Xerces parseren.

Vi vil i dette afsnit se på nogle af de muligheder man får hvis man downloader Xerces parseren. Denne parser kan downloades fra http://www.apache.org. Den kan hentes både i en C (C++) udgave og en Java udgave. Her vil vi se på Java udgaven som er den egentlige eller oprindelige parser. Det skal straks siges at Xerces ikke består af een, men af mange forskellige parsere og processorer til forskellige opgaver!

Det materiale som downloades består primært af 5 Java JAR filer bl.a. xmlParserAPIs.jar, xercesImpl.jar og xercesSamples.jar. En JAR fil er en samling af Java klasser.

I det følgende vil vi kun se på de færdige Java klasser (programmer) som følger med parseren! Vi vil altså ikke selv programmere med Xerces parseren!

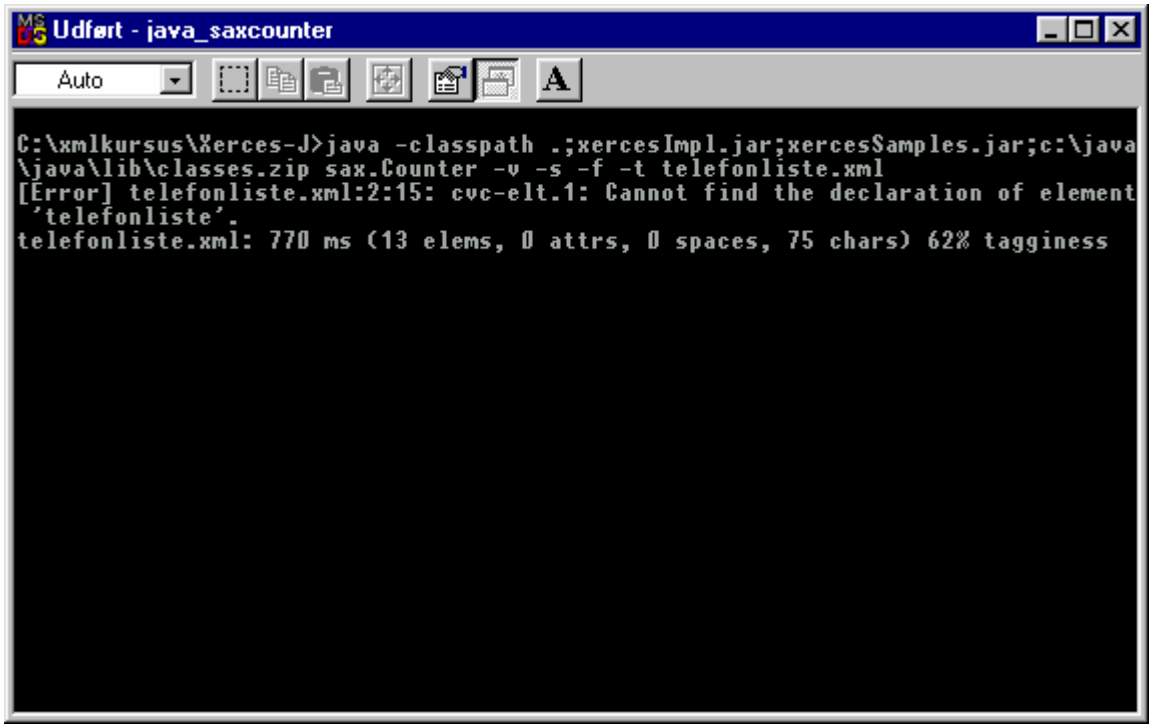
sax.Counter:

Counter er en simpel SAX parser som kan validere med DTD eller XSD skema, undersøge om dokumentet er velformet, beregne dokumentets tagginess (hvor stor en procent udgør de rene tags!) mv.

