

Webbutveckling XML-API

Assignment

Emma Jonsson

Webbutveckling – Webbplatsdesign
VT 2012

Innehåll

1. Inledning	4
2. Mindre uppgifter för att lära sig de olika API'erna	4
2.1 Första uppgiften – Introduktion i PHP (se Bilaga A och B).....	4
2.1.1 1A Fråga 1 - Diskutera varför varje ny select-box och textinput-box måste döpas olika.....	4
2.1.2 1B Fråga 1 - Varför är det svårt skriva ut datan i arrayen i en table?.....	4
2.1.3 1B Fråga 2 - Om vi skulle vilja lägga till fler selectboxar för ytterligare data i arrayen, vilka steg skulle bli nödvändiga?.....	4
2.1.4 1B Fråga 3 - Hur skulle koden förändras om vi gjorde två separata arrayer, en för tillverkarna och en för lastbilarna, och skulle denna förändring göra det enklare eller svårare att skapa tabellen?.....	4
2.2 Andra uppgiften – SAX (se Bilaga C och D).....	4
2.2.1 Discuss why formatting is so complicated with a SAX API.	4
2.2.2 Explain the thinking behind your styling in the table and your parsing code.....	5
2.3 Tredje uppgiften – DOM (se Bilaga C och E).....	5
2.3.1 Discuss why formatting is so much simpler using DOM than with the SAX API.	5
2.3.2 Explain the thinking behind your styling in the table and your parsing code.....	5
2.4 Fjärde uppgiften – DOM-validering med DTD (se Bilaga C, F och G).....	5
2.4.1 Detail the two well-formedness errors introduced in the file.	5
2.4.2 Detail the validation errors introduced in the file.....	6
2.5 Femte uppgiften – XPATH (se Bilaga H)	6
2.5.1 Discuss why working with XPATH can be considered both simpler and more difficult (for different scenarios) when compared to formatting using DOM navigation or the SAX API.	6
2.5.2 Explain the thinking behind your styling in the table and your parsing code.....	6
2.6 Sjätte uppgiften – XSLT (se Bilaga I och J).....	6
2.6.1 Discuss why working with XSLT can be seen as both more complicated and easier than working with the DOM programming.	7
2.6.2 Explain the thinking behind your styling in the table and your parsing code.....	7
2.7 Sjunde uppgiften – XSLT apply templates (se Bilaga I och K)	7
2.7.1 Discuss why working with Apply Templates can be seen as both more complicated and easier than working with the DOM-programming or XSLT navigation.	7
2.7.2 Discuss the use of grouping and why grouping can be very useful in many applications.....	7
2.7.3 Explain the thinking behind your styling in the table and your parsing code.....	7
3. Huvuduppgiften	8
3.1 Kort fakta om den utdelade XML-filen.....	8
3.1.1 What kind of document is it, database centric (like tables in a database with a simple and well known structure) or document centric (like a web page or a word processor document with unknown structure). ...	8
3.1.2 Who designed the file format, and when was it designed?.....	8
3.1.3 What is the format called and what is it used for?.....	8
3.1.4 Are there any controversies surrounding the format?	8
3.1.5 Is the format made for any specific type of application?.....	8
3.1.6 Is the format an official standard (such as ISO) and/or is there a DTD or schema for the file format? .	8
3.1.7 Are there specific API-s already made for working with this type of file?	8
3.1.8 Are there other related formats?.....	8
3.1.9 Are there other versions of this format, if so what are the chief differences?	8

3.1.10 Is there anything else that is interesting about this file?.....	9
3.1.11 Which form of presentation and which form of API is best for this type of file?	9
3.1.12 Is there any book or other form of documentation that gives any type of information about this type of file?	9
3.1.13 Are there any other interesting things regarding this type of file that will be interesting for a reader of the report?	9
3.2 Lösningen (Omvandling från XML till HTML)	9
3.3 Extra finesser.....	10
3.3.1 Sortering.....	10
3.3.2 DTD	10
3.3.3 Byte av layout	10
3.4 Det lilla som borde gjorts annorlunda	10
4. Reflektioner.....	10
Bilaga A. Uppgift 1A (Simpel layout med PHP).....	a
Bilaga B. Uppgift 1B (Nästlade arrayer i tabell)	a
Bilaga C. XML-filen som parsas i alla mindre uppgifter	a
Bilaga D. Uppgift 2 (SAX).....	a
Bilaga E. Uppgift 3 (DOM)	a
Bilaga F. Uppgift 4 (PHP-filen som användes)	a
Bilaga G. Uppgift 4 (DTD-filen som användes).....	a
Bilaga H. Uppgift 5 (XPATH)	a
Bilaga I. PHP-filen som används i uppgift 6 och 7	a
Bilaga J. Uppgift 6 (XSLT).....	a
Bilaga K. Uppgift 7 (XSLT apply templates).....	a
Bilaga L. XML-filen till huvuduppgiften.....	a
Bilaga M. PHP-filen till huvuduppgiften	a
Bilaga N. XSL-filen till huvuduppgiften	a
Bilaga O. DTD-filen till huvuduppgiften	a
Bilaga P. PHP-filen med CSS-arrayen.....	a
Bilaga Q. PHP-filen som skapar en cookie	a
Bilaga R. CSS-filen till huvuduppgiften	a
Bilaga S. Sluresultatet.....	a

1. Inledning

Denna rapport är ett resultat av ett antal uppgifter i kursen Webbutveckling XML-API. Första delen handlar om att lära sig de olika API'erna som finns till hands för att parse XML-filer och andra delen handlar om hur den större betygsgrundande uppgiften löstes. Rapporten avslutas med några korta reflektioner om kursen och dess uppgifter som sådant.

2. Mindre uppgifter för att lära sig de olika API'erna

Kursen startade alltså med ett antal mindre uppgifter som skulle utföras i tur och ordning för att bekanta sig med de olika verktyg som man sedan skulle använda sig utav för att kunna lösa den stora huvuduppgiften. Varje mindre uppgift åtföljdes av ett antal frågor som redovisas i detta kapitel. Varje uppgift resulterade även i en källkod som bifogas som bilaga.

2.1 Första uppgiften – Introduktion i PHP (se Bilaga A och B)

Den första uppgiften var uppdelad i två delar. Första delen gick ut på att skapa en hemsida där man kunde välja olika stilar på ett stycke text som man skrev in. Funktionen skulle kontrolleras av en array via ett formulär. I andra delen skulle man skriva ut innehållet i en nästlad array i en tabell. Nedan följer ett antal frågor på vardera deluppgift.

2.1.1 1A Fråga 1 - Diskutera varför varje ny select-box och textinput-box måste döpas olika.

Anledningen är att för att kunna kalla på ett unikt värde måste man använda sig utav unika namn. Utan unika namn kan datorn inte urskilja unika data från varandra.

2.1.2 1B Fråga 1 - Varför är det svårt skriva ut datan i arrayen i en table?

Detta är svårt för att uppgiften omfattar ett stort antal nästlade tabeller som därmed är svåra att lokalisera sig i. Det gäller att veta vart man befinner sig i "trädet" för att utforma sin frågeställning/kontroll korrekt.

2.1.3 1B Fråga 2 - Om vi skulle vilja lägga till fler selectboxar för ytterligare data i arrayen, vilka steg skulle bli nödvändiga?

1. Först hade man varit tvungen att skapa en array som innehåller alternativen för selectboxen.
2. Sedan hade man varit tvungen att skapa själva selectboxen i HTML-koden.
3. Därefter hade man fått iterera fram alternativen från arrayen och placera dessa i selectboxen.
4. Och till sist hade man fått placera ut resultatet av valet från selectboxen så att det faktiskt händer något när man gjort ett val.

2.1.4 1B Fråga 3 - Hur skulle koden förändras om vi gjorde två separata arrayer, en för tillverkarna och en för lastbilarna, och skulle denna förändring göra det enklare eller svårare att skapa tabellen?

Det skulle orsaka väldigt mycket redundans och man skulle dessutom tappa bort kopplingen mellan tillverkningsplatsen och fordonet. Med andra ord skulle man alltså tappa kontrollen över datan och det skulle därmed bli betydligt svårare att skapa tabellen. Det går att åtgärda men kräver mer avancerad kod.

2.2 Andra uppgiften – SAX (se Bilaga C och D)

Den andra uppgiften gick ut på att parse en XML-fil med hjälp av API-verktyget SAX. Nedan följer ett antal frågor på uppgiften.

2.2.1 Discuss why formatting is so complicated with a SAX API.

Anledningen är för att du lägger in koden i funktioner vilket orsakar en onaturlig följd av flödet och gör det svårt för den som programmerar att se logiken i koden.

2.2.2 Explain the thinking behind your styling in the table and your parsing code.

Först skapades en tabell i ett HTML-dokument. Efter en stunds funderande på vilka element som tabellen skulle innehålla utformades tabellen utifrån detta. Därefter flyttades denna kod över till PHP-dokumentet och formaterades om så att den skulle fungera i PHP. Detta innebar bland annat att byta ut citationstecknen till apostrofer. Till sist placerades start- och sluttaggen för tabellen runt självaste parsnings-koden.

När HTML-biten var avklarad påbörjades kodandet i SAX. För att plocka fram de olika elementen och placera dem i sina respektive celler placerades en foreach i funktionen StartElement som gick igenom varje starttagg och med hjälp av if-satser skrev den ut taggnamn och attributvärden i celler. Funktionen charData skrev ut PCDATA (parsed character data) i sina respektive celler och funktionen endElement såg endast till att det blev en ny rad i tabellen. Ett problem löstes dock inte. Detta var att om taggen comment inte existerade skulle en tom cell skrivas ut, dvs. en cell som innehöll . Vi diskussion med handledaren blev dock kommentaren att inte hänga upp sig på detta utan gå vidare med nästa uppgift.

2.3 Tredje uppgiften – DOM (se Bilaga C och E)

Den tredje uppgiften gick ut på att parsea en XML-fil med hjälp av API-verktyget DOM. Nedan följer ett antal frågor på uppgiften.

2.3.1 Discuss why formatting is so much simpler using DOM than with the SAX API.

Detta beror på att i DOM så skriver du din tabell uppifrån och ner och kallar på varje element i tur och ordning. Detta blir lättare både att läsa och skriva då man inte måste hoppa runt så mycket i koden för att förstå den.

