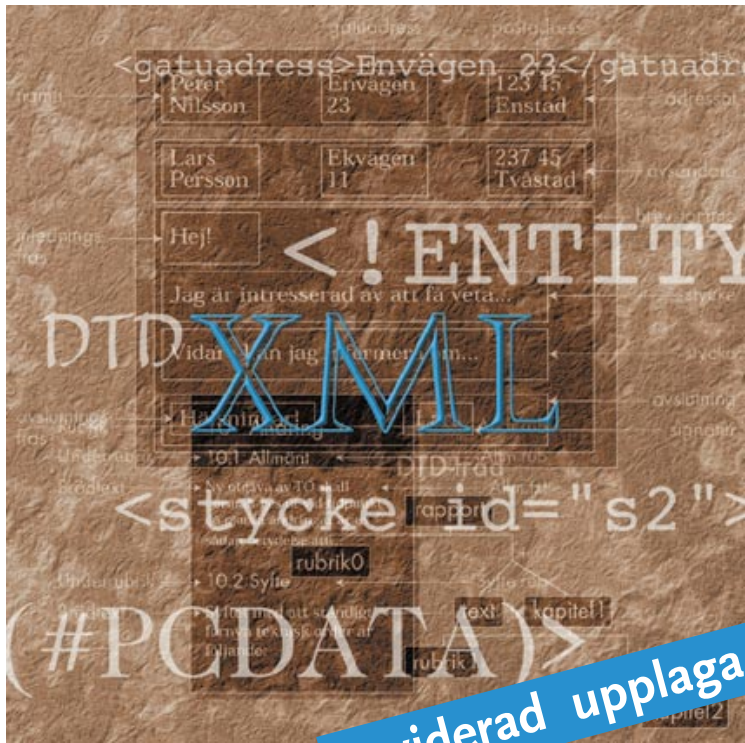


# VAD ÄR XML?



# VAD ÄR XML?

REVIDERAD UPPLAGA 1999

Publikationen kan beställas från:

Statskontoret  
Publikationsservice  
Box 2280  
103 17 Stockholm  
Tfn: 08-454 46 43  
Tfx: 08-454 46 45  
E-post: publikations.service@statskontoret.se

Mer information om Statskontoret finns på Internet:  
[www.statskontoret.se](http://www.statskontoret.se)

© STATSKONTORET  
ISBN: 91-7220-340-4  
Reviderad upplaga  
Layout: Dreamforce Infomedia, Solna  
Tryck: Williamssons Offset, Solna 1999

## Innehåll

<b>1 Inledning</b> .....	5
<b>2 HTML och SGML</b> .....	6
<b>2.1 HTML</b> .....	6
2.1.1 Varför HTML? .....	6
2.1.2 Vad är problemet med HTML?.....	7
<b>2.2 SGML</b> .....	9
2.2.1 Varför SGML? .....	9
2.2.2 Vad är då problemet med SGML? .....	11
<b>3 XML</b> .....	12
<b>3.1 XML, lösningen på problemen?</b> .....	12
<b>3.2 Lagringsformat</b> .....	13
<b>3.3 Uppbyggnad</b> .....	13
3.3.1 Märkord och element.....	13
3.3.2 Attribut .....	15
3.3.3 Entiteter och notationer .....	16
<b>3.4 Struktur</b> .....	18
3.4.1 DTD-syntax.....	20
3.4.2 Samma information – olika struktur.....	21
<b>3.5 Presentation</b> .....	23
<b>3.6 Länkar</b> .....	25
3.6.1 Utvecklingen av XML.....	25
3.6.2 Fördefinierade attribut för XML-länkar .....	26
3.6.3 Enkla och utökade länkar .....	27
3.6.4 Länkar inuti dokument .....	28
<b>3.7 Verktyg</b> .....	29
<b>4 XML och tillämpningar</b> .....	31
<b>4.1 Tillämpningsområden</b> .....	31
<b>4.2 XML som universalt dataformat</b> .....	32
4.2.1 XML och EDI .....	33
4.2.2 XML och Java .....	34
<b>4.3 Metainformation</b> .....	35
<b>4.4 Exempel</b> .....	37
<b>5 Bedömning av framtidsutsikter</b> .....	40
<b>6 Referenser</b> .....	42
<b>6.1 WWW-referenser</b> .....	42
<b>6.2 Litteratur</b> .....	42
<b>6.3 Övriga publikationer</b> .....	43
<b>7 Ordförklaringar</b> .....	44
<b>8 Statskontorets publikationer för 1998/1999</b> .....	45

