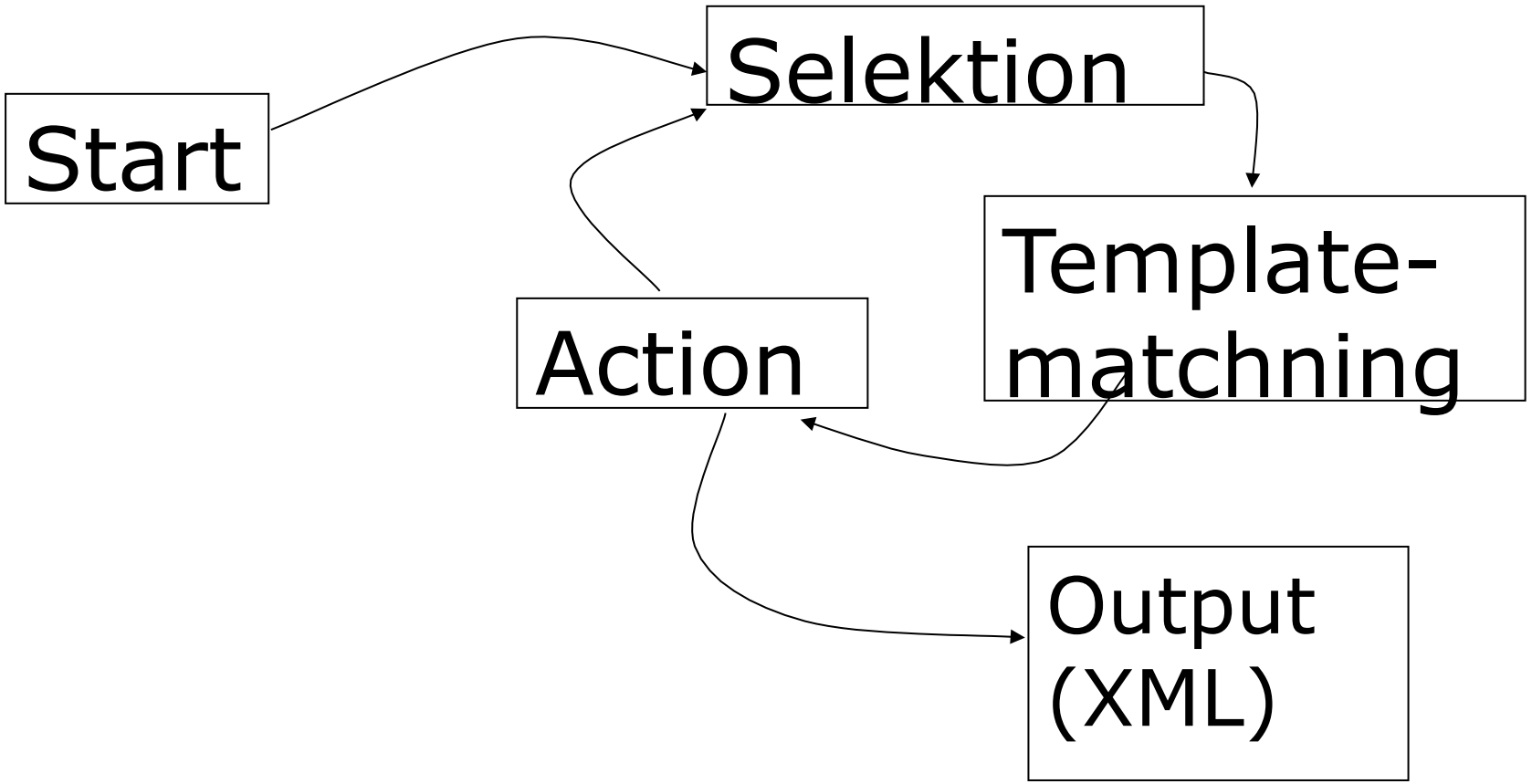
Kontextnoden

- Kontextnoden är den nod i käll-trädet som behandlas för tillfället.
- Relativa Xpath-uttryck utgår från kontextnoden
- Apply-templates väljer ut en mängd noder, vilka sedan gås igenom i tur och ordning. Dessa noder blir då i tur och ordning kontextnoder, och alla Xpath-uttryck utgår då från kontextnoden.
- XSLT-processorn utgår från dokumentnoden och om inget annat anges blir därefter root-noden kontextnod.