2.3.2 Explain the thinking behind your styling in the table and your parsing code.

Precis som i 2.2 så skapades först tabellen i HTML-kod för att sedan flyttas över till PHP-koden. Eftersom samma XML-fil som tidigare skulle parsas så användes en identisk tabell. Därefter placerades tabellens element ut allt eftersom mellan raderna som kallade på de olika elementen i XML-filen. API-verktyget DOM användes sedan i form av ett antal foreach som itererade över de olika elementen och med hjälp av if-satser skrevs varje element ut på sin respektive plats. Dock återstod samma problem som i andra uppgiften med taggen comment och det löstes inte heller med denna API.

2.4 Fjärde uppgiften – DOM-validering med DTD (se Bilaga C, F och G)

Den fjärde uppgiften gick ut på att skapa en DTD-fil som matchade ett XML-dokument och sedan undersöka hur de betedde sig och vilka felmeddelanden som kom ut när man medvetet ändrade i koden så att det blev fel. Nedan följer ett antal frågor på uppgiften.

2.4.1 Detail the two well-formedness errors introduced in the file.

På rad 9 (se Bilaga C) tog jag bort </text> vilket resulterade i följande fyra felmeddelanden:

```
2 DOMDocument::load(): Opening and ending tag mismatch: text line 10
and article in /home/StudentHome/2011/allemmjo/public_html/xml-
api/Uppg 4/example2b.xml, line: 13
```

```
2 DOMDocument::load(): Opening and ending tag mismatch: article line 6
and newspaper in /home/StudentHome/2011/allemmjo/public_html/xml-
api/Uppg 4/example2b.xml, line: 31
```

```
2 DOMDocument::load(): expected '>' in
/home/StudentHome/2011/allemmjo/public_html/xml-api/Uppg
4/example2b.xml, line: 88
```

```
2 DOMDocument::load(): Premature end of data in tag newspapers line 4
in /home/StudentHome/2011/allemmjo/public_html/xml-api/Uppg
4/example2b.xml, line: 88
```

På rad 26-27 (se Bilaga C) plockade jag ner `</text>` så att den hamnade efter `</article>` istället och därmed resulterade i ett nästlingsfel vilket resulterade i följande två felmeddelanden:

```
2 DOMDocument::load(): Opening and ending tag mismatch: text line 27
and article in /home/StudentHome/2011/allemmjo/public_html/xml-
api/Uppg 4/example2b.xml, line: 30
```

```
2 DOMDocument::load(): Opening and ending tag mismatch: article line
23 and text in /home/StudentHome/2011/allemmjo/public_html/xml-
api/Uppg 4/example2b.xml, line: 30
```

2.4.2 Detail the validation errors introduced in the file.

På rad 6 lade jag till ett extra attribut, `author="Jane Austen"`, till `article`-taggen vilket resulterade i följande felmeddelanden:

```
2 DOMDocument::validate(): No declaration for attribute author of
element article
```

2.5 Femte uppgiften – XPATH (se Bilaga H)

Femte uppgiften gick ut på att parsea en XML-fil med hjälp av API-verktyget XPATH. Nedan följer ett antal frågor på uppgiften.

2.5.1 Discuss why working with XPATH can be considered both simpler and more difficult (for different scenarios) when compared to formatting using DOM navigation or the SAX API.

Eftersom XPATH kan liknas med ett frågespråk så kan man fråga specifikt efter precis den tagg man vill få ut. Detta kan dock bli omständigt om man vill få ut all information ur ett xml-dokument som innefattar väldigt många taggar. Om man då istället använder sig utav DOM så kan den göra väldigt mycket utav arbetet åt en med hjälp av väldigt lite kod om den formuleras på rätt sätt.

Om man däremot bara vill plocka ut en liten del någonstans mitt i ett xml-dokument så kan det med DOM eller SAX bli väldigt mycket kod som krävs för att navigera sig fram dit man vill utgå ifrån medan man med en enkel rad kan välja att utgå från vilken rad man vill om man använder sig av XPATH.

2.5.2 Explain the thinking behind your styling in the table and your parsing code.

Eftersom man skulle använda sig utav samma xml-fil som tidigare återanvändes samma design på tabellen som hade använts i de tidigare uppgifterna. Start- och sluttaggen kopierades därmed över från en tidigare uppgift och därefter startades den egentligen uppgiften med att lägga in alla attribut från taggen `newspaper` genom att plocka fram och spara alla noder under sökvägen `//newspaper` i variabeln `papers`. Därefter lades attributen från taggen `article` in genom att plocka fram och spara alla noder under sökvägen `//article` i variabeln `articles`. Därefter användes precis som tidigare både `foreach` och `if`-satser för att plocka ut det man ville ha på sin respektive plats.

2.6 Sjätte uppgiften – XSLT (se Bilaga I och J)

Sjätte uppgiften gick ut på att parsea en XML-fil med hjälp av API-verktyget XSLT i en enda template. Nedan följer ett antal frågor på uppgiften.

2.6.1 Discuss why working with XSLT can be seen as both more complicated and easier than working with the DOM programming.

Det är svårare eftersom man måste vara väldigt försiktig så att man inte får några brutna taggar då det icke är tillåtet. Detta kan ibland komplicera tankegångarna en aning. Men det är samtidigt lättare eftersom man mer direkt kan komma åt alla xml-dokumentets element och attribut mer direkt utan att behöva leta sig fram med hjälp av långa krångliga adresser.

2.6.2 Explain the thinking behind your styling in the table and your parsing code.

Den första lösningen var att utgå direkt från tidningen för att skriva ut tidningsinfor och först därefter kliva neråt för att skriva ut informationen om artiklarna, men detta resulterade i trasiga taggar och därmed en trasig parsning. Efter en stunds funderande provades istället att plocka ut informationen om tidningen direkt från artikeln istället med hjälp av "../". Detta fungerade mycket bättre. Sedan återkom problemet med att tidningsinfor bara skulle skrivas ut för den första artikeln. Detta löstes hjälp av "when test='position()=1'" för första artikeln i varje tidning och en variant för resterande som inte skrev ut tidningsinformationen. Där det eventuellt saknades en tagg eller ett attribut användes istället en count() för att plocka fram alternativen.

2.7 Sjunde uppgiften – XSLT apply templates (se Bilaga I och K)

Sjunde uppgiften gick ut på att parse en XML-fil med hjälp av API-verktyget XSLT i flera stycken templates. Nedan följer ett antal frågor på uppgiften.

2.7.1 Discuss why working with Apply Templates can be seen as both more complicated and easier than working with the DOM-programming or XSLT navigation.

Precis som i uppgift 6 så är det svårare eftersom att man inte får ha några brutna taggar vilket kan komplicera tankegångarna. Men det är samtidigt lättare eftersom man mer direkt kan komma åt alla xml-dokumentets element och attribut. Jämfört mot uppgiften tidigare så är det ännu lättare att plocka ut alla enskilda element och attribut eftersom man inte längre utgår ifrån en specifik punkt och måste gräva dig neråt med långa adresser utan du kan istället kalla på en annan template som inte har någon långt formulerad adress. Den är dock inte lika logisk som den förra uppgiften då allt listades i tur och ordning utan istället så hoppar man nu fram och tillbaka i koden mellan olika templates vilket lätt kan bli lite förvirrande.

2.7.2 Discuss the use of grouping and why grouping can be very useful in many applications.

Enkel gruppering är inte svår att förstå sig på men den är svårare att implementera i koden eftersom man får tänka om lite i vart man placerar de olika elementen för att få det att fungera som man tänkt. Gruppering kan naturligtvis vara väldigt användbart om man vill lista något i en tabell efter en viss ordning, dvs. sortera efter olika villkor.

2.7.3 Explain the thinking behind your styling in the table and your parsing code.

Först skapades en template för varje steg i trädet, men sedan konstaterades det att man inte behövde detta utan istället kan hänvisa direkt till det element som man vill utgå ifrån. Detta resulterade i att parsningen istället fick utgå ifrån article med villkoret position()=1 och position()>1 för att kunna skriva ut tidningsinformationen enbart för den första artikeln på varje tidning precis som tidigare. Heading och text fick bli egna templates medan attributen time och description fick ligga kvar i artikeln eftersom man behövde använda sig utav en choose på både dessa då datan inte alltid fanns.

Till sist när sorteringen skulle implementeras så fick man lyfta ut tidningsinformationen för att kunna få sorteringen att fungera och lägga detta i en egen template. Dock behölls skillnaden på första och resterande artiklar eftersom den första inte skulle hamna på ny rad medans de andra skulle detta.

3. Huvuduppgiften

Huvuduppgiften bestod i att presentera innehållet i en slumpvis utdelad XML-fil med hjälp av HTML, PHP och det API man föredrog att använda sig utav för att parse XML-filens innehåll på lämpligast sätt.

3.1 Kort fakta om den utdelade XML-filen

I detta kapitel kommer ett antal obligatoriska frågor som angivits i uppgiftsbeskrivningen att listas tillsammans med sitt respektive svar.

3.1.1 What kind of document is it, database centric (like tables in a database with a simple and well known structure) or document centric (like a web page or a word processor document with unknown structure).

Denna XML-fil är en databascentrerad fil. Detta eftersom det är ett format som används för att fylla på och utbyta information emellan databaser som innehåller ord för en ordlista.

3.1.2 Who designed the file format, and when was it designed?

Skaparna av XDXF är av allt att döma fyra personer med följande alias: dict, jaesar, mrcoder1234 och thelivingone enligt sourceforge (u.å.).

3.1.3 What is the format called and what is it used for?

Formaten kallas för XDXF som står för 'The Extensible (XML) Dictionary Exchange Format' (Google Groups, u.å.) och används för att utbyta ord mellan ordlistor enligt Revdanica (u.å.).

3.1.4 Are there any controversies surrounding the format?

Inte vad som gått att finna på Internet.

3.1.5 Is the format made for any specific type of application?

Ja, formaten är tänkt att läsas in av en ordboksapplikation så att man kan ta del utav ordbokens innehåll. Dels återfinns detta online på bl.a. The ARTFL Project (u.å.) och dels listar Wikipedia (u.å.) ett antal applikationer både till dator och till mobiltelefon som använder sig utav XDXF-filer.

3.1.6 Is the format an official standard (such as ISO) and/or is there a DTD or schema for the file format?

Ja, det måste vara en typ av officiell standard då den används till så många applikationer, men det verkar inte vara en ISO-standard.

Ja det finns en DTD att ladda ner på sourceforge (u.å.) att hämta.