# 1 INLEDNING

Introduktionen av HTTP (HyperText Transport Protocol) och HTML (HyperText Markup Language), utgjorde tillsammans grunden för den explosionsartade utveckling av Internet som ledde till World Wide Web.

Utvecklingen fortsätter med hög fart men hålls möjligen tillbaka av att HTML inte riktigt räcker till för att skapa mer avancerade tillämpningar. En lösning på detta finns nu tillgänglig. XML (eXtensible Markup Language) är ett mer avancerat språk, eller egentligen ett metaspråk, för att hantera information på ett strukturerat sätt. Medan man i HTML arbetar med en fast uppsättning märkord, är XML ett språk som låter användaren definiera sina egna märkord, XML är utvidgningsbart (extensible).

XML bygger på väl beprövade tekniker, och utgör en delmängd och specialanpassning till Internet av Standard Generalized Markup Language (SGML). SGML som är mer omfattande och mer komplext har en stor men i allmänhet inte särskilt känd användning inom telekomindustrin, fordonsindustrin, läkemedelsindustrin, försvarsindustrin m.m. Framtagningen av den tekniska rekommendationen XML har skett inom ramen för World Wide Web Consortium (W3C). Konsortiet har en rad riktigt tunga medlemsföretag som till exempel Microsoft, Netscape, Sun samt de ledande inom SGML-branschen. Kring XML, som blev en officiell rekommendation från W3C i februari 1998, råder idag en närmaste total uppslutning.

Denna rapport vänder sig i första hand till anställda inom offentlig sektor som arbetar med informationshantering anknuten till WWW. Vi vill uppmärksamma Er på den aktuella utvecklingen, ge en kort introduktion till XML och peka på de möjligheter som XML erbjuder. Mycket talar för att XML kommer få avgörande betydelse för Internets fortsatta utveckling, vi tror också att det i hög grad kommer att påverka hur man hanterar sin information internt. Det finns med andra ord starka skäl för offentlig sektor att snabbt tillägna sig denna nya teknik.

Rapporten har utarbetats av Statskontorets projekt Elektronisk dokumenthantering inom IT-enhetens uppdragsområde Elektronisk infrastruktur. Rapporten som är en ny reviderad utgåva har utformats i samarbete mellan Jan Lundh, projektledare och Daniel Björkman och Marie Österberg, Enator Information Management AB.

## 2 HTML OCH SGML

Utvecklingen av XML är resultatet av ett arbete där tanken har varit att komplettera enkelheten hos HTML med flexibiliteten hos SGML.

HTML, som även XML, specificeras idag av W3C ([www.w3.org/](http://www.w3.org/)) och är det språk som används för den mesta informationen på webben.

SGML är en internationell ISO-standard (8879-1986) som definierar ett språk för att beskriva innehåll och struktur i dokument på ett applikations- och plattformsoberoende sätt. Det finns idag en rad tillämpningar av SGML, bl.a. för att hantera teknisk dokumentation och juridisk information i elektroniska dokument.

### 2.1 HTML

De flesta dokumenten på webben är kodade i HTML. För att läsa dessa använder man en webbläsare, t. ex. Netscape eller Internet Explorer. "Webben" är en mycket stor samling sammanlänkade dokument som sträcker sig över hela världen.

#### 2.1.1 Varför HTML?

Syftet med HTML var att göra information allmänt tillgängligt och oberoende av någon enskild leverantörs programvaror. Användningen av HTML har lett till att man som läsare och användare kan ta del av en mängd information oavsett var den publiceras i världen, vilken webbläsare man använder och vilken dator man har.