<xsl:apply-templates>

```
<xsl:stylesheet version=”1.0” xmlns:xsl=
  ”http://www.w3.org/1999/XSL/Transform”>
  <xsl:template match=”/”>
    <html>
      <xsl:apply-templates select=”party” />
    </html>
  </xsl:template>
  <xsl:template match=”party”>
    <xsl:apply-templates select=”date”/>
    <xsl:apply-templates select=”title”/>
  </xsl:template>
</xsl:stylesheet>
```

Loopen selektion-> template-matchning-> actions->selektion



Binda XSLT-stylesheets till XML-dokument

- Det enklaste sättet att binda ett xslt-stylesheet till ett xml-dokument är att ange båda som inparametrar vid anropet av xslt-processorn.
- Ett annat är att ange en processinstruktion i xml-dokumentet med en href till xslt-dokumentet.
- Ett tredje sätt är att göra det via ett XSLT-api från något programmerinsspråk.

Exempel XSLT-bindning

Vid anrop av XSLT-processorn

```
>java org.apache.xalan.xslt.Process -IN Party.xml -XSL
   Party.xsl
```

Genom processinstruktion

```
<?xml-stylesheet type=”text/xsl” =”Party.xsl”?>
<!DOCTYPE party SYSTEM "party.dtd">
<party>
  <title>Amphiox Gasque</title>
  <date>2001-09-08</date>
  <starttime>18.00</starttime>
  <endtime>03.00</endtime>
</party>
```

Komplett exempel

Party.xml

```
<?xml-stylesheet type=”text/xsl”
    href=”Party.xsl”?>
<!DOCTYPE party SYSTEM
    "party.dtd">
<party>
  <title>Amphiox Gasque</title>
  <date>2001-09-08</date>
<starttime>18.00</starttime>
  <endtime>03.00</endtime>
</party>
```

output

```
<html>
  <head><title>
    Amphiox Gasque
  </title></head>
</html>
```

Party.xsl

```
<xsl:stylesheet version=”1.0” xmlns:xsl=
    ”http://www.w3.org/1999/XSL/
    Transform”>
  <xsl:template match=”party”>
    <html>
      <xsl:apply-templates />
    </html>
  </xsl:template>
  <xsl:template match=”title”>
    <head><title>
      <xsl:apply-templates/>
    </title></head>
  </xsl:template>
  <xsl:template match=”date”/>
  <xsl:template match=”starttime”/>
  <xsl:template match=”endtime”/>
</xsl:stylesheet>
```