3.1.7 Are there specific API-s already made for working with this type of file?

Eftersom de flesta länkar som man hänvisas till när man söker på XDXF är brutna så är det mycket svårt att få fram någon information alls, och hittills har inga tecken funnits som tyder på att det finns något fastslaget API som bör användas för att parse XDXF.

3.1.8 Are there other related formats?

Ja, dicML är ett markup-språk som har ett liknande syfte som XDXF men för tillfället är inget utav dem helt specificerade (Wikipedia, u.å.).

3.1.9 Are there other versions of this format, if so what are the chief differences?

Nej, inte vad som gått att finna på Internet.

3.1.10 Is there anything else that is interesting about this file?

Detta format är en uppkomst ifrån ett projekt (Revdanica, u.å.) som verkar ha samarbetats fram utav ett antal programmerare (sourceforge, u.å.) under namnet Revdanica.

3.1.11 Which form of presentation and which form of API is best for this type of file?

Om man har tänkt lista filens innehåll i en tabell som denna uppgift har krävt så bör det rimligtvis vara enklast att använda sig utav XSLT. Detta för att de taggar som XDXF-filen innehåller ofta är väldigt ostrukturerade och därför lämpar det sig troligtvis inte att skriva ut deras innehåll direkt uppifrån och ner då detta inte skulle bli särskilt strukturerat och logiskt.

3.1.12 Is there any book or other form of documentation that gives any type of information about this type of file?

Ja, det finns en bok som heter "XDXF" som är skriven av Ronald Cohn Jesse Russell som släpptes 1 januari 2012 (amazon, 2012).

3.1.13 Are there any other interesting things regarding this type of file that will be interesting for a reader of the report?

Den 15 december 2006 innehöll den sammanlagda samlingen 615 ordlistor, på en total storlek av 936 189 613 bytes och 24 804 355 artiklar (Wikipedia, u.å.).

3.2 Lösningen (Omvandling från XML till HTML)

Det verktyg som valdes för att parsa filen var XSLT. Detta dels på grund av att det var den sista API:n som lärdes ut, och dels för att det verkade som ett lämpligt verktyg att använda med tanke på XML-filen och vad som skulle plockas ut ur den.

Processen startade med att skapa en template som utgick ifrån rotnoden "xdxf" och som kallade på alla de andra templatesen. Under kommandot "apply-templates" skrevs en tabell in som resulterade i att en tabell med förkortningarnas betydelse skrevs ut nedanför den stora tabellen.

Därefter skapades en template var för taggarna "full_name" och "description" och skrev ut dessa på var sin rad i tabellen som rubrik och underrubrik. I templatens för "description" hårdkodades dessutom tabellens kolumnrubriker. Sådär långt var det inga större problem.

Första problemet som dök upp var när man skulle skriva ut innehållet i taggen "ar". Eftersom denna tagg fanns i en massa varianter så fick man börja med att skilja på dessa. För den första ar-taggen skulle ingenting hända. Detta eftersom den inte innehöll vare sig ett enskilt ord eller en förklaring till ett ord. Det var ju trots allt ord från en ordlista som skulle skrivas ut. Lösningen blev att kontrollera ar-taggen med en count(). Om ar-taggen innehöll en k-tag som i sin tur innehöll exakt en opt-tag så betydde det att det var den första ar-taggen i serien. Detta sparades som en template och när templatens kallades så hände just ingenting.

Ytterligare en specialvariant på ar-taggen fanns. Även detta kontrollerades med en count() och sparades som en template. Om ar-taggen hade ett attribut som hette f så betydde det att den var förhållandevis välstrukturerad och hade lite annorlunda taggar under sig än ordinarie ar-taggen. Alla attribut utom def (definitionen på ordet) skrev ut i tur och ordning. Def-taggen däremot krävde en foreach som itererade över den och för varje rad skulle en taggens ordning i "kön" skrivas ut och efter varje definition skulle ett radbyte komma.

Lösningen på detta blev att använda sig utav en position() som räknade taggens turordning och skrev ut detta.

Den ordinarie ar-taggen lades i ytterligare en template. Eftersom varje tagg som skulle fylla raden inte alltid fanns så fick lösningen bli att en choose skapades för vardera en utav dem. Om taggen fanns skrevs innehållet ut, i annat fall skrevs det ut att informationen saknades.

Det sista och mest bekymmersamma problemet som dök upp var att PCDATAN i taggen ar i vissa fall innehöll skräptecken som inte kändes nödvändiga att skriva ut i tabellen.

Lösningen blev att skapa en foreach enligt mallen på dpawson.co.uk (u.å.) som kallade på en specifik template för varje textsträng. Denna template döptes passande nog till "clean" och det den gjorde var att spara den aktuella textsträngen i en variabel. Därefter kontrollerades denna variabel med en choose och om den innehöll vissa specifika tecken så ersattes eller raderades dessa och enbart texten före och efter skrevs ut.

3.3 Extra finesser

När den obligatoriska delen av uppgiften var avklarad så lades några ytterligare finesser till.

3.3.1 Sortering

En xsl:sort lades till i for-eachen som skrev ut de olika förkortningarna och deras betydelse i den mindre tabellen. Med hjälp av sort'en sorterades sedan förkortningarna i bokstavsordning se rad 15 i Bilaga N.

3.3.2 DTD

En DTD skapades med en hel del motgångar (se Bilaga O), men tillslut visade resultatet "Validation successful!"

3.3.3 Byte av layout

En idé om att användaren själv skulle kunna byta layout på hemsidan dök upp och det löstes med att ta hjälp av grow collectives (2006) hemsida.

Lösningen gick ut på att lägga in en while-loop i PHP-dokumentet (se rad 37-45 i Bilaga M) som listade innehållet i en extern PHP-fil (se Bilaga P) som länkar. För att kunna byta CSS-layout med en enkel länk så sköttes detta via ytterligare en extern PHP-fil (se Bilaga Q) som innehöll en cookie som såg till att det val som användaren senast hade gjort sparades inför framtida besök.

Till sist skapades ett antal CSS-filer (se Bilaga R) med enda skillnaden att några färgkoder byttes ut.

3.4 Det lilla som borde gjorts annorlunda

Det enda som kanske borde ha gjorts annorlunda är egentligen att ar-taggen på något sätt borde ha placerats annorlunda och istället sedan kallat på sina respektive undertaggar. Detta för att man då kunde ha valt att sortera orden i bokstavsordning vilket känns lämpligt när det handlar om en ordlista.

4. Reflektioner

Att få prova på att programmera i olika API'er har varit riktigt roligt, men samtidigt väldigt utmanande. Man har grubblat en hel del på hur man ska lösa de olika uppgifterna, och så plötsligt så har man sett logiken och sedan är det hur enkelt som helst och man tycker: Hur kunde det ta så lång tid att lista ut detta?

XML-filen som delades ut till mig var dock riktigt lurig att knäcka. I princip varenda tagg hade olika och blandat innehåll och dessutom var undertaggar helt plötsligt i en annorlunda ordning än tidigare. Att knäcka DTD'n tog en hel del tid men till slut så gick det ändå. Jag är i alla fall väldigt nöjd med mitt slutgiltiga resultat (se Bilaga S) och hoppas nu bara att det står upp till mina förväntningar när det gäller betyget i denna kurs.

Referenser

amazon (2012) XDXF [Paperback]. Tillgänglig på Internet: http://www.amazon.com/XDXF-Ronald-Cohn-Jesse-Russell/dp/B007OKITV6/ref=sr_1_1?ie=UTF8&qid=1337856543&sr=8-1 [Hämtad 12.05.24].

dpawson.co.uk (u.å.) Replace. Tillgänglig på Internet: www.dpawson.co.uk/xsl/sect2/replace.html [Hämtad 12.05.19].

Google Groups (u.å.) XDXF - The Extensible (XML) Dictionary Exchange Format. Tillgänglig på Internet: <http://groups.google.com/group/xdxf-format> [Hämtad 12.05.24].

grow collectives (2006) - CSS Stylesheet Switcher using PHP (Javascript free). Tillgänglig på Internet: http://2008.gr0w.com/articles/code/css_stylesheet_switcher_using_php_javascript_free/ [Hämtad 12.05.25].

Revdanica (u.å.) XDXF XML Dictionary Exchange Format. Tillgänglig på Internet: <http://xdxf.revdanica.com/> [Hämtad 12.05.24].

sourceforge (u.å.) XDXF - XML Dictionary Exchange Format. Tillgänglig på Internet: <http://sourceforge.net/projects/xdxf/> [Hämtad 12.05.23].

The ARTFL Project (u.å.) Webster's Revised Unabridged Dictionary (1913 + 1828). Tillgänglig på Internet: <http://machaut.uchicago.edu/websters> [Hämtad 12.05.23].

Wikipedia (u.å.) XDXF. Tillgänglig på Internet: <http://en.wikipedia.org/wiki/XDXF> [Hämtad 12.05.23]

Bilaga A. Uppgift 1A (Simpel layout med PHP)

```

1  <html><body><pre>
2  <?php
3      $pixelsizes=Array("10","20","30","40");
4      $fontcolor=Array("red","blue","yellow","green","pink");
5      $fontweight=Array("thin","normal","bold");
6
7      if(isset($_POST['textbox'])){
8          $textboxvalue=$_POST['textbox'];
9      }else{
10         $textboxvalue="";
11     }
12
13     echo "<form method='post' action='a.php'>";
14
15     // The code iterates over the array $pixelsizes and creates
16     corresponding option tags
17     echo "<select name='pixelsize'>";
18     foreach($pixelsizes as $pixelsize){
19         echo "<option>".$pixelsize;
20     }
21     echo "</select>";
22
23     // The code iterates over the array $fontcolor and creates
24     corresponding option tags
25     echo "<select name='color'>";
26     foreach($fontcolor as $color){
27         echo "<option>".$color;
28     }
29     echo "</select>";
30
31     // The code iterates over the array $fontweight and creates
32     corresponding option tags
33     echo "<select name='weight'>";
34     foreach($fontweight as $weight){
35         echo "<option>".$weight;
36     }
37     echo "</select>";
38
39     echo "<input type='text' name='textbox' value='$textboxvalue'>";
40
41     echo "<input type='submit' value='Ok!'>";
42
43     echo "</form>";
44
45     // This is where the new code goes!?
46
47     print_r($_POST);
48     echo "<br><br>";
49
50     if(isset($_POST['pixelsize'])){
51         $pixelsize=$_POST['pixelsize'];
52     }else{
53         $pixelsize="5";
54     }
55
56     if(isset($_POST['color'])){
57         $color=$_POST['color'];
58     }else{