När man skapar en HTML-sida använder man märkord. Exempel på märkord som finns i HTML är `<BODY>`, `<H1>`, `<TABLE>`, `<PARA>`. Så här ser det ut när man beskriver en rubrik på nivå ett med hjälp av HTML:

```
<H1>Detta är en rubrik på nivå ett</H1>
```

HTML har en fast uppsättning märkord. Leverantörsspecifika tillägg finns, men gemensamma specifikationer har utvecklats av W3C (<http://www.w3.org/MarkUp/>). Aktuell version är 4.0.

### 2.1.2 Vad är problemet med HTML?

Problemet med HTML har sin grund i att man bara har tillgång till en begränsad mängd fördefinierade märkord. Dessa märkord räcker inte alltid till för att skapa mer avancerade presentationer och tillämpningar.

När webben fick sitt verkliga genombrott i mitten på 90-talet var det huvudsakliga användningsområdet publicering. Alla företag och organisationer skulle ha en egen hemsida där man ofta presenterade övergripande information om verksamheten. I takt med den tekniska utvecklingen och en ökande mognad hos användarna började Internettekniken användas även internt. I intranäten handlade det fortfarande i huvudsak om publicering men också om tillämpningar för t. ex. bokning av möteslokaler, ekonomisk rapportering och kvalitetssäkring. Man började också koppla databaser till tillämpningarna på intranäten. Efter en tids provande vågade man sig även på att bygga Internettillämpningar där man släppte in användarna direkt mot de interna systemen genom väldefinierade gränssnitt, bl. a såg de första Internetbankerna dagens ljus.

Fler och fler började upptäcka att HTML hade ganska stora begränsningar men det gick ofta att komma runt problemen. Den största utmaningen var att hålla systemen uppdaterade, i takt med utvecklingen av nya webbläsare, utökning av HTML mm. Idag har vi tagit ytterligare ett steg upp för komplexitetstrappan i Internetutvecklingen och det finns gott om exempel på verksamhetskritiska tillämpningar på Internet. Som exempel kan nämnas Internetbankerna, elektroniska affärer, spelverksamhet etc. Självfallet är det helt avgörande att tekniken bakom de här lösningarna är pålitlig, att det är lätt att uppdatera informationen, att säkerheten är tillförlitlig och att systemen är skalbara när användarna blir fler. En timmes avbrott i ett verksamhetskritiskt system kan få stora konsekvenser.

Det är här XML kommer in i bilden, som ett mer avancerat format för att hantera information. HTML klarar helt enkelt inte av att leva upp till alla dessa krav eftersom det är utvecklat för att presentera relativt enkel information, inte för att bygga verksamhetskritiska tillämpningar på. Att utöka antalet märkord i HTML är inte möjligt eftersom varje verksamhet behöver sina egna märkord.

Många leverantörer av HTML-verktyg har utökat HTML med sina egna märkord för att överbrygga begränsningar och skapa nya

möjligheter att presentera information. Dessa ansträngningar har lett till viss inkompatibilitet mellan olika HTML-verktyg. Resultaten av detta syns på HTML-sidor med ikoner innehållande texter som "Denna sida visas bäst i Internet Explorer" eller "Denna sida visas bäst i Netscape". Skaparen av en sådan HTML-sida har valt att använda märkord som endast stöds av en viss leverantör. Detta leder till att en HTML-sida som ser bra ut i ett verktyg kanske knappt går att läsa i ett annat. Den ursprungliga intentionen att alla dokument skall vara läsbara världen över oavsett vilken typ av webbläsare man uppnås inte längre i samma grad.

Strukturen som definieras i HTML-specifikationen är mycket bristfällig. Det innebär att man mycket väl kan skriva syntaktiskt "korrekt" HTML som strukturellt sett är ganska egendomlig. Innehållet i HTML-märkordet <BODY> har definierats så att alla märkord som får användas kan placeras i vilken ordning som helst. Ett exempel på syntaktiskt korrekt HTML är:

```
<BODY>
<P>Denna rapport beskriver XML som är...</P>
<H3>Inledning</H3>
<H1>Avslutning</H1>
</BODY>
```