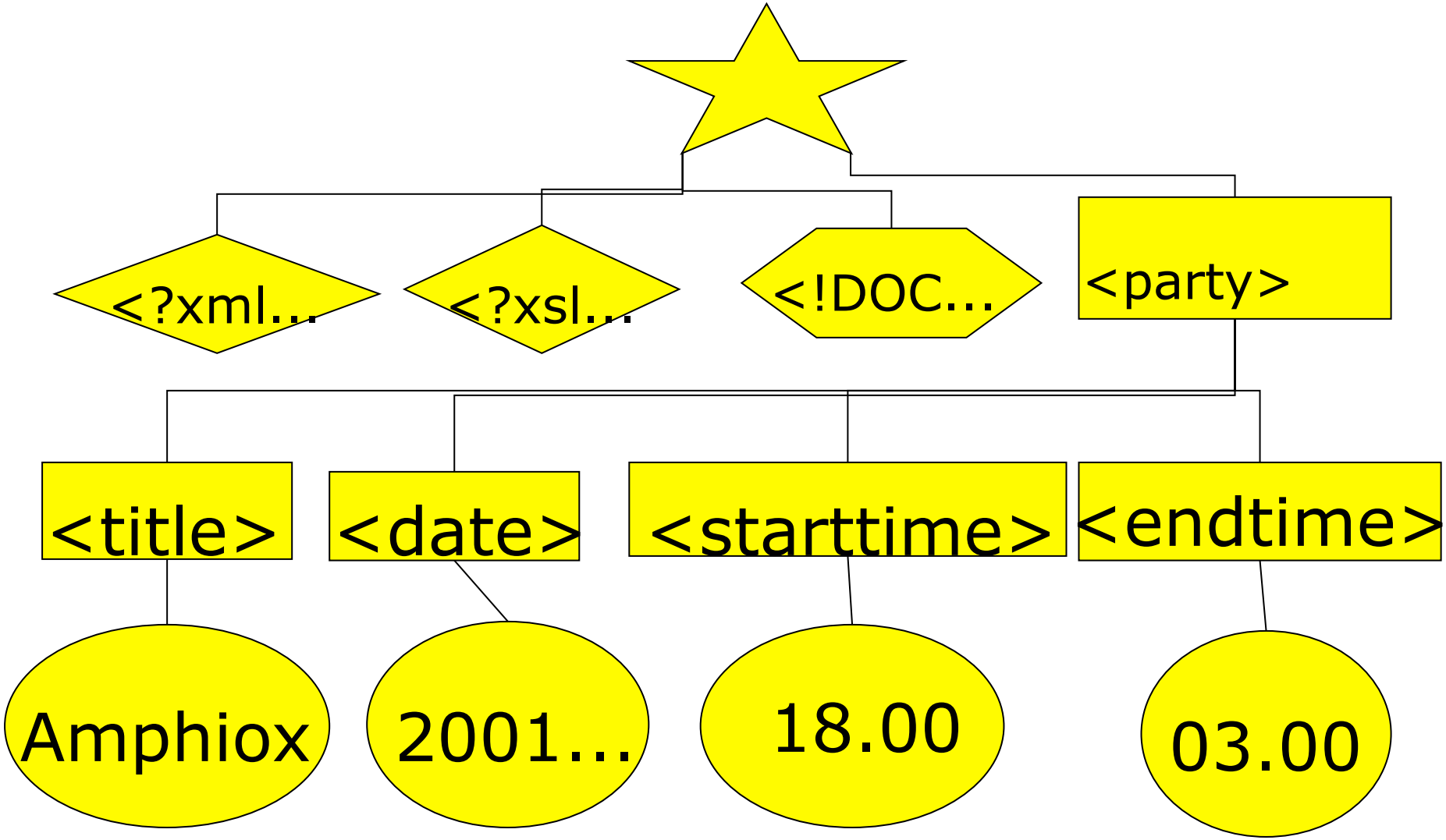
Kokboksmetod - 1

- Här följer en kokboksmetod för hur man kan komma igång med att skapa ett stylesheet givet ett visst xml-dokument, t.ex. party.xml.
- Figurerna till höger kommer användas.

Kokboksmetod - 1
Party.xml

```
<?xml version=”1.0”?>
<?xml-stylesheet type=”text/xsl”
  href=”Party.xsl”?>
<!DOCTYPE party SYSTEM "party.dtd">
<party>
  <title>Amphiox Gasque</title>
  <date>2001-09-08</date>
  <starttime>18.00</starttime>
  <endtime>03.00</endtime>
</party>
```

Kokboksmetod - 1
Grafisk representation



Kokboksmetod - 2

- Skapa en template för varje typ av elementnod i trädet (dvs om elementet <guest> förekommer fler än en gång i inputdokumentet ska ändå endast en template skapas).
- Låt ”action” vara <xsl:apply-templates/>