```

Bilaga A

```
59         $color="red";
60     }
61
62     if(isset($_POST['weight'])) {
63         $weight=$_POST['weight'];
64     }else{
65         $weight="normal";
66     }
67
68     echo "<br><br>";
69
70     echo "<div style='font-size:".$pixelsize."px; color:".$color."; font-
71 weight:".$weight."'>".$textboxvalue."</div>";
72 ?>
73 </pre></body></html>
```

Bilaga B. Uppgift 1B (Nästlade arrayer i tabell)

```

1  <html><body><pre>
2  <?php
3
4  $trucks=Array( //array=trucks
5      Array( "KrAZ", //array=truck
6          "Kremenchuk",
7          "Ukraine",
8          Array( Array("KrAZ-65055","6x6","330Hp"), //array=vehicle
9              Array("KrAZ-6130C4","6x6","330Hp"),
10             Array("KrAZ-5133H2","4x2","330Hp"),
11             Array("KrAZ-7140H6","8x6","400Hp")
12             )
13         ),
14     Array( "EBIAM",
15         "Thessaloniki",
16         "Greece",
17         Array( Array("EBIAM MVM","4x4","86Hp")
18             )
19         ),
20     Array( "KaMAZ",
21         "Naberezhnye Chelny",
22         "Tatarstan",
23         Array( Array("KAMAZ 54115","6x4","240Hp"),
24             Array("KAMAZ 6560","8x8","400Hp"),
25             Array("KAMAZ 5460","8x8","340Hp")
26             )
27         ),
28     Array( "LIAZ",
29         "Rynovice",
30         "Czechoslovakia",
31         Array( Array("LIAZ 706 RT","2x4","160Hp")
32             )
33         ),
34     Array( "IRUM",
35         "Brasov",
36         "Romania",
37         Array( Array("TAF 690","2x4","90Hp")
38             )
39         ),
40     Array( "MAZ",
41         "Minsk",
42         "Belarus",
43         Array( Array("MAZ 535","8x8","375Hp"),
44             Array("MAZ 7310","8x8","525Hp"),
45             Array("MAZ 7907","4x12","1250Hp"),
46             Array("MAZ 6317","6x6","425Hp"),
47             Array("MAZ 6430","6x6","360Hp"),
48             Array("MAZ 5551","4x2","160Hp")
49             )
50         ),
51     ),
52     Array( "BelAz",
53         "Zohodino",
54         "Belarus",
55         Array( Array("Belaz 75600","4x4","3400Hp")
56             )
57         ),
58     ),

```

Bilaga B

```

59     Array( "Oshkosh",
60           "Oshkosh",
61           "USA",
62           Array( Array("Oshkosh P-15","8x8","840Hp"),
63                 Array("Oshkosh MK-36","6x6","425Hp")
64             )
65         ),
66     Array( "Tatra",
67           "Koprivnice",
68           "Czechoslovakia",
69           Array( Array("Tatra T 813","4x4","266Hp"),
70                 Array("Tatra T 815","10x10","436Hp"),
71             )
72         )
73 );
74
75     $countries=Array(); //Loopen under tar fram varje land som sedan
76 kommer att ligga i listan och sparar detta i denna array.
77
78     foreach($trucks as $truck){
79         // Is the key $truck[2] i.e. the country name set in the array
80 $countries or not?
81         // If not this means that this country does not exist in the array
82 and therefore should be added to the array
83         // This creates a list of countries without duplicates
84         if(!isset($countries[$truck[2]])){
85             $countries[$truck[2]]=$truck[2];
86         }
87     }
88
89     if(isset($_POST['country'])){
90         $showcountry=$_POST['country'];
91     }else{
92         $showcountry="Ukraine";
93     }
94
95     // For each country create an option tag for the select named country
96     echo "<form method='post' action='b.php'><select name='country'
97 onchange='this.form.submit()'>";
98     foreach($countries as $country){
99         if($showcountry==$country){
100             echo "<option selected='selected'>".$country."</option>";
101         }
102         else{
103             echo "<option>".$country."</option>";
104         }
105     }
106     echo "</select></form>";
107
108     // This is where the new code goes!
109     echo "<br /><table border='5'>";
110     echo "<tr><th colspan='5' style='background-color:#ffd35b'><div
111 style='text-transform:uppercase'>".$showcountry."</div></th></tr>";
112     echo "<tr style='background-
113 color:#ffeec2'><th>City</th><th>Manufacturer</th><th>Model</th><th>Wheels</
114 th><th>Engine</th></tr>";
115     foreach($trucks as $truck){
116         if($truck[2]==$showcountry){
117             $i=1;
118             foreach($truck[3] as $vehicle){
119                 if($i % 2){

```

Bilaga B

```
120         echo "<tr style='background-color:#c2dfff'>";
121     }
122     else{
123         echo "<tr style='background-color:#e5f1ff'>";
124     }
125     echo "<td>".$struck[1]."</td>";
126     echo "<td>".$struck[0]."</td>";
127     echo "<td>".$vehicle[0]."</td>";
128     echo "<td text-align:center'>".$vehicle[1]."</td>";
129     echo "<td text-align:center'>".$vehicle[2]."</td>";
130     echo "</tr>";
131     $i++;
132 }
133 }
134 }
135 echo "</table><br />";
136
137 //print_r($countries);
138
139 ?>
140 </pre></body></html>
```


Bilaga C. XML-filen som parsas i alla mindre uppgifter

```
1 <newspapers>
2 <newspaper name="Times" subscribers="100.000" type="Morning Edition">
3   <article id="1232" time="2001-01-01">
4     <heading>
5       This and that
6     </heading>
7     <text>
8       This and that happened during the day and all were amazed.
9     </text>
10  </article>
11  <article id="1595">
12    <heading>
13      Another Great Happening
14    </heading>
15    <text>
16      Another Happening!
17    </text>
18    <comment description="Review"/>
19  </article>
20  <article id="1692" time="2001-01-02">
21    <heading>
22      Mother Goose Seen Again
23    </heading>
24    <text>
25      Very interesting information about childrens book character.
26    </text>
27  </article>
28 </newspaper>
29 <newspaper name="News of The World" subscribers="800.000" type="Evening
30 Edition">
31   <article id="1832" time="2001-01-01">
32     <heading>
33       Horrible Happenings
34     </heading>
35     <text>
36       This and that happened during the evening and all were apalled.
37     </text>
38     <comment description="Review"/>
39   </article>
40   <article id="1895">
41     <heading>
42       Another Great Happening
43     </heading>
44     <text>
45       Another Happening!
46     </text>
47     <comment description="News"/>
48   </article>
49   <article id="1892" time="2001-01-02">
50     <heading>
51       Mother Goose Sadly Not Seen Again
52     </heading>
53     <text>
54       Childrens book character snuffed out by maffia.
55     </text>
56   </article>
57 </newspaper>
58 <newspaper name="The SUN" subscribers="300.000" type="Evening Edition">
```

Bilaga C

```
59 <article id="1882" time="2011-01-01">
60   <heading>
61     All is Well
62   </heading>
63   <text>
64     Events unfolded and things happened.
65   </text>
66   <comment description="News"/>
67 </article>
68 <article id="1883">
69   <heading>
70     Even More Things
71   </heading>
72   <text>
73     Unjust musements.
74   </text>
75   <comment description="Review"/>
76 </article>
77 <article id="1884" time="2001-01-02">
78   <heading>
79     Mother Goose Lost In Battle
80   </heading>
81   <text>
82     Childrens book character still missing.
83   </text>
84 </article>
85 </newspaper>
86 </newspapers>
```

Bilaga D. Uppgift 2 (SAX)

```

1  <html>
2    <body>
3      <pre>
4      <?php
5
6          function startElement($parser, $entityname, $attributes) {
7              global $indent;
8              $line = xml_get_current_line_number($parser);
9              echo $line." ";
10             for($i=0;$i<$indent;$i++) echo "    ";
11             $indent++;
12             echo "<";
13             foreach ($attributes as $attname => $attvalue) {
14                 if($attname=='NAME'){
15                     echo "<td rowspan='3' width='90'>".$attvalue."</td>";
16                 }
17                 if($attname=='SUBSCRIBERS'){
18                     echo "<td rowspan='3'>".$attvalue."</td>";
19                 }
20                 if($attname=='TYPE'){
21                     echo "<td rowspan='3'>".$attvalue."</td>";
22                 }
23                 if($attname=='ID'){
24                     echo "<td>".$attvalue."</td>";
25                     if(isset($attributes['TIME'])){
26                         echo "<td>".$attributes['TIME']."</td>";
27                     }
28                     else{
29                         echo "<td style='color:grey'><div align=center>no
30 date</div></td>";
31                     }
32                 }
33                 if($entityname=='COMMENT'){
34                     echo "<td>".$attributes['DESCRIPTION']."</td>";
35                 }
36                 echo " ".$attname."='".$attvalue."'";
37             }
38             echo "><br>";
39             print_r($attributes);
40         }
41
42         function endElement($parser, $entityname) {
43             global $indent;
44             $line = xml_get_current_line_number($parser);
45             echo $line." ";
46             $indent--;
47             for($i=0;$i<$indent;$i++) echo "    ";
48             echo "</".$entityname."><br>";
49             if($entityname=='ARTICLE'){
50                 echo "</tr><tr>";
51             }
52         }
53
54         function charData($parser, $chardata) {
55             global $indent;
56             $line = xml_get_current_line_number($parser);
57             $chardata=trim($chardata);
58             if($chardata=="") return;

```

Bilaga D

```
59         echo $line." ";
60         for($i=0;$i<$indent;$i++) echo "    ";
61         echo $chardata."<br>";
62
63         echo "<td>".$chardata."</td>";
64     }
65
66     $parser = xml_parser_create();
67     $indent = 0;
68     xml_set_element_handler($parser, "startElement", "endElement");
69     xml_set_character_data_handler($parser, "charData");
70
71     $file = 'example2.xml';
72     $data = file_get_contents($file);
73     echo "<table border='1' align=center><tr><th colspan='8'
74 style='background-color:#3f63ff; color:#ffffff'>NEWSPAPERS</th></tr>";
75     echo "<tr style='background-
76 color:#9baeff'><th>name</th><th>subscribers</th><th>edition</th><th>article
77 ID</th><th>date</th><th>headline</th><th>content</th><th>type</th></tr>";
78     echo "<tr>";
79     if(!xml_parse($parser, $data, true)){
80         printf("<P> Error %s at line %d</P>",
81 xml_error_string(xml_get_error_code($parser)),xml_get_current_line_number($
82 parser));
83     }else{
84         print "<br>Parsing Complete!</br>";
85     }
86
87
88     echo "</tr></table>";
89     xml_parser_free($parser);
90     ?>
91 </pre>
92 </body>
93 </html>
```