En paragraf <P> kan mycket väl förekomma utan någon inledande rubrik i HTML. Likaväl kan en rubrik på nivå 3 <H3> uppträda före en rubrik på nivå 1 <H1>. Det här kan man ju tycka är en fördel eftersom det ger utrymme för att använda märkorden på en mängd kreativa sätt. Märkordet <H3> behöver ju inte alls innehålla en rubrik på nivå 3 utan kan mycket väl innehålla ett namn på en produkt som skaparen av HTML-dokumentet vill ha i lite större stil. Många skapare av HTML-dokument har visat prov på mycket stor kreativitet när det gäller att använda HTML-märkord för att formatera ett dokument på ett sätt som ursprungligen inte var tänkt att stödjas. Exempel på detta är att t ex. tabeller används för att indentera stycken och osynliga bilder används för att formatera text. Eftersom det är svårt att göra avancerad formatering av HTML-kodad text tvingas man i stället att använda bilder. Problemet med detta är att HTML-sidorna blir svåra att skapa och underhålla, dessutom blir nedladdningstiderna långa eftersom grafik tar mycket utrymme.

En annan konsekvens av HTMLs bristande interna struktur är att det är svårt att skapa program som bearbetar informationen med automatik. Antag att någon vill göra ett program som skapar en innehållsförteckning genom att läsa alla rubriker som ingår i ett dokument. Innehållsförteckningen blir minst sagt förvirrande om det dyker upp annan text än just rubriker, dvs. om man har använt märkordet för rubrik till något annat än en rubrik. Det kan också vara svårt att formatera en innehållsförteckning med olika indrag för varje rubriknivå eftersom man inte vet om rubriknivåerna används på avsett vis. HTMLs avsaknad av struktur leder också till att användaren inte kan få stöd i en HTML-editor angående vilka märkord som får infogas. HTML-editorn kan inte avgöra (utifrån HTML-specifikationen) vilka element som får förekomma i vilken ordning.

## 2.2 SGML

Standard Generalized Markup Language (SGML) är ett språk för att definiera ett dokumentets struktur och koda informationen på ett applikations- och plattformsoberoende sätt. SGML är sedan 1986 en internationell ISO-standard.

SGML används idag i större omfattning än vad som är allmänt känt. De främsta användningsområdena är stora tillämpningar med komplex information med lång livslängd. SGML används bland annat inom telekomindustrin, fordonsindustrin, läkemedelsindustrin, försvarsindustrin och nyhetsbranschen m.m.

### 2.2.1 Varför SGML?

SGML har utvecklats med intentionen att lösa ett antal konkreta problem. Tanken är att SGML skall underlätta utbytet av information mellan användare av olika datorer och plattformar, underlätta återanvändning av information, fungera som en mall för att strukturera information samt ge förutsättningar för att skapa mer intelligenta informationslösningar.

#### 2.2.1.1 Utbyte av information

Under 1970-talet och början av 80-talet upplevde man stora problem med att utbyta information mellan olika datorer och system. I vissa fall var det omöjligt och man fick helt enkelt skapa informationen på

nytt, i andra fall kunde man lösa det med konverteringar. Idag klarar de flesta system att utbyta information i ASCII-format. Problemet är dock på intet sätt löst, alla som har provat vet att det ofta är helt smärtfritt att flytta data mellan t ex. PC, Mac och Unix. Ofta är det t.o.m. problem att flytta data mellan olika versioner av en viss programvara. Hur skall man kunna säkerställa att informationen kan läsas om tio, tjugo eller trettio år när vi har problem att läsa information som är ett år gammal?

### **2.2.1.2 Återanvändning**

Även teknikutveckling utanför IT-området går fort och många produkter blir mer och mer komplexa, vilket ställer ökade krav på bl.a. den tekniska dokumentationen. Dokumentationen blir avgörande för att man skall kunna förstå och använda produkten fullt ut.

Att ta fram avancerad teknisk dokumentation, som är det område där SGML i första hand kommit att användas, är mycket kostsamt. Många organisationer har idag insett att återanvändning av information har stor ekonomisk potential men också att stora mängder av den information som används inom en organisation går att återanvända.