Vi kan køre programmet på denne måde med en .bat fil der ligger i samme mappe som JAR filerne (NB stien til classes.zip skal så tilpasses til din egen maskine!):

java -classpath .;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip sax.Counter -v -s -f -t telefonliste.xml

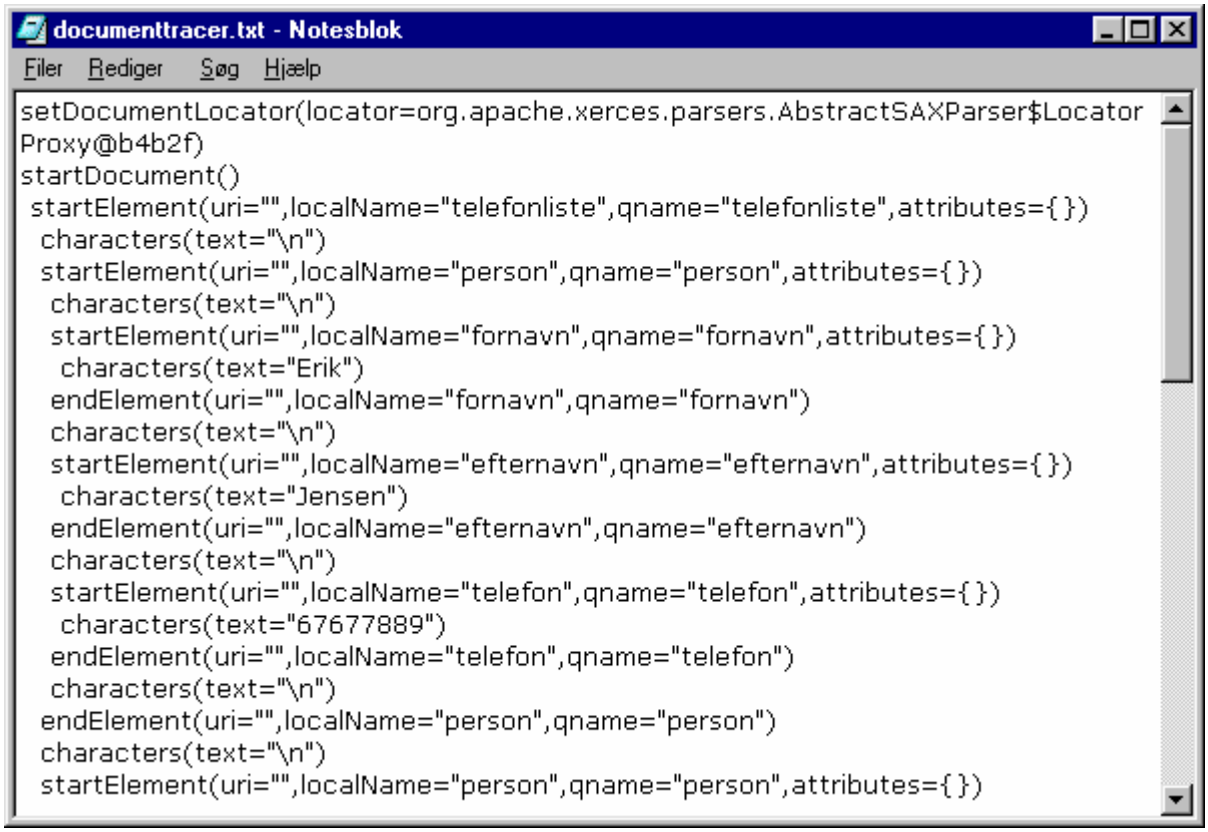
-v validerer med DTD og –s –f validerer med et XSD skema. –t udskriver tagginess:



Problemet her at vi slet ikke har angivet noget skema til XML dokumentet!

sax.DocumentTracer:

Programmet DocumentTracer er en SAX parser som leverer en meget detaljeret analyse og gennemgang af det angivne XML dokument:



Programmet kan køres ved at køre dette bat program:

java -classpath .;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip sax.DocumentTracer telefonliste.xml > documenttracer.txt