Bilaga E. Uppgift 3 (DOM)

```

1  <html>
2  <body>
3  <pre>
4  <?php
5      $file = 'example2.xml';
6
7      $dom = new DomDocument;
8      $dom->preserveWhiteSpace = FALSE;
9      $dom->load($file);
10
11      echo "<br>";
12      echo "<table border='1'>";
13      echo "<tr>";
14      echo "    <th colspan='8' style='background-color:#3f63ff;
15color:#ffffff'><div style='text-
16transform:uppercase'>Newspapers</div></th>";
17      echo "</tr>";
18      echo "<tr style='background-color:#9baeff'>";
19      echo "    <th width='90'>name</th><th>subscribers</th><th
20width='130'>edition</th><th>articleID</th><th>date</th><th
21width='267'>headline</th><th>content</th><th>type</th>";
22      echo "</tr>";
23
24      $papers = $dom->getElementsByTagName('newspaper');          //Denna
25      rad sparar varje tagg som heter "newspaper" och dess innehåll i arrayen
26      $papers.
27      foreach ($papers as $paper){
28          echo $paper->tagName."<br>";          //Denna rad skriver ut
29      taggnamnet på varje nod i arrayen $papers i den övre texten.
30          echo "<tr>";          //Denna rad skapar en ny tabellrad
31      för varje tidning.
32
33          $paperAttributes = $paper->attributes;
34          foreach ($paperAttributes as $index=>$attr) {          //Denna
35      rad gör det möjligt att skriva ut attributen i newspaper-taggen.
36          echo "<td rowspan='3'>".$attr->value."</td>";
37      }
38
39          foreach ($paper->childNodes as $child){          //Denna
40      rad itererar över varje newspaper-tag och sparar barntaggen "article" och
41      dess innehåll i arrayen $child.
42          echo " ".$child->tagName;          //Denna rad skriver ut
43      taggnamnet på varje nod i arrayen $child i den övre texten.
44
45          $attributes = $child->attributes;          //På denna rad sparar
46      article's attribut och dess innehåll i arrayen $attributes.
47          foreach ($attributes as $index=>$attr) {          //Denna rad gör det
48      möjligt att skriva ut attributen i article-taggen.
49          echo " ".$attr->name."=".$attr->value;          //Denna
50      rad skriver ut attributnamnet och attributvärdet på varje nod i arrayen
51      $attributes i den övre texten.
52          if($attr->name=="id"){
53              echo "<td>".$attr->value."</td>";
54          }
55          if($attr->name!="id"){
56              echo "<td>".$attr->value."</td>";
57          }
58          if($attributes->item(1)== "") {

```

Bilaga E

```
59         echo "<td style='color:grey'><div align=center>no
60 date</div></td>";
61     }
62 }
63
64     foreach ($schild->childNodes as $children){           //Denna
65 rad sparar barntaggarna till article (heading,text,comment) i arrayen
66 $children.
67
68         if($children->tagName=="heading"){
69             echo "<td>".$children->nodeValue."</td>";
70         }
71         if($children->tagName=="text"){
72             echo "<td>".$children->nodeValue."</td>";
73         }
74         if($schild->lastChild->nodeName=="comment"){
75             echo "<td>&nbsp;</td>";
76         }
77     }
78     echo "</tr>";
79     echo "    <br>".$schild->nodeValue."<br>";           //Denna
80 rad skriver ut taggvärdet på varje nod i arrayen $schild i den övre texten.
81 }
82 }
83 echo "<tr><td colspan='8'>&nbsp;<td colspan='8'>&nbsp;</td></tr>";
84 if ($schild->firstChild->nodeName=="heading"){
85     echo "hej";
86 }
87
88 echo "</td></tr>";
89 echo "</table>";
90 ?>
91 </pre>
92 </body>
93 </html>
```

Bilaga F. Uppgift 4 (PHP-filen som användes)

```
1 <html>
2 <body>
3 <pre>
4 <?php
5
6 function HandleXmlError($errno, $errstr, $errfile, $errline)
7 {
8     echo $errno." ".$errstr."<br>";
9 }
10
11 $file = 'example2.xml';
12
13 $dom = new DomDocument;
14 $dom->preserveWhiteSpace = FALSE;
15
16 set_error_handler('HandleXmlError');
17 $loadstate=$dom->load($file);
18 if($loadstate){
19     $validatestate=$dom->validate();
20     if($validatestate){
21         echo "Validation successful!<br>";
22     }else{
23         echo "Validation test failed!<br>";
24     }
25 }else{
26     echo "Well-formedness test Failed!<br>";
27 }
28 restore_error_handler();
29
30 ?>
31 </pre>
32 </body>
33 </html>
```

Bilaga G. Uppgift 4 (DTD-filen som användes)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!ELEMENT newspapers (newspaper+)>
4
5  <!ELEMENT newspaper (article+)>
6  <!ATTLIST newspaper name CDATA #REQUIRED>
7  <!ATTLIST newspaper subscribers CDATA #REQUIRED>
8  <!ATTLIST newspaper type CDATA #REQUIRED>
9
10 <!ELEMENT article (heading,text,comment?)>
11 <!ATTLIST article id CDATA #REQUIRED>
12 <!ATTLIST article time CDATA #IMPLIED>
13
14 <!ELEMENT heading (#PCDATA)>
15
16 <!ELEMENT text (#PCDATA)>
17
18 <!ELEMENT comment EMPTY>
19 <!ATTLIST comment description CDATA #REQUIRED>
```


Bilaga H. Uppgift 5 (XPath)

```

1  <html><body><pre>
2  <?PHP
3      $dom = new DomDocument;
4      $dom->preserveWhiteSpace = FALSE;
5
6      $file = 'example2.xml';
7      $loadstate=$dom->load($file);
8
9      $xp = new DOMXPath($dom);
10
11
12      //Detta stycke skriver ut hela XML-dokumentet med XPath.
13      $xpathString="//*";
14      echo "<b>Example1: ".$xpathString."</b><br>";
15      $nodes = $xp->query($xpathString);
16      foreach ($nodes as $node) {
17          echo "    Name:".$node->nodeName." Value:".$node->nodeValue;//."
18      NodeText: ".$node->textContent."<br>";
19      }
20
21      //TEST!
22      $xpathString="//article";
23      echo "<b>Example1: ".$xpathString."</b><br>";
24      $headings = $xp->query($xpathString);
25      foreach ($headings as $heading) {
26          echo "    Name:".$heading->nodeName." Value:".$heading->nodeValue;//."
27      NodeText: ".$heading->textContent."<br>";
28      }
29
30      //Detta stycke skapar början på min tabell.
31      echo "<table border='1'><tr><th colspan='8' style='background-
32      color:#3f63ff; color:#ffffff'><div style='text-
33      transform:uppercase'>Newspapers</div></th></tr>";
34      echo "<tr style='background-
35      color:#9baeff'><th>name</th><th>subscribers</th><th>edition</th><th>article
36      ID</th><th>date</th><th>headline</th><th>content</th><th>type</th></tr>";
37
38
39      //Början på det som låg i koden från start.
40      $xpathString="//newspaper"; //Denna rad sparar
41      strängen "//newspaper" i variabeln $xpathString.
42      $papers = $xp->query($xpathString); //Denna
43      rad skickar med adressen som sparats i $xpathString till en funktion som
44      sen sparar alla taggar som ligger under newspaper och deras innehåll i
45      arrayen $papers.
46      foreach ($papers as $paper) { //Denna rad itererar
47      över arrayen $papers och för varje tidning händer följande....
48
49          echo "<tr>"; //En ny tabellrad startas för
50      varje tidning.
51          $paperAttributes = $paper->attributes;
52          //Attributen sparas i arrayen $paperAttributes.
53          foreach ($paperAttributes as $index=>$attr) {
54              echo "<td rowspan='3'>".$attr->value."</td>"; //Varje
55      attribut skrivs ut i en cell.
56          }
57
58          $innerdom = new DomDocument; //

```

Bilaga H

```
59     $otherxml=$dom->saveXML($paper); //Detta stycke skapar
60 en inre XPATH-väg som kommer användas nedan.
61     $innerdom->loadXML($otherxml); //
62     $innerxp = new DOMXPath($innerdom); //
63
64     $xpathString="//article"; //Denna rad sparar strängen
65 "//article" i variabeln $xpathString.
66     $articles = $innerxp->query($xpathString);
67     //Denna rad skickar med adressen som sparats i $xpathString till en
68 funktion som sen sparar alla taggar som ligger under newspaper och deras
69 innehåll i arrayen $articles.
70
71     foreach ($articles as $article) { //Denna rad itererar
72 över arrayen $articles och för varje artikel händer följande....
73         $articleAttributes = $article->attributes;
74         //Attributen sparas i arrayen $paperAttributes.
75         foreach ($articleAttributes as $index=>$attr) {
76             echo "<td>".$attr->value."</td>"; //Varje attribut
77 skrivs ut i en cell.
78             if($articleAttributes->item(1)==""){
79                 echo "<td style='color:grey'><div align=center>no
80 date</div></td>"; //Om datumfältet är tomt skrivs detta ut.
81             }
82         }
83
84         // $articleNodes = $article->nodeValue;
85         // echo "<td>hej ".$articleNodes."</td>";
86
87         $innerstdom = new DomDocument; //
88         $thirdxml=$dom->saveXML($paper); //Detta
89 stycke skapar en inre XPATH-väg som kommer användas nedan.
90         $innerstdom->loadXML($thirdxml); //
91         $innerstxp = new DOMXPath($innerstdom);
92         //
93
94         $xpathString="//heading";
95         $headings = $innerstxp->query($xpathString);
96         // foreach($headings as $heading){
97             // $headingNodes = $heading->nodeValue;
98             // echo "<td>".$headingNodes."</td>";
99         // }
100         $heading="&nbsp;";
101         $text="&nbsp;";
102         $comment="&nbsp;";
103
104         //$headingNodes = $headings->nodeValue;
105         echo "<td>".$heading."</td>";
106         echo "<td>".$text."</td>";
107         echo "<td>".$comment."</td></tr>";
108     }
109
110     //echo "</tr>";
111
112 }
113
114 //Slut på tabell!
115 echo "</tr></table>";
116 ?>
117 </pre></body></html>
```

Bilaga I. PHP-filen som används i uppgift 6 och 7