Återanvändning innebär inte enbart att redan skriven text återanvänds, det handlar även om att information skall kunna publiceras på olika medier utan alltför mycket arbete. Ett exempel på detta kan t ex. vara en organisation som vill distribuera information på papper, CD-ROM, fax, intranet och WWW i princip samtidigt. För att detta enkelt skall kunna uppnås förutsätts att sakinformationen är separerad från information som styr utformning av layout.

Återanvändning är också starkt förknippat med sökning eftersom man måste hitta informationen för att kunna återanvända den. För att kunna återanvända informationen måste den märkas upp på ett sätt som gör att vi vet vad den handlar om. Man behöver också bryta ner informationen i mindre komponenter så att man kan återanvända delar av den i olika sammanhang.

### **2.2.1.3 Strukturerat arbetssätt**

Ett annat argument för SGML är att det stödjer ett strukturerat arbetssätt. Med tanke på kostnaderna för framtagandet av information finns intresse av att författarna av dokumenten inte skall

behöva ägna sig åt layout utan kunna koncentrera sig på innehållet. Användandet av SGML innebär ofta också att man anammar ett mer ingenjörsmässigt synsätt på informationen.

Med SGML finns möjlighet att ge författarna tillgång till mallar som definierar struktur och layout. Mallarna bygger i sin tur på noggranna dokumentanalyser utförda av specialister. Eftersom alla i ett företag använder samma mallar får all dokumentation ett likformigt utseende, vilket i sin tur gör att läsaren känner igen sig. Det räcker dock ofta inte att förse författaren med mallar, man vill dessutom kunna validera att författaren verkligen har använt mallen på korrekt sätt, dvs. att dokumentet är korrekt i formell mening. För att kunna validera dokument vill man bygga in regler för strukturen direkt i mallen vilket också stöds av SGML.

#### **2.2.1.4 Intelligent information**

Information kan presenteras på mer eller mindre intelligent sätt. Om vi väljer att beteckna ett pappersdokument som en "dum" informationsbärare så kan vi beteckna en avancerad multimediapresentation som en intelligent informationsbärare.

En förutsättning för mer avancerade tillämpningar är att informationen man skall bearbeta eller visa, finns beskriven på ett sätt som kan bearbetas av ett datorprogram. Information kodad enligt SGML är mycket väl lämpad att använda i till exempel multimediapresentationer eller elektroniska manualer.

#### **2.2.2 Vad är då problemet med SGML?**

SGML är en mycket omfattande standard som klarar att uppfylla de flesta önskemål vad gäller modularisering, strukturering och länkning av information. Priset för denna flexibilitet är en avsevärd komplexitet.

SGML är väl lämpat för att hantera komplicerad teknisk dokumentation, framförallt i elektroniskt format, t ex. i form av elektroniska manualer för handhavande- och underhåll för datorer, bilar, fartyg eller flyg. Teknikinformatörer som arbetar med att ta fram tekniskt avancerad information har ofta ett mer ingenjörsmässigt synsätt på informationen än vad författare till mer "vardaglig" information har. Den tekniska informationen bryts ner i moduler och kanske t.o.m. i fragment som sedan kan återanvändas i olika infor-

mationsprodukter. Presentationen kan utformas olika beroende på behov. Man kan till exempel förse informationen med olika utseende beroende på om den presenteras på papper, CD-ROM eller WWW. Ofta har de som arbetar med SGML och teknisk dokumentation en mycket hög ambitionsnivå när det gäller att hantera sin information på ett strukturerat sätt. Anledningen till detta är att informationen är mycket kostsam att ta fram och att den skall leva och därmed underhållas under lång tid

SGML är inte framtaget för att användas på Internet. En HTML-läsare kan idag inte tolka de märkord som författaren till SGML-dokumentet själv har definierat. Idag löser man ofta problemet genom att översätta SGML-dokument till HTML som därmed kan läsas i en HTML-läsare. Det innebär i praktiken att man översätter sina egendefinierade märkord med de som finns tillgängliga i HTML. Sådan översättningar är relativt enkla att åstadkomma eftersom den SGML-kodad information kan bearbetas på automatisk väg.

Dock är det många som idag arbetar med SGML och har anammat SGML:s sätt att tänka som upplever det som mycket otillfredsställande att behöva konvertera informationen till HTML innan den presenteras. Man går från ett rikare och mer strukturerat format till ett fattigare format i princip utan struktur. Istället för att konvertera till HTML skulle man önska att det vore möjligt att presentera sin SGML-information direkt i en webbläsare.

Ett avgörande problem är också att det visat sig vara svårt att utveckla programvaror för SGML. Standarden är mycket komplex och tillåter en mängd specialfall, bland annat avseende hur man får koda sina märkord i ett SGML-dokument. Bra och billiga verktyg för SGML-produktion har därför låtit vänta på sig och för bredare kretsar är användbarheten hos dagens SGML-editorer alltför låg.

## 3 XML

### 3.1 XML, lösningen på problemen?

Uppenbarligen finns ett glapp mellan den omfattande och komplexa SGML-standarden och den enkla och fattiga HTML-standard. XML har utvecklats för att fylla detta tomrum. XML är en delmängd av SGML som särskilt utvidgats för användning på Internet. I XML

har uteslutits det som i SGML varit komplext och svårt att förstå och att bygga programvaror för. Det viktigaste hos SGML, flexibiliteten (att själv kunna bestämma märkordsuppsättning och struktur), finns kvar i XML.

I detta kapitel beskrivs XML närmare. Vi går igenom lagringsformat, syntax, länkar med mera. På de webbplatser som anges i kapitel 8 finns material för vidare studier.

## 3.2 Lagringsformat

För att åstadkomma plattformsoberoende skrivs HTML- och SGML-dokument i ASCII (American Standard Code for Information Interchange), som är en av de mest använda standardiserade koderna inom datortekniken. Koden består av sju binära siffror, vilket ger möjlighet att representera 128 alfanumeriska tecken.

I XML skall man använda ISO-standarderna ISO 10646 (Unicode), vilken är en internationell teckenstandard som använder 31 binära siffror. Med denna kan man representera alla tecken, inklusive till exempel kinesiska.

En XML-fil är en enkel textfil. Man kan således läsa innehållet i en XML-fil utan XML-programvara genom att öppna den i en vanlig texteditor. Syntaxen för en XML-fil är så enkel att man kan skriva filen i en vanlig editor även om det givetvis är betydligt enklare med en XML-editor som infogar märkorden med korrekt syntax åt dig.

## 3.3 Uppbyggnad

### 3.3.1 Märkord och element

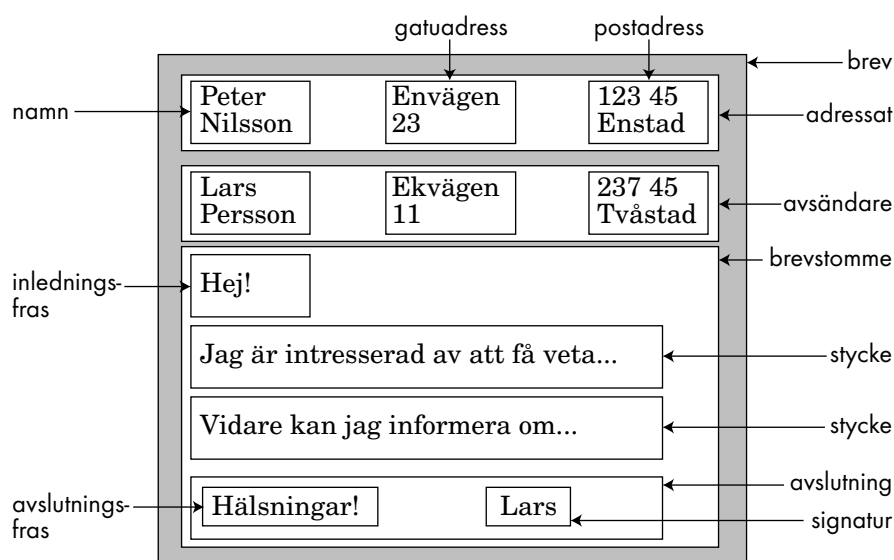
XML-dokument består dels av själva innehållet och dels av uppmärkningen av innehållet. Det som man brukar kalla märkord är det mest centrala när det gäller uppmärkningen. De används så att märkordsnamnet återspeglar den information märkordet ringar in. För att bildligt beskriva märkord kan man tänka sig märkorden som lådor som håller informationen. En stor byrå representerar hela XML-dokumentet. Denna byrå innehåller en eller flera lådor och varje låda kan i sig innehålla fler lådor eller ha innehåll i form av text direkt. För att illustrera detta tänker vi oss ett antal märkord för innehållet i ett brev. Brev kan tänkas bestå av adressat, avsändare och brevstomme. Adressat och avsändare består i sin tur t ex.

av namn, gatuadress och postadress. Själva brevstommen kan tänkas bestå av inledningsfras, ett antal stycken och avslutning. Se bild nedan.

Märkorden illustrerade som lådor i en byrå full med information.

Hur kodar man märkorden i XML-dokument? Låt oss börja med ett enskilt märkord. Antag att vi som i exemplet ovan behöver ett märkord som talar om att vårt innehåll avser en gatuadress. Vi väljer därför att sätta namnet på vårt märkord till "gatuadress". I vårt XML-dokument kommer detta att skrivas som:

"<gatuadress>Envägen 23</gatuadress>"



<gatuadress> anger början på märkordet och kallas starttagg.

</gatuadress> anger slutet och kallas sluttagg.

Märkorden tillsammans med sitt innehåll kallas för element. "<gatuadress>Envägen 23</gatuadress>" är alltså ett element med namnet "gatuadress" och innehållet "Envägen 23".

Vårt brexempel skulle i XML-dokumentet se ut som följande:

<brev>

<adressat>

```

<namn>Peter Nilsson</namn>
<gatuadress>Envägen 23</gatuadress>
<postadress>123 45 Enstad</postadress>
</adressat>
<avsändare>
<namn>Lars Persson</namn>
<gatuadress>Ekvägen 11</gatuadress>
<postadress>237 45 Tvåstad</postadress>
</avsändare>
<brevstomme>
<inlednfras>Hej!</inlednfras>
<stycke>Jag är intresserad av att få ...</stycke>
<stycke>Vidare kan jag informera om ...</stycke>
<avslutning>
<avslutnfras>Hälsningar!</avslutnfras>
<signatur>Lars</signatur>
</avslutning>
</brevstomme>
</brev>

```

I exemplet ovan omsluter elementet `brev` de andra elementen. Detta är en regel i XML som måste följas, ett enda element måste omsluta alla de andra. I exemplet är inte all syntax angiven som krävs för ett fullständigt XML-dokument.

I ett XML-dokument finns ingen information om vilken layout, såsom typsnitt, storlek eller färg, olika informationsdelar skall ha. Även om märkordsnamnet talar om att det är en rubrik på nivå 1 (t ex. `rubrik1`) finns det alltså ingen uppgift i XML-filen som säger hur denna rubrik skall se ut (t ex. att den skall skrivas med Helvetica, fetstil och ha storleken 18 punkter). Anledningen till detta är att man vill särskilja layouten från innehållet. I kapitel 3.5 beskrivs detta närmare, dessutom beskrivs kopplingen av information till layout.

### 3.3.2 Attribut

Attribut är en möjlighet att sätta en egenskap på ett element, att ge en ytterligare bestämning av sitt element. Attribut kan användas i en mängd olika sammanhang. Ett vanligt användningsområde från SGML-världen är att använda attribut för att unikt identifiera

märkord, vilket till exempel behövs för att målet på en länk skall bli unikt:

```
<stycke id="s2">
```

Som framgår av exemplet ovan skrivs attributet inom elementets starttagg. Attributets namn är i exemplet id och har värdet s2. Värdet skrivs alltid inom citationstecken (till skillnad mot HTML där de kan uteslutas) och föregås av likhetstecken (=). Fler exempel på användning av attribut är möjligheten att versionshantera en rapport och ange status på ett dokument. Se dessa och fler exempel nedan:

- Ange versionsnummer för en rapport t ex. <rapport ver="1.2">
- Ange ett dokument eller elements status t ex. <memo status="utkast">
- Identifiera text som skall genereras av systemet t. ex. numrering av lista t ex. <lista typ="nummer">
- Sätta en bestämning på ett elements innehåll, t ex. valuta för en prisuppgift t ex. <pris valuta="SEK">599</pris>

### 3.3.3 Entiteter och notationer

Entiteter och notationer används för att på olika sätt infoga informationsobjekt i ett XML-dokument. En **entitet** kan ses som en slags "behållare" med ett namn som man själv anger. Denna "behållare" kan innehålla olika informationsobjekt såsom dokument, bilder, fraser, ljud och video. Det finns flera olika typer av entiteter, men vi nöjer oss här med exemplifiera vad några typer av entiteter kan användas till.

Ett exempel på användning av entiteter är en möjlighet att definiera sina egna förkortningar för att underlätta skrivandet. Vid presentationen av dokumentet vill man att förkortningen ersätts av det kompletta uttrycket. Antag till exempel att man skriver om XML, men att man anser att det är bättre att skriva ut hela eXtensible Markup Language vid presentationen istället för förkortningen. Man kan då deklarerar entiteten XML. Det gör man genom att använda det i XML reserverade ordet ENTITY.

<!ENTITY XML "eXtensible Markup Language">

Entiteten XML är nu deklarerad med innehållet "eXtensible Markup Language". Om man nu i sitt XML-dokument skriver t ex.

<p>Standarden &XML; kommer att ha stor ...</p>

kommer detta i en webb-läsare att presenteras som:

Standarden eXtensible Markup Language kommer att ha stor...

Denna typ av entiteter används i själva XML-specifikationen. Eftersom man i XML använder sig av tecknen <, >, &, ' och " i syntaxen för uppmärkningen av dokument kan man inte skriva t ex. "<" i löpande text. När man skriver "<" tolkas det som inledningen på ett märkord. För de här syntaxtecknen har man därför fördefinerat entitetsreferenser. Om man vill använda tecknet "<" i texten måste man följaktligen skriva &lt;. I de XML-editorer vi kommer att få se kan vi utgå från att användaren inte behöver bry sig om detta. XML-editorn kommer automatiskt att införa syntaxen för elementen och kommer att "veta" att vi verkligen vill ha tecknet "<" och inte avser att påbörja uppmärkningen av ett nytt element.

Med entiteter kan man även underlätta underhållet av ett dokument. Antag att man från ett XML-dokument refererar till ett annat XML-dokument ett flertal gånger. För varje referens till det andra dokumentet anges det andra dokumentets filnamn. Ett alternativ är att istället definiera en entitet för det andra dokumentet och referera till entiteten. En fördel är att om namnet på det andra dokumentet eller dess lagringsplats ändras behöver man bara ändra i entiteten, inte på flera ställen i dokumentet.

**Notationsdeklarationer** används för att identifiera andra format än XML-formatet. De används t ex. för att specificera vilka bildformat som kan förekomma i informationsmängden. Exemplet nedan visar hur en notationsdeklaration för en bitmappbild kan se ut. En notationsdeklaration skrivs i DTDn.

<!NOTATION BMP SYSTEM "bmpview.exe">

För den bild man vill lägga in skapas i XML-dokumentet en entitet som talar om att informationen är av typen BMP. NDATA är ett reserverat ord och betyder Notation Data. Det måste anges då man definierar entiteter för annat än XML-kodad information.

<!ENTITY logohome SYSTEM "logohome.bmp" NDATA BMP>

### 3.4 Struktur

En av de stora fördelarna med XML jämfört med HTML är att man kan skapa sina egna märkord och sätta egna namn på dem. Men förmodligen är det inte meningsfullt för en applikation att märkorden får förekomma i en fullständigt godtycklig ordning. Därför finns möjligheten att specificera vilka märkord som får förekomma och i vilken ordning de får förekomma. Med andra ord kan man specificera vilken struktur ett dokument skall ha. Strukturen definieras i något som kallas DokumentTypsDefinition, vilket förkortas DTD.