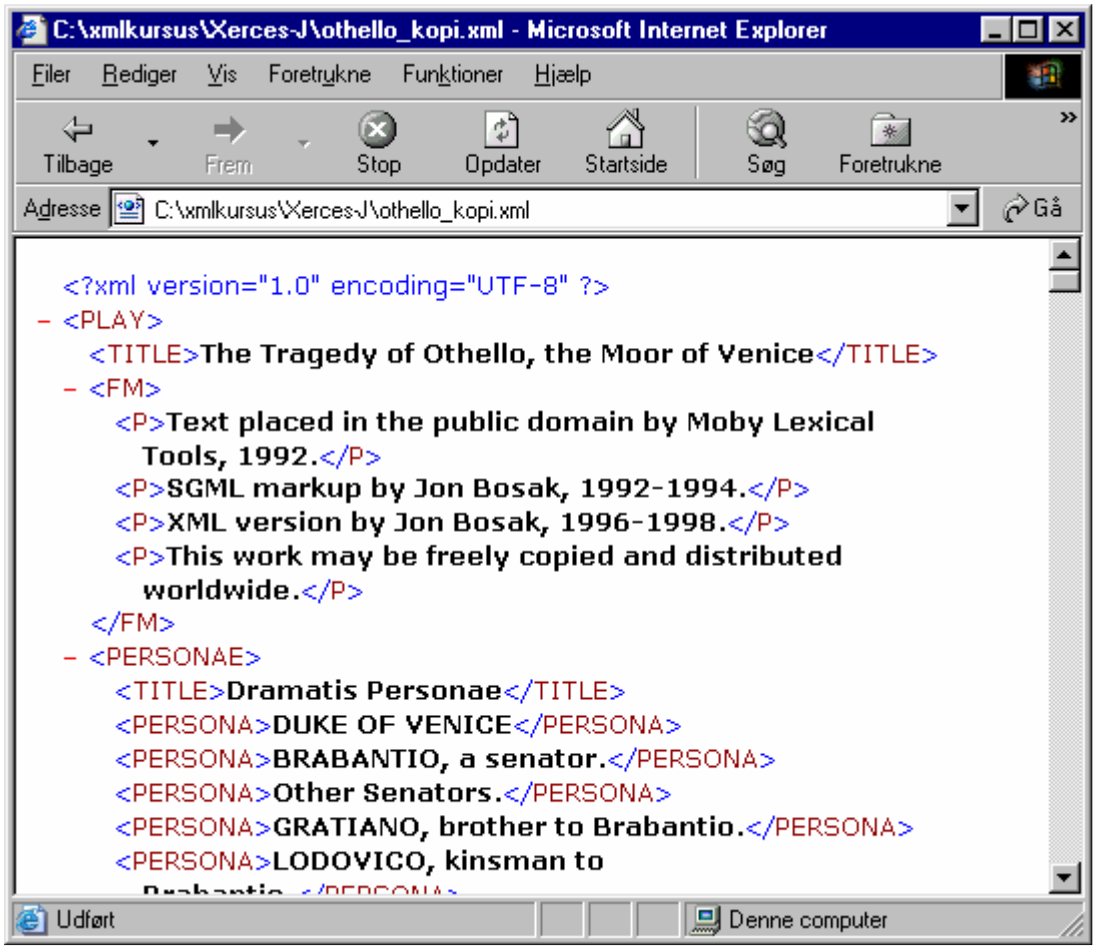
Output bliver omdirigeret til en tekstfil!

sax.Writer:

Programmet sax.Writer er værdifuldt i Java sammenhæng fordi det udskriver det XML dokument som gives som input i kommandolinjen. Denne funktion – som i Microsoft parseren MSXML bruges med egenskaben .xml – skal i Java implementeres med en Writer klasse. Vi kan starte programmet således:

java -classpath .;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip sax.Writer ny_othello.xml > othello_kopi.xml

Java programmet skriver så en kopi af dokumentet til othello_kopi.xml:



dom.GetElementsByTagName:

Klassen dom.GetElementsByTagName er en DOM parser som fungerer som en XPATH maskine. Programmet startes med et XML dokument, et ønsket element og evt en ønsket attributværdi for dette element! Man kan altså søge i et f. eks. stort XML dokument med dette program!