Kokboksmetod - 2
Party.xsl

```
<xsl:stylesheet version=”1.0” xmlns:xsl= ”http://www.w3.org/1999/XSL/
    Transform”>
  <xsl:template match=”party”>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match=”title”>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match=”date”>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match=”starttime”>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match=”endtime”>
    <xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

Kokboksmetod - 3

- Tag bort apply-templates från de regler som ska returnera tomt resultat och ta bort eventuella templates som därigenom aldrig blir åtkomliga.

Kokboksmetod - 3
Party.xsl

```
<xsl:stylesheet version=”1.0” xmlns:xsl=”http://
   www.w3.org/1999/XSL/Transform”>
  <xsl:template match=”party”>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match=”title”>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match=”date”/>
  <xsl:template match=”starttime”/>
  <xsl:template match=”endtime”/>
</xsl:stylesheet>
```

Kokboksmetod - 4

- Lägg till och modifiera action-delen på återstående templates.
- Exempelvis lägga till <html> och </html> till root-elementets action ifall output ska vara html.

Kokboksmetod - 4
Party.xsl

```
<xsl:stylesheet version=”1.0” xmlns:xsl=”http://www.w3.org/1999/
   XSL/Transform”>
  <xsl:template match=”party”>
    <html>
      <xsl:apply-templates/>
    </html>
  </xsl:template>
  <xsl:template match=”title”>
    <head><title>
      <xsl:apply-templates/>
    </title></head>
  </xsl:template>
  <xsl:template match=”date”/>
  <xsl:template match=”starttime”/>
  <xsl:template match=”endtime”/>
</xsl:stylesheet>
```

XSLT-dokument är även XML-dokument

- Ett XSLT-dokument är även ett fullständigt XML-dokument.
- Det betyder att det måste vara well-formed, dvs exempelvis att alla starttags måste ha motsvarande sluttags på rätt plats.

Exempel well-formed och icke-well-formed

Korrekt template

```
<xsl:template match=”title”>
  <head><title>
    <xsl:apply-templates/>
  </title></head>
</xsl:template>
```

Felaktig template (ej well-formed)

```
<xsl:template match=”title”>
  <head><title>
  <xsl:apply-templates/>
</xsl:template>
```

Mer om XPath

- Mönstren i ett match-attribut till ett xsl:template-element är XPath-mönster.
- De används huvudsakligen för att identifiera en eller flera delar (element, attribut mm) i ett XML-dokument.
- XPath är en W3C-rekommendation
- Det finns 7 olika nod-typer i XML-dokument som kan kommas åt via XPath-uttryck.
  - Root-noden
  - Textnoder
  - Kommentarnoder
  - Processinstruktionsnoder
  - Elementnoder
  - Attributnoder
  - Namespacenoder

XPath: Root-noden

- Uttrycket i XPath som matchar root-noden (dvs dokumentet) är ”/”.
- Innehåller alltid exakt en elementnod som barnelement. Kan innehålla kommentar- och processinstruktionsnoder.
- Jämför filsystemet i Unix eller absoluta sökvägar i URLar.

Rootnoden i XPath

```
<xsl:template match=”/”>
  <xsl:apply-templates/>
</xsl:template>
```

XPath: Elementnoder

- Elementnoder matchas med sitt namn.
- Det går att matcha flera element i samma template genom att separera elementnamnen med ”|”

Elementnoder i XPath

```
<xsl:template match=”party”>
  <xsl:apply-templates/>
</xsl:template>
```

Fler matchningar i samma template

```
<xsl:template match=”date|title”>
  <xsl:apply-templates/>
</xsl:template>
```

XPath: Attributnoder

- Attribut kan matchas genom @attributnamn.
- Notera skillnaden mellan food[@type] och food/@type till höger
- Exemplen utgår från kontextnoden

Väljer food-element som har ett type-attribut

<xsl:apply-templates select=”food[@type]” />

Väljer type-attribut ingående i food-element

<xsl:apply-tempates select=”food/@type” />

Väljer food-element som har ett type-attributmed värdet ”dairy”.

<xsl:apply-tempates select=”food[@type=’dairy’]” />

XPath: Kommentarer processinstruktioner, text

- Det går även att matcha kommentarnoder, processinstruktionsnoder och textnoder genom speciella XPath-funktioner.

Kommentarer