```

1  <html>
2  <body>
3  <?php
4
5      $xslDoc = new DOMDocument();
6      $xslDoc->load("XSLT Example2.xsl");
7
8      $xmlDoc = new DOMDocument();
9      $xmlDoc->load("example2.xml");
10
11     $proc = new XSLTProcessor();          //En xslt-processor sparas i
12     variabeln $proc.
13     $proc->importStylesheet($xslDoc);      //Jag kör metoden importStylesheet
14     på xslt-processorn $proc och skickar med XSL-dokumentet som sparats i
15     $xslDoc.
16
17     //Detta stycke skapar början på min tabell.
18     echo "<table border='1'><tr><th colspan='8' style='background-
19     color:#3f63ff; color:#ffffff'><div style='text-
20     transform:uppercase'>Newspapers</div></th></tr>";
21     echo "<tr style='background-
22     color:#9baeff'><th>name</th><th>subscribers</th><th>edition</th><th>article
23     ID</th><th>date</th><th>headline</th><th>content</th><th>type</th></tr>";
24
25
26     echo $proc->transformToXML($xmlDoc);    //Här kopplas stylesheet och xml
27     ihop via metoden transformToXML och skrivs sedan ut.
28
29     //Slut på tabell!
30     echo "</tr></table>";
31     ?>
32 </body>
33 </html>

```

Bilaga J. Uppgift 6 (XSLT)

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3  version="1.0">
4      <xsl:template match="/"> <!-- Talar om att jag vill utgå från rotnoden --
5  >
6          <xsl:for-each select="./newspapers/newspaper"> <!-- Jag använder en
7  foreach för varje tidning så att all information buntas ihop -->
8          <xsl:for-each select="./article"> <!-- Eftersom jag inte vill ha
9  brutna taggar måste jag gå ner ytterligare ett steg i trädet-->
10         <xsl:choose>
11             <xsl:when test="position()=1"> <!-- Denna rad ser till att
12  tidningsnamnet skrivs ut med rowspan 3 endast på första artikeln -->
13                 <tr>
14                     <td rowspan="3"><xsl:value-of select="../@name"/></td>
15                     <!-- All tidningsinfo listas på dessa 3 rader -->
16                     <td rowspan="3"><xsl:value-of
17  select="../@subscribers"/></td>
18                     <td rowspan="3"><xsl:value-of select="../@type"/></td>
19                     <td><xsl:value-of select="../@id"/></td> <!-- Här listas
20  artikelns id-nr. -->
21                     <xsl:choose>
22                         <xsl:when test="count(../*)!=2"> <!-- Om article inte
23  har två attribut betyder det att datumet saknas och då skrivs no date -->
24                             <td style="color:grey"><div align="center">no
25  date</div></td>
26                             </xsl:when>
27                             <xsl:otherwise>
28                                 <td><xsl:value-of select="../@time"/></td> <!-- I annat
29  fall, datumet -->
30                                 </xsl:otherwise>
31                             </xsl:choose>
32                             <td><xsl:value-of select="..@heading"/></td> <!-- Artikelns
33  heading och text listas -->
34                             <td><xsl:value-of select="..@text"/></td>
35                             <xsl:choose>
36                                 <xsl:when test="count(..*)=3"> <!-- Om article har 3
37  undertaggar betyder det att comment finns -->
38                                     <td><xsl:value-of
39  select="..@comment/@description"/></td> <!-- Isf skrivs beskrivningen ut -->
40                                     </xsl:when>
41                                     <xsl:otherwise>
42                                         <td style="color:grey"><div align="center">no
43  info</div></td> <!-- I annat fall no info -->
44                                         </xsl:otherwise>
45                                     </xsl:choose>
46                                 </td>
47                             </tr>
48                         </xsl:when>
49                         <xsl:otherwise> <!-- Om artikeln inte är först i ordning skrivs
50  inte tidningsnamnet ut, men i övrigt är resten likadant -->
51                             <tr>
52                                 <td><xsl:value-of select="../@id"/></td>
53                                 <xsl:choose>
54                                     <xsl:when test="count(../*)!=2">
55                                         <td style="color:grey"><div align="center">no
56  date</div></td>
57                                         </xsl:when>
58                                         <xsl:otherwise>

```

```

59         </xsl:otherwise>
60     </xsl:choose>
61     <td><xsl:value-of select="./heading"/></td>
62     <td><xsl:value-of select="./text"/></td>
63     <xsl:choose>
64         <xsl:when test="count(./*)=3">
65             <td><xsl:value-of
66 select="./comment/@description"/></td>
67             </xsl:when>
68             <xsl:otherwise>
69                 <td style="color:grey"><div align="center">no
70 info</div></td>
71             </xsl:otherwise>
72         </xsl:choose>
73     </tr>
74 </xsl:otherwise>
75 </xsl:choose>
76 </xsl:for-each>
77 </xsl:for-each>
78 </xsl:template>
79 </xsl:stylesheet>

```

Bilaga K. Uppgift 7 (XSLT apply templates)

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3  version="1.0">
4
5      <xsl:template match="newspapers">
6          <xsl:for-each select="newspaper">
7              <xsl:sort select="@subscribers" order="descending"/>
8              <tr>
9                  <td rowspan="3"><xsl:value-of select="@name"/></td> <!-- rowspan=3
10 skulle kunna bytas ut mot en count för antal artiklar. <xsl:value-of
11 select="count(./*)" /> -->
12                  <td rowspan="3"><xsl:value-of select="@subscribers"/></td>
13                  <td rowspan="3"><xsl:value-of select="@type"/></td>
14                  <xsl:apply-templates/>
15              </tr>
16          </xsl:for-each>
17      </xsl:template>
18
19      <xsl:template match="article[position()=1]">
20          <td><xsl:value-of select="./@id"/></td>
21          <xsl:choose>
22              <xsl:when test="count(./@*)!=2">
23                  <td style="color:grey"><div align="center">no date</div></td>
24              </xsl:when>
25              <xsl:otherwise>
26                  <td><xsl:value-of select="./@time"/></td>
27              </xsl:otherwise>
28          </xsl:choose>
29          <xsl:apply-templates/>
30          <xsl:choose>
31              <xsl:when test="count(./@*)=3">
32                  <td><xsl:value-of select="./comment/@description"/></td>
33              </xsl:when>
34              <xsl:otherwise>
35                  <td style="color:grey"><div align="center">no info</div></td>
36              </xsl:otherwise>
37          </xsl:choose>
38      </xsl:template>
39
40      <xsl:template match="article[position()>1]">
41          <tr>
42              <td><xsl:value-of select="./@id"/></td>
43              <xsl:choose>
44                  <xsl:when test="count(./@*)!=2">
45                      <td style="color:grey"><div align="center">no date</div></td>
46                  </xsl:when>
47                  <xsl:otherwise>
48                      <td><xsl:value-of select="./@time"/></td>
49                  </xsl:otherwise>
50              </xsl:choose>
51              <xsl:apply-templates/>
52              <xsl:choose>
53                  <xsl:when test="count(./@*)=3">
54                      <td><xsl:value-of select="./comment/@description"/></td>
55                  </xsl:when>
56                  <xsl:otherwise>
57                      <td style="color:grey"><div align="center">no info</div></td>
58                  </xsl:otherwise>

```

Bilaga K

```
59         </xsl:choose>
60     </tr>
61 </xsl:template>
62
63 <xsl:template match="heading">
64     <td><xsl:value-of select="."/></td>
65 </xsl:template>
66
67 <xsl:template match="text">
68     <td><xsl:value-of select="."/></td>
69 </xsl:template>
70
71 </xsl:stylesheet>
```

Bilaga L. XML-filen till huvuduppgiften

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE xdxfl SYSTEM "xdxfl.dtd">
3
4  <xdxfl lang_from="ENG" lang_to="ENG" format="visual">
5      <full_name>
6          Webster's Unabridged Dictionary
7      </full_name>
8      <description>
9          Webster's Unabridged Dictionary published 1913 by the...
10     </description>
11
12     <abbreviations>
13         <abr_def>
14             <k>
15                 n.
16             </k>
17             <v>
18                 noun
19             </v>
20         </abr_def>
21         <abr_def>
22             <k>
23                 v.
24             </k>
25             <v>
26                 verb
27             </v>
28         </abr_def>
29         <abr_def>
30             <k>
31                 Av.
32             </k>
33             <k>
34                 Ave.
35             </k>
36             <v>
37                 Avenue
38             </v>
39         </abr_def>
40     </abbreviations>
41
42     <ar>
43         <k>
44             <opt>
45                 The
46             </opt>
47             United States
48             <opt>
49                 of America
50             </opt>
51         </k>
52     </ar>
53
54     <ar>
55         <k>
56             record
57         </k>
58     <pos>

```



```

59         <abr>
60         n.
61     </abr>
62 </pos>
63 [
64 <tr>
65     re'kord
66 </tr>
67 ] Anything written down and preserved.
68 <pos>
69     <abr>
70     v.
71     </abr>
72 </pos>
73 [
74 <tr>
75     reko'rd
76 </tr>
77 ]
78 To write down for future use.
79 </ar>
80
81 <ar>
82     <k>
83     home
84     </k>
85     [
86     <tr>
87         ho:um
88     </tr>
89     ]
90     <pos>
91         <abr>
92         n.
93         </abr>
94     </pos>
95     <rref start="16384" size="512">
96         sounds_of_words.ogg
97     </rref>
98     1) One's own dwelling place; the house in which one livs.2) One's
99 native land; the place or country in which one dwells.
100     3) The abiding place of the affections.
101     <ex>
102         For without hearts there is no home.
103     </ex>
104     4)
105     <dtrn>
106         дом
107     </dtrn>
108     at home - make yourself at home -
109     <ex>
110         XDXF Home page:
111         <iref>
112             http://xdxf.sourceforge.net
113         </iref>
114     </ex>
115     See also:
116     <kref>
117         home-made
118     </kref>
119 </ar>