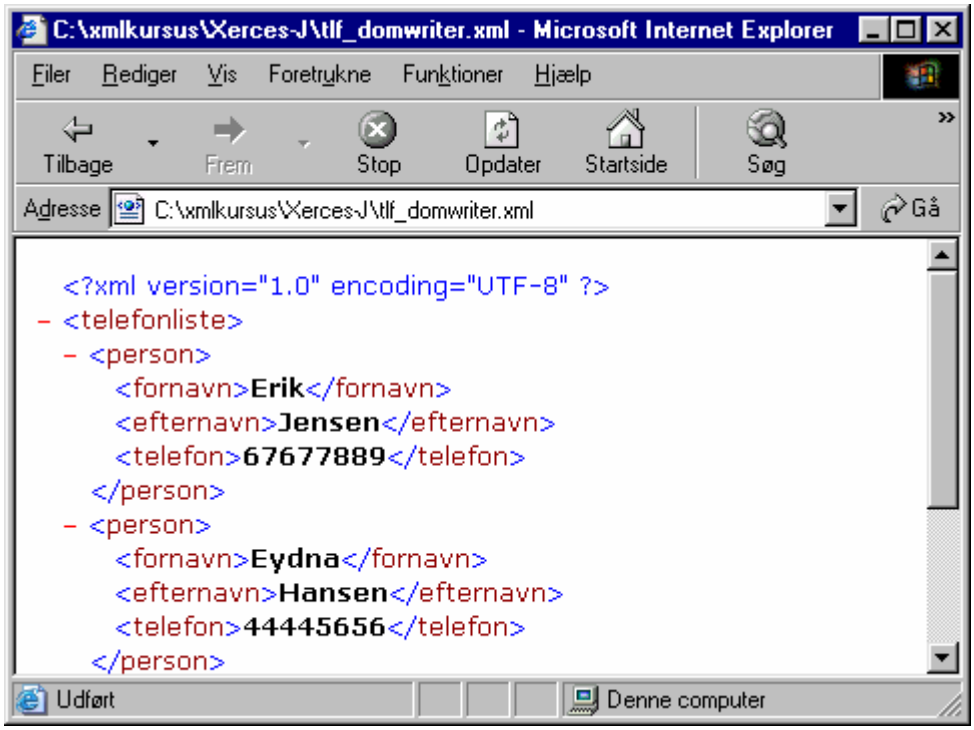
Programmet kan startes på denne måde f. eks. hvor vi beder om alle fornavn elementer i telefonliste.xml:

java -classpath .;xmlParserAPIs.jar;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip dom.GetElementsByTagName -e fornavn telefonliste.xml > tlf_dom.txt

dom.Writer:

Dette program som er en DOM parser udskriver et dokument med DOM metoder:

java -classpath .;xmlParserAPIs.jar;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip dom.Writer telefonliste.xml > tlf_domwriter.xml



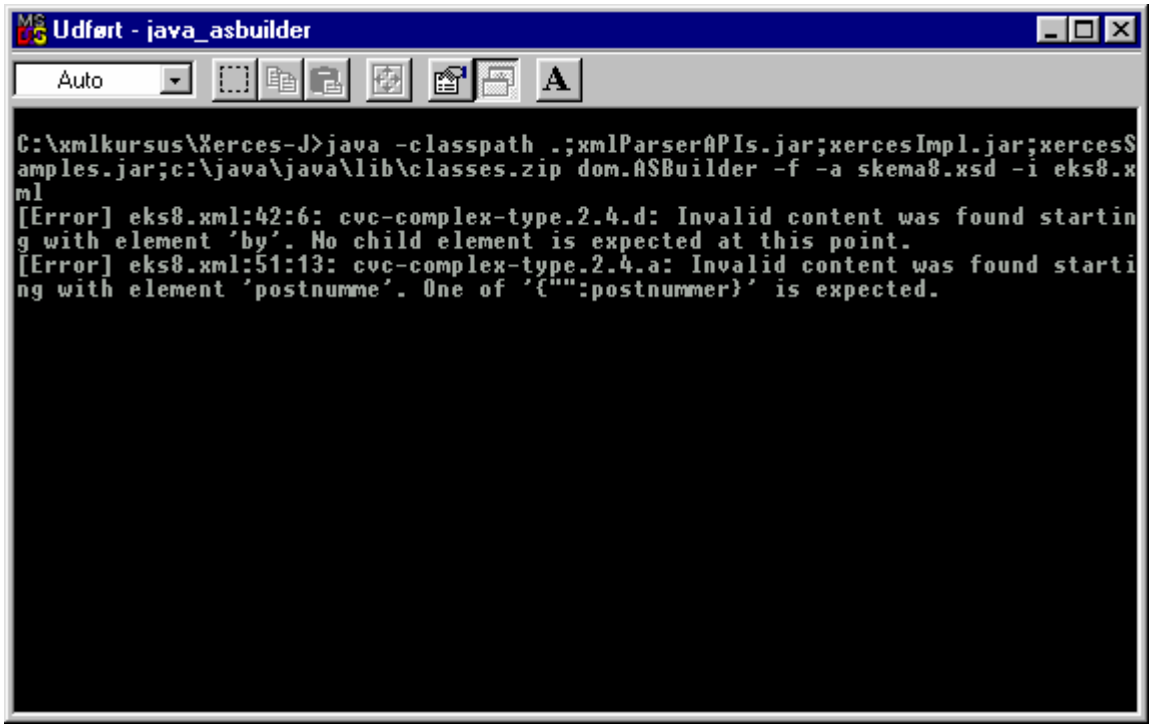
dom.ASBuilder:

Programmet ASBuilder er en Java skema **validerings** maskine som kan validere over for forskellige slags skemaer. Programmet kan køres således:

java -classpath .;xmlParserAPIs.jar;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip dom.ASBuilder -f -a skema8.xsd -i eks8.xml

Her prøver vi at validere XML dokumentet eks8.xml i forhold til et XSD skema. –f betyder her full checking.

Det viser sig at XML dokumentet ikke er gyldigt:



Vi har udkommenteret et element postnummer for bevidst at lave en skema fejl i dokumentet! Desuden har vi indsat et helt nyt element by som slet ikke findes i XSD skemaet!

ASBuilder fungerer altså som et program der kan validere et XML dokument.

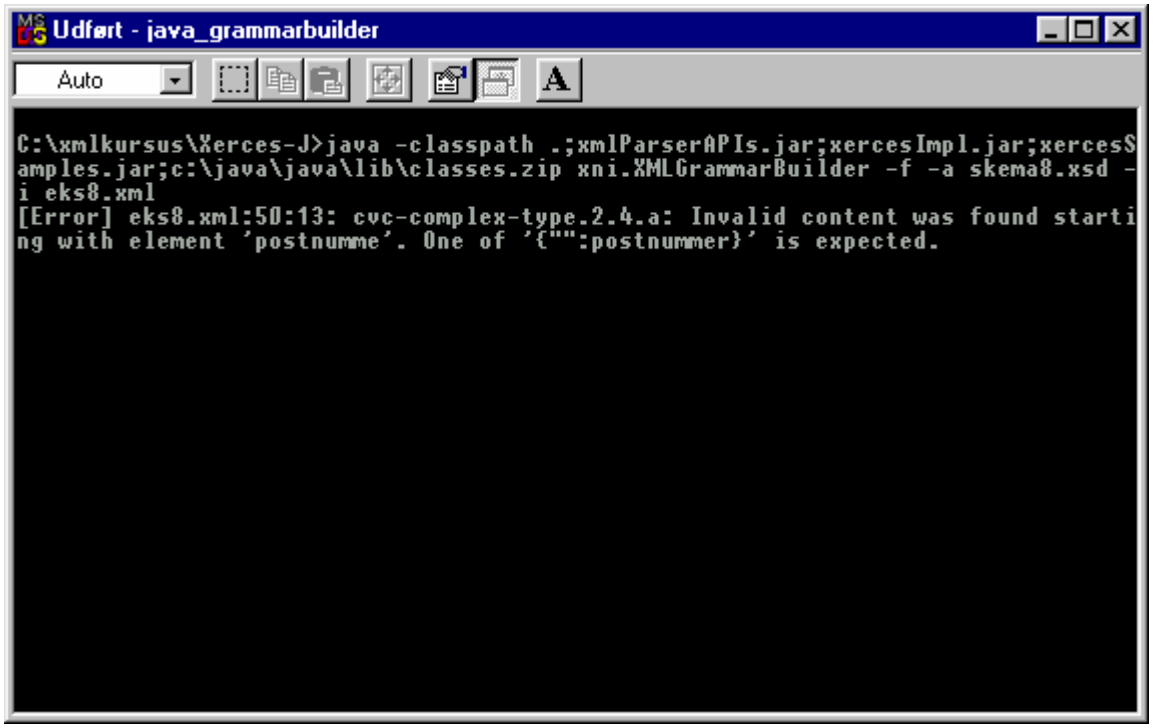
XMLGrammarBuilder:

Dette program gør grundlæggende det samme som det forrige – validerer et XML dokument mod et eller flere skemaer – dels DTD dels XSD skemaer. Det er en **moderniseret** udgave af det forrige:

java -classpath .;xmlParserAPIs.jar;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip xni.XMLGrammarBuilder -f -a skema8.xsd -i eks8.xml

Som det ses tilhører det et specielt namespace under Xerces (egentligt Xerces2) der hedder xni – Xerces Native Interface. XMLGrammarBuilder **kompilerer** de forskellige skemaer inden det går i gang med at validere XML dokumenterne!

Hvis vi laver fejl i XML dokumentet får vi en fejl meddelelse:

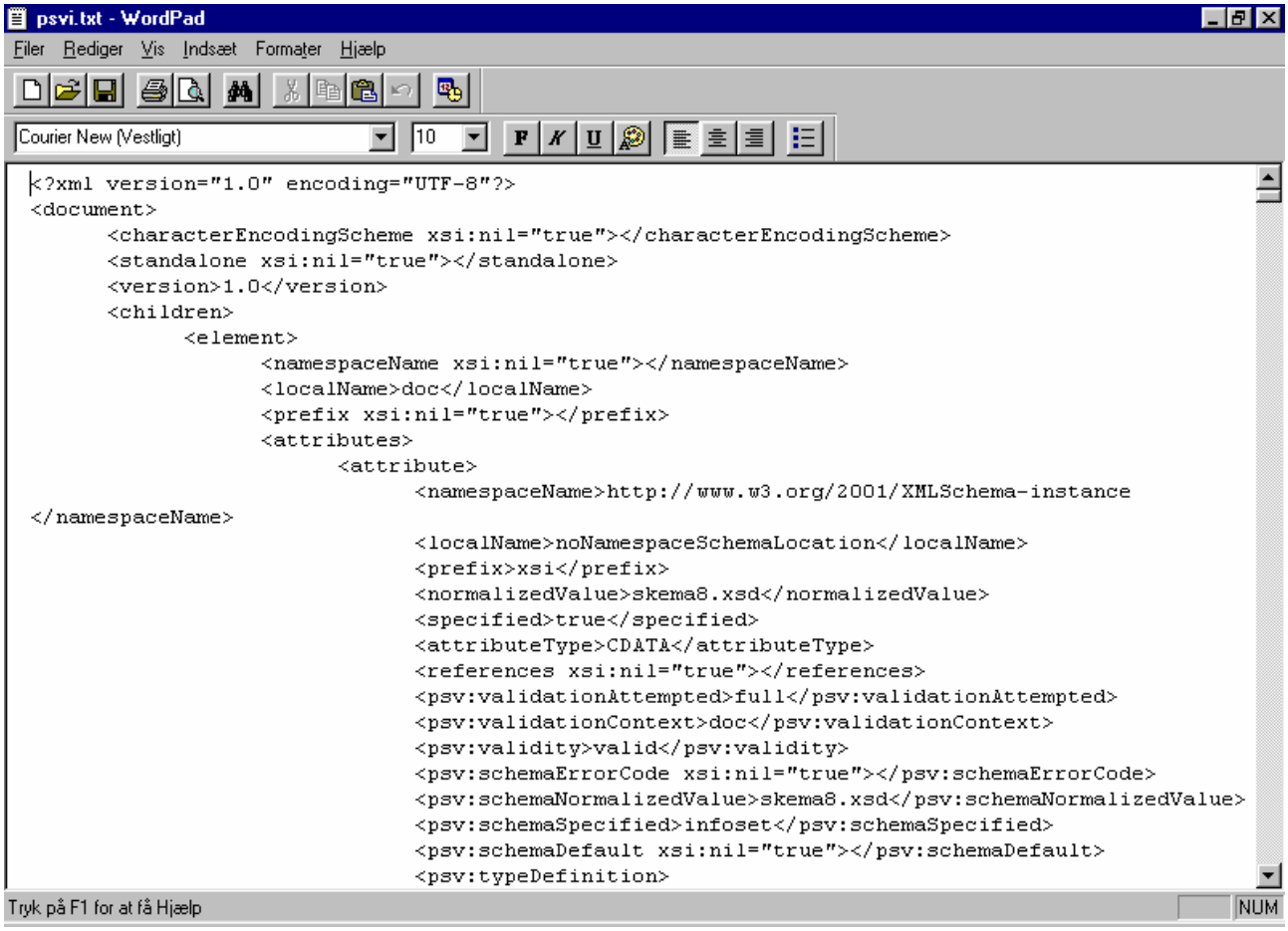


Man kan bruge en parameter –d og en liste af efterfølgende DTD skemaer. Efter parameteren –i kan også følge eventuelt mange instans dokumenter – som altså kan valideres i et hug!

PSVIParser:

PSVI er **Post Schema Validation Infoset** – d.v.s. en samlet DOM repræsentation af både XML og XSD dokumentet! Dette Infoset produceres af en PSVIParser efter at XML dokumentet er blevet valideret. Dette Infoset er det mest omfattende materiale der overhovedet kan produceres for et XML dokument! Den producerede fil – et slags XML dokument – men ikke velformet – er derfor også meget lang!!

Her er starten på et Infoset dokument for eks8.xml:




```
<?xml version="1.0" encoding="UTF-8"?>
<document>
      <characterEncodingScheme xsi:nil="true"></characterEncodingScheme>
      <standalone xsi:nil="true"></standalone>
      <version>1.0</version>
      <children>
            <element>
                  <namespaceName xsi:nil="true"></namespaceName>
                  <localName>doc</localName>
                  <prefix xsi:nil="true"></prefix>
                  <attributes>
                        <attribute>
                              <namespaceName>http://www.w3.org/2001/XMLSchema-instance
</namespaceName>