```
<xsl:template match="comment()" />
```

Processinstruktioner

```
<xsl:template match="processing-
  instruction()" />
```

Textnoder

```
<xsl:template match="text()" />
```

XPath: Hierarkier och wildcards

- Det går att matcha noder relativt deras position gentemot andra noder.
- ”Samtliga noder” uttrycks med *

Title-element som är barn till party-element

<xsl:template match=”party/title” />

Title-element som är ättlingar till party-element

<xsl:template match=”party//title” />

Alla element som är barn till party-element

<xsl:template match=”party/*” />

<xsl:value-of>

- Med <xsl:value-of> går det att plocka ut ”värdet” av ett element eller ett attribut.
- Kontextnodsbyte görs EJ.
- <xsl:value-of> har ett attribut, ”select”, vilket har ett XPath-värde.
- Sökvägen utgår från noden där matchningen görs, dvs kontextnoden.

```
<xsl:template match=”party”>
  <html>
    <head><title>
      <xsl:value-of select=”title”/>
    </title></head>
  </html>
</xsl:template>
```

<xsl:value-of>: ”.” och ”..”

- ”.” matchar den nod i input-trädet i vilken matchningen görs, dvs kontextnoden.
- ”..” matchar noden ovanför kontextnoden.
- Jfr unix och dos.

```
<xsl:template match=”starttime”>
  Fest mellan
  <xsl:value-of select=”.”/>
  -
  <xsl:value-of select=”../endtime”/>
</xsl:template>
```

Villkor:<xsl:if>

- XSLT stöder även vissa typer av villkor
- <xsl:if> har ett attribut ”test” som evalueras till sant eller falskt.
- Om sant så utföres innehållet i tagen, annars inte

```
<xsl:if test=”.[@type=’dairy’]” >
  <xsl:apply-templates/>
</xsl:if>
```

Villkor:<xsl:choose>

- <xsl:choose> används för if-then-else-konstruktioner.
- För ”if”-delen används <xsl:when>
- För ”else”-delen används <xsl:otherwise>

```
<xsl:choose>
  <xsl:when test=”.[@type=’dairy’]” >
    <xsl:value-of select=”foo”/>
  </xsl:when>
  <xsl:when test=”.[@type=’meat’]” >
    <xsl:apply-templates select=”bar”/>
  </xsl:when>
  <xsl:otherwise >
    <xsl:apply-templates/>
  </xsl:otherwise>
</xsl:choose>
```

<xsl:element> och <xsl:attribute>

- Ibland kan det vara önskvärt att använda en speciell xslt-konstruktion för att sätta in ett element i resultat-trädet.
- T.ex. om man vill sätta in ett attribut vars värde måste evalueras.
- Detta kan göras med <xsl:attribute>

```
<xsl:element name=”a”>
  <xsl:attribute name=”href”>
    <xsl:value-of select=”gurka”/>
  </xsl:attribute>
</xsl:element>
```

Attribute Value Templates

- Ett smidigt sätt att inkludera värden i ett attribut i resultatträdet utan att använda xsl:attribute är att stoppa in en XPath inom {}.
- I resultatträdet evalueras värdet inom klamrarna.

FEL (<> får ej finnas i ett attribut)

```
<xsl:template match=”foo”>
  <a href=”<xsl:value-of select=’gurka’/>”>gurka </a>
</xsl:template>
```

RÄTT

```
<xsl:template match=”foo”>
  <a href=”{gurka}”>gurka</a>
</xsl:template>
```

<xsl:for-each>

- Med <xsl:for-each> kan man iterera över noder som väljs med ett select-attribut.