```

```

120
121 <ar f="l">
122   <k>
123     home
124   </k>
125   <tr>
126     houm
127   </tr>
128   <pos>
129     <abr>
130       n.
131     </abr>
132   </pos>
133   <rref start="16384" size="512">
134     sounds_of_words.ogg
135   </rref>
136   <def>
137     One's own dwelling place; the house in which one lives.
138   </def>
139   <def>
140     One's native land; the place or country in which one dwells.
141   </def>
142   <def>
143     The abiding place of the affections.
144   <ex>
145     For without hearts there is no home.
146   </ex>
147   </def>
148   <def>
149     <dtrn>
150
151     </dtrn>
152     at ome - make yourself at home -
153   </def>
154   <ex>
155     XDXF Home page:
156     <iref>
157       http://xdxf.sourceforge.net
158     </iref>
159   </ex>
160   See also:
161   <kref>
162     home-made
163   </kref>
164 </ar>
165
166 <ar>
167   <k>
168     indices
169   </k>
170   Plural form of word
171   <kref>
172     index
173   </kref>
174 </ar>
175
176 <ar>
177   <k>
178     disc
179   </k>
180   <k>

```

Bilaga L

181 disk
182 </k>
183 <pos>
184 <abr>
185 n.
186 </abr>
187 </pos>
188 A flat, circular plate; as, a disk of metal or paper.
189 </ar>
190
191 <ar>
192 <k>
193 CO
194 <nu>
195
196 </nu>
197 2
198 <nu>
199
200 </nu>
201 </k>
202 Carbon dioxide (CO₂) - a heavy odorless gas formed during respiration.
203 </ar>
204
205 </xdef>

Bilaga M. PHP-filen till huvuduppgiften

```

1  <html>
2  <head>
3      <!-- <link href="xdxf.css" rel="stylesheet" type="text/css" /> -->
4  </head>
5  <body>
6  <?php
7      include('stylearray.php'); //Denna rad inkluderar PHP-filen som styr
8      layouten på sidan.
9
10     //Rad 10-32 används till DTD-valideringen!
11     function HandleXmlError($errno, $errstr, $errfile, $errline)
12     {
13         echo $errno." ".$errstr."<br>";
14     }
15
16     $file = 'xdxf.xml';
17
18     $dom = new DomDocument;
19     $dom->preserveWhiteSpace = FALSE;
20
21     set_error_handler('HandleXmlError');
22     $loadstate=$dom->load($file);
23     if($loadstate){
24         $validatestate=$dom->validate();
25         if($validatestate){
26             echo "<div class='validate'>Validation of DTD successful!</div>";
27         }else{
28             echo "<div class='validate'>Validation of DTD-test
29 failed!</div><br/>";
30         }
31     }else{
32         echo "<div class='validate'>Well-formedness test Failed!</div><br/>";
33     }
34     restore_error_handler();
35
36     //Rad 35-40 används för att skapa layoutalternativstaggar som skall kunna
37 väljas mellan.
38     echo "<div id='colors'>";
39     echo "Choose your preferred color-scheme: ";
40     while(list($key, $val) = each($styleSheets)){
41         echo "<a
42 href='styleswitcher.php?SETSTYLE=".$key."'>".$val["name"]."</a> ";
43     }
44     echo "</div>";
45
46     $xslDoc = new DOMDocument();
47     $xslDoc->load("xdxf.xsl");
48
49     $xmlDoc = new DOMDocument();
50     $xmlDoc->load("xdxf.xml");
51
52     $proc = new XSLTProcessor(); //En xslt-processor sparas i variabeln
53 $proc.
54     $proc->importStylesheet($xslDoc); //Jag kör metoden importStylesheet på
55 xslt-processorn $proc och skickar med XSL-dokumentet som sparats i $xslDoc.
56
57

```

Bilaga M

```
58     echo "<table>";           //Detta stycke skapar början på min
59 tabell.
60
61     echo $proc->transformToXML($xmlDoc); //Här kopplas stylesheet och xml
62 ihop via metoden transformToXML och skrivs sedan ut.
63
64     echo "</table>";           //Slut på tabell!
65 ?>
66 </body>
67 </html>
```

Bilaga N. XSL-filen till huvuduppgiften

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3  version="1.0">
4
5      <xsl:template match="xdxf">
6          <xsl:apply-templates/>
7          <table width="300"> <!-- Denna tabell skriver ut förkortningarnas
8  betydelse nedanför den stora tabellen -->
9              <tr>
10                 <td class='about' colspan="6"><xsl:value-of select
11 ="name(./abbreviations)"/></td>
12             </tr>
13             <tr>
14                 <xsl:for-each select="./abbreviations/abr_def">
15                     <xsl:sort select="./k" order="ascending"/> <!-- Denna rad används
16  för att sortera förkortningarna i bokstavsordning -->
17                     <td><xsl:value-of select="./k"/></td><td><xsl:value-of
18  select="./v"/></td>
19                 </xsl:for-each>
20             </tr></table>
21         </xsl:template>
22
23         <xsl:template match="full_name"> <!-- Skriver ut rubriken på tabellen -->
24             <tr>
25                 <th class='heading' colspan="6">
26                     <xsl:value-of select="."/>
27                 </th>
28             </tr>
29         </xsl:template>
30
31         <xsl:template match="description"> <!-- Skriver ut underrubriken på
32  tabellen och hårdkodar sedan kolumnerna -->
33             <tr>
34                 <td class='about' colspan="6">
35                     <xsl:value-of select="."/>
36                 </td>
37             </tr>
38             <tr>
39                 <th>word</th>
40                 <th>phonetic</th>
41                 <th>type</th>
42                 <th width="300">definition</th>
43                 <th>example</th>
44                 <th>soundfile</th>
45             </tr>
46         </xsl:template>
47
48         <xsl:template match="ar[count(./k/opt)>=1]"> <!-- Plockar fram den första
49  udda ar som innehåller taggen opt i k, och gör absolut ingenting med den --
50  >
51
52         </xsl:template>
53
54         <xsl:template match="ar[count(./@f)=1]"> <!-- Om ar har ett attribut som
55  heter f görs detta. Dett gäller alltså för [houm]. -->
56             <tr>
57                 <td class="word"><xsl:value-of select="./k"/></td> <!-- Skriver ut
58  ordet -->

```

Bilaga N

```

59         <td align="center">[<xsl:value-of select="./tr"/>]</td> <!-- Skriver
60 ut fonetiken -->
61         <td align="center"><xsl:value-of select="./pos/abr"/></td> <!--
62 Skriver ut förkortningen -->
63         <td>
64             <xsl:for-each select="./def"> <!-- Skriver ut alla definitioner
65 med sitt ordningsnr följt av ) först -->
66                 <xsl:value-of select="position()"/>) <xsl:value-of
67 select="."/><br/>
68             </xsl:for-each>
69         </td>
70         <td><xsl:value-of select="./def/ex"/></td> <!-- Skriver ut en
71 exempelmening -->
72         <td align="center"><xsl:value-of select="./rref"/><br/>filesize:
73 <xsl:value-of select="./rref/@size"/>kB</td> <!-- Skriver ut ljudfilen -->
74     </tr>
75 </xsl:template>
76
77 <xsl:template match="ar"> <!-- Plockar fram ordinarie ar -->
78     <tr>
79         <td class="word"><xsl:value-of select="./k"/></td> <!-- Skriver ut
80 ordet -->
81
82         <xsl:choose> <!-- Skriver ut fonetiken om den finns, i annat fall
83 talar den om att det inte finns -->
84             <xsl:when test="count(./tr)=0">
85                 <td align="center" class="no">no<br/>phonetic</td>
86             </xsl:when>
87             <xsl:otherwise>
88                 <td align="center">[<xsl:value-of select="./tr"/>]</td>
89             </xsl:otherwise>
90         </xsl:choose>
91
92         <xsl:choose> <!-- Skriver ut förkortningen om den finns, i annat
93 fall talar den om att den inte finns -->
94             <xsl:when test="count(./pos/abr)=0">
95                 <td align="center" class="no">no<br/>type</td>
96             </xsl:when>
97             <xsl:otherwise>
98                 <td align="center"><xsl:value-of select="./pos/abr"/></td>
99             </xsl:otherwise>
100         </xsl:choose>
101
102         <td> <!-- Kallar på en template som städar och sedan skriver ut alla
103 definitioner -->
104             <xsl:for-each select="text()">
105                 <xsl:call-template name="clean"/>
106             </xsl:for-each>
107         </td>
108
109         <xsl:choose> <!-- Skriver ut ett exempel om det finns, i annat fall
110 talar den om att det inte finns -->
111             <xsl:when test="count(./ex)=0">
112                 <td align="center" class="no">no example found</td>
113             </xsl:when>
114             <xsl:otherwise>
115                 <td align="center"><xsl:value-of select="./ex"/></td>
116             </xsl:otherwise>
117         </xsl:choose>
118

```

Bilaga N

```
119         <xsl:choose> <!-- Skriver ut namn på ljudfil om det finns, i annat
120 fall tom cell -->
121         <xsl:when test="count(./rref)=0">
122             <td align="center" class="no">no sound found</td>
123         </xsl:when>
124         <xsl:otherwise>
125             <td align="center"><xsl:value-of
126 select="./rref"/><br/>filesize: <xsl:value-of
127 select="./rref/@size"/>kB</td>
128         </xsl:otherwise>
129     </xsl:choose>
130 </tr>
131 </xsl:template>
132
133     <xsl:template name="clean"> <!-- Denna template städar
134 definitionstexterna på oönskade tecken -->
135     <xsl:param name="text" select="."/>
136     <xsl:choose>
137     <xsl:when test="contains($text, '&#x5b;')"> <!-- Denna when plockar
138 bort tecknet '[' -->
139         <xsl:value-of select="substring-before($text, '&#x5b;')"/>
140         <xsl:call-template name="clean">
141             <xsl:with-param name="text" select="substring-
142 after($text, '&#x5b;')"/>
143         </xsl:call-template>
144     </xsl:when>
145     <xsl:when test="contains($text, '&#x5d;')"> <!-- Denna when plockar
146 bort tecknet ']' -->
147         <xsl:value-of select="substring-before($text, '&#x5d;')"/>
148         <xsl:call-template name="clean">
149             <xsl:with-param name="text" select="substring-
150 after($text, '&#x5d;')"/>
151         </xsl:call-template>
152     </xsl:when>
153     <xsl:when test="contains($text, '&#x2e;')"> <!-- Denna when ser till
154 så att det blir en ny rad efter varje punkt, dvs. före varje ny definition
155 -->
156         <xsl:value-of select="substring-before($text, '&#x2e;')"/>.<br/>
157         <xsl:call-template name="clean">
158             <xsl:with-param name="text" select="substring-
159 after($text, '&#x2e;')"/>
160         </xsl:call-template>
161     </xsl:when>
162     <xsl:when test="contains($text, 'See also:')"> <!-- Denna when plockar
163 bort uttrycket "See also:" -->
164         <xsl:value-of select="substring-before($text, 'See also:')"/>
165         <xsl:call-template name="clean">
166             <xsl:with-param name="text" select="substring-after($text, 'See
167 also:')"/>
168         </xsl:call-template>
169     </xsl:when>
170     <xsl:otherwise>
171         <xsl:value-of select="$text"/>
172     </xsl:otherwise>
173     </xsl:choose>
174 </xsl:template>
175
176     <xsl:template match="abbreviations"> <!-- Gör så att inte innehållet i
177 abbreviationstaggens skrivs ut högst upp på sidan. -->
178     </xsl:template>
179 </xsl:stylesheet>
```


Bilaga O. DTD-filen till huvuduppgiften