                              <localName>noNamespaceSchemaLocation</localName>
                              <prefix>xsi</prefix>
                              <normalizedValue>skema8.xsd</normalizedValue>
                              <specified>true</specified>
                              <attributeType>CDATA</attributeType>
                              <references xsi:nil="true"></references>
                              <psv:validationAttempted>full</psv:validationAttempted>
                              <psv:validationContext>doc</psv:validationContext>
                              <psv:validity>valid</psv:validity>
                              <psv:schemaErrorCode xsi:nil="true"></psv:schemaErrorCode>
                              <psv:schemaNormalizedValue>skema8.xsd</psv:schemaNormalizedValue>
                              <psv:schemaSpecified>infoset</psv:schemaSpecified>
                              <psv:schemaDefault xsi:nil="true"></psv:schemaDefault>
                              <psv:typeDefinition>
```

Som det måske kan anes indeholder dokumentet **dels** en validerings rapport **dels** en meget minutiøs gennemgang af dokumentet!

PSVIParser kan **ikke** køres som **standalone** program men kan køres som en parameter til f. eks. en Java Writer på denne måde:

java -classpath .;xmlParserAPIs.jar;xercesImpl.jar;xercesSamples.jar;c:\java\java\lib\classes.zip sax.Writer -v -s -p xni.parser.PSVIParser eks8.xml > psvi.txt

Som det kan ses kan en Writer opstartes med en bestemt **navngivet** Java parser med parameteren –p som vist her! –v og –s angiver validering. Det namespace (**package**) som hedder **xni** er et særligt Xerces namespace med nye, forbedrede, eksperimentelle klasser og programmer. xni anvender blandt andet **pipes** (kommunikations rør) som i dette eksempel hvor Writer'en går **igennem** en PSVIParser (eller omvendt om man vil)!