```
<xsl:template match=”foo”>
  <xsl:for-each select=”bar”>
    <xsl:value-of select=”gurka”/>
  </xsl:for-each>
</xsl:template>
```

<xsl:copy>

- <xsl:copy> kopierar den aktuella noden till resultatträdet. Barn-element eller attribut följer inte med.
- Om noden är en elementnod eller root-nod kan noden ges ett innehåll med en template.

```
<xsl:template match=”foo|bar”>
  <xsl:copy select=”gurka”/>
</xsl:template>
```

<xsl:copy-of>

- <xsl:copy-of> kopierar ett trädfragment från input-trädet till resultatträdet.
- Trädfragmentet lokaliseras med ett select-attribut.
- Till skillnad från <xsl-copy> kopieras även attribut, barnelement och ättlingar.

```
<xsl:template match=”foo|bar”>
  <xsl:copy-of select=”gurka”/>
</xsl:template>
```

<xsl:text>

- <xsl:text> kan användas för att placera text i resultat-trädet.
- Skillnaden är att man kan få in exempelvis & och < i resultatträdet istället för deras entitetsreferenser vilka annars hamnar i resultatträdet, samt whitespace.

```
<xsl:template match=”foo”>
  <xsl:text>
     Hejhopp &lt; &gt;
  </xsl:text>
</xsl:template>
```

<xsl:output>

- Med <xsl:output> som är barnelement till <xsl:stylesheet> kan man styra sådant som doctype, kodning, indrag och outputmetod för resultatträdet.

```
<xsl:output
  method=”html”
  standalone=”no”
  doctype-system=”party.dtd”
  indent=”yes”
  media-type=”text/html”/>
```

<xsl:number/>, <xsl:counter/>

- <xsl:number> och <xsl:counter> kan användas för att skapa numreringar.
- <xsl:number opererar på input-trädet medan <xsl:counter> opererar på resultat-trädet
- Default är att alla noder av samma nodtyp och med samma nodnamn som den aktuella noden räknas.

```
<xsl:template match=”foo”>
  Foo-element nummer <xsl:number/>
</xsl:template>
```

generate-id()

- Funktionen generate-id() används för att skapa en unik identifierare för ett element i källträdet.
- För varje gång funktionen anropas för ett visst element kommer samma identifierare skapas.
- Om/när funktionen anropas för ett annat element skapas en annan unik identifierare.
- Bra för länkar/referenser inom ett dokument, t.ex. en länkad innehållsförteckning.

```
<xsl:value-of select="generate-id()"/>
```

Default-templates

- Om inga templates explicit anges för ett visst mönster finns ett antal default-templates.

Attribut- och textnoder

```
<xsl:template match=”text()|@*>
  <xsl:value-of select=”.”/>
</xsl:template>
```

Root- och elementnoder

```
<xsl:template match=”/|*”>
  <xsl:apply-templates/>
</xsl:template>
```

Processinstruktion- och kommentarnoder

```
<xsl:template match=”processing-instruction()|
   comment()”/>
```

Var sker transformerna?

- I en web-miljö finns tre alternativ för var och när transformerna ska ske.
  - När dokumentet skapas på servern. T.ex. med Xalan vid en kommandoprompt.
  - På servern precis när dokumentet ska skickas till klienten. T.ex. med Cocoon, en servlet som kan användas i t.ex. Apache-webservern, eller ett API från ett programmerinsspråk.
  - På klienten, i dagsläget IE>=6 och Mozilla/Firefox >=1.0 samt numera även Safari, Opera och Chrome.
- Vilka för- och nackdelar finns med respektive alternativ?

Mer information

- Xpath innehåller mycket mer, bland annat en hel del aritmetiska operationer.
- I XML in a nutshell finns bra referensmanualer till Xpath och XSLT.
- Mycket bra exempel på mer avancerade xpath-uttryck på http://www.w3.org/TR/xpath. STUDERA GÄRNA DESSA!