```

1  <?xml version="1.0" encoding="utf-8"?>
2
3  <!ELEMENT xdx (full_name,description,abbreviations,ar+)>
4  <!ATTLIST xdx lang_from CDATA #REQUIRED>
5  <!ATTLIST xdx lang_to CDATA #REQUIRED>
6  <!ATTLIST xdx format CDATA #REQUIRED>
7
8      <!ELEMENT full_name (#PCDATA)>
9
10     <!ELEMENT description (#PCDATA)>
11
12     <!ELEMENT abbreviations (abr_def*)>
13
14         <!ELEMENT abr_def (k*,v)>
15
16             <!ELEMENT k (#PCDATA|opt|nu)* >
17
18             <!ELEMENT v (#PCDATA)>
19
20     <!ELEMENT ar (#PCDATA|k|def|pos|tr|dtrn|kref|rref|ex)* >
21     <!ATTLIST ar f CDATA #IMPLIED>
22
23         <!ELEMENT opt (#PCDATA)>
24
25         <!ELEMENT nu (#PCDATA)*>
26
27     <!ELEMENT pos (abr)>
28
29         <!ELEMENT abr (#PCDATA)>
30
31     <!ELEMENT tr (#PCDATA)>
32
33     <!ELEMENT rref (#PCDATA)>
34     <!ATTLIST rref start CDATA #REQUIRED>
35     <!ATTLIST rref size CDATA #REQUIRED>
36
37     <!ELEMENT ex (#PCDATA|iref)*>
38
39         <!ELEMENT iref (#PCDATA)>
40
41     <!ELEMENT dtrn (#PCDATA)*>
42
43     <!ELEMENT kref (#PCDATA)>
44
45     <!ELEMENT def (#PCDATA|ex|dtrn)*>

```

Bilaga P. PHP-filen med CSS-arrayen

```
1  <?
2  $styleSheets = array();
3
4  // Här definieras mina länkar till mina olika css-filer
5  $styleSheets[0]["name"]='Purple';
6  $styleSheets[0]["sheet"]='<link href="xdxf.css" rel="stylesheet"
7  type="text/css" />';
8
9  $styleSheets[1]["name"]='Green';
10 $styleSheets[1]["sheet"]='<link href="green.css" rel="stylesheet"
11 type="text/css" />';
12
13 $styleSheets[2]["name"]='Orange';
14 $styleSheets[2]["sheet"]='<link href="orange.css" rel="stylesheet"
15 type="text/css" />';
16
17 $styleSheets[3]["name"]='Blue';
18 $styleSheets[3]["sheet"]='<link href="blue.css" rel="stylesheet"
19 type="text/css" />';
20
21 // Denna css-fil används om inget annat är satt.
22 $defaultStyleSheet=0;
23
24 // Denna sats sätter vilket stylesheet som skall användas.
25 if(!isset($_COOKIE["STYLE"])){
26     if(isset($_SESSION["STYLE"])){
27         echo $styleSheets[$_SESSION["STYLE"]]["sheet"];
28     }else{
29         echo $styleSheets[$defaultStyleSheet]["sheet"];
30     }
31 }else{
32     echo $styleSheets[$_COOKIE["STYLE"]]["sheet"];
33 }
34 ?>
```

Bilaga Q. PHP-filen som skapar en cookie

```
1  <?
2  // Denna sats sparar en cookie med vald layout i 1 år.
3  if(isset($_REQUEST["SETSTYLE"])){
4      if(setcookie("testcookie",true)){
5          setcookie("STYLE",$_REQUEST["SETSTYLE"],time()+31622400);
6      }else{
7          $_SESSION["STYLE"]=$_REQUEST["SETSTYLE"];
8      }
9  }
10
11 // Återvänder till sidan som kallade på detta PHP.
12 header("Location: ".$_SERVER["HTTP_REFERER"]);
13 ?>
```

Bilaga R. CSS-filen till huvuduppgiften

```
1  html{      font-size:125%;}
2  body{      font-size: 50%;
3             background-image:url('holes.png');
4             font-family: Arial;}
5
6  table,td,th{ border:2px solid #9900CC;
7               border-collapse:collapse;
8               padding:5px;
9               font-size: 1.1em;}
10
11 table{ margin: 10px auto 20px;
12        clear:both;}
13
14 th{ background-color:#BD59DE;}
15
16 td{ background-color:#FFFFFF;}
17
18 a{ color:#FFFFFF;
19    font-weight:bold;}
20
21 /*Klasser*/
22 .heading{ color:#FAF2FC;
23           background-color:#9900CC;
24           font-size: 2.0em;
25           text-transform:uppercase;}
26
27 .about{ color: #EBCCF5;
28         background-color:#9900CC;
29         font-size: 1.4em;
30         font-style: italic;
31         text-align: center;}
32
33 .word{ text-transform:capitalize;}
34
35 .no{color:#999999;}
36
37 .validate{ color:#FFFFFF;
38            font-size:1.5em;
39            margin:20px 0 20 10px;
40            float:left;}
41
42 /*Boxar*/
43 #colors{ color:#FFFFFF;
44          font-size:1.5em;
45          margin:20px 0 20 10px;
46          float:left;}
```

Bilaga S. Slutresultatet (http://wwwlab.iki.his.se/~a11emmjo/xml-api/Upppg%20*/xdxf.php)

Validation of DTD successful! Choose your preferred color-scheme: [Purple](#) [Green](#) [Orange](#) [Blue](#)

WEBSTER'S UNABRIDGED DICTIONARY					
Webster's Unabridged Dictionary published 1913 by the...					
word	phonetic	type	definition	example	soundfile
Record	[re'kɔrd]	n.	Anything written down and preserved. To write down for future use.	no example found	no sound found
Home	[ho:um]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. 4) at home - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Home	[houm]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. For without hearts there is no home. 4) at ome - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Indices	no phonetic	no type	Plural form of word	no example found	no sound found
Disc	no phonetic	n.	A flat, circular plate; as, a disk of metal or paper.	no example found	no sound found
CO 2	no phonetic	no type	Carbon dioxide (CO2) - a heavy odorless gas formed during respiration.	no example found	no sound found

abbreviations

Av.	Avenue	n.	noun	v.	verb
-----	--------	----	------	----	------

Validation of DTD successful! Choose your preferred color-scheme: [Purple](#) [Green](#) [Orange](#) [Blue](#)

WEBSTER'S UNABRIDGED DICTIONARY					
Webster's Unabridged Dictionary published 1913 by the...					
word	phonetic	type	definition	example	soundfile
Record	[re'kɔrd]	n.	Anything written down and preserved. To write down for future use.	no example found	no sound found
Home	[ho:um]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. 4) at home - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Home	[houm]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. For without hearts there is no home. 4) at ome - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Indices	no phonetic	no type	Plural form of word	no example found	no sound found
Disc	no phonetic	n.	A flat, circular plate; as, a disk of metal or paper.	no example found	no sound found
CO 2	no phonetic	no type	Carbon dioxide (CO2) - a heavy odorless gas formed during respiration.	no example found	no sound found

abbreviations

Av.	Avenue	n.	noun	v.	verb
-----	--------	----	------	----	------

Bilaga R

Validation of DTD: successful! Choose your preferred color-scheme: [Purple](#) [Green](#) [Orange](#) [Blue](#)

WEBSTER'S UNABRIDGED DICTIONARY					
Webster's Unabridged Dictionary published 1913 by the...					
word	phonetic	type	definition	example	soundfile
Record	[re'kɔrd]	n.	Anything written down and preserved. To write down for future use.	no example found	no sound found
Home	[ho:um]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. 4) at home - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Home	[houm]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. For without hearts there is no home. 4) at ome - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Indices	no phonetic	no type	Plural form of word	no example found	no sound found
Disc	no phonetic	n.	A flat, circular plate; as, a disk of metal or paper.	no example found	no sound found
CO 2	no phonetic	no type	Carbon dioxide (CO2) - a heavy odorless gas formed during respiration.	no example found	no sound found

abbreviations

Av.	Avenue	n.	noun	v.	verb
-----	--------	----	------	----	------

Validation of DTD: successful! Choose your preferred color-scheme: [Purple](#) [Green](#) [Orange](#) [Blue](#)

WEBSTER'S UNABRIDGED DICTIONARY					
Webster's Unabridged Dictionary published 1913 by the...					
word	phonetic	type	definition	example	soundfile
Record	[re'kɔrd]	n.	Anything written down and preserved. To write down for future use.	no example found	no sound found
Home	[ho:um]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. 4) at home - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Home	[houm]	n.	1) One's own dwelling place; the house in which one lives. 2) One's native land; the place or country in which one dwells. 3) The abiding place of the affections. For without hearts there is no home. 4) at ome - make yourself at home -	For without hearts there is no home.	sounds_of_words.ogg filesize: 512kB
Indices	no phonetic	no type	Plural form of word	no example found	no sound found
Disc	no phonetic	n.	A flat, circular plate; as, a disk of metal or paper.	no example found	no sound found
CO 2	no phonetic	no type	Carbon dioxide (CO2) - a heavy odorless gas formed during respiration.	no example found	no sound found

abbreviations

Av.	Avenue	n.	noun	v.	verb
-----	--------	----	------	----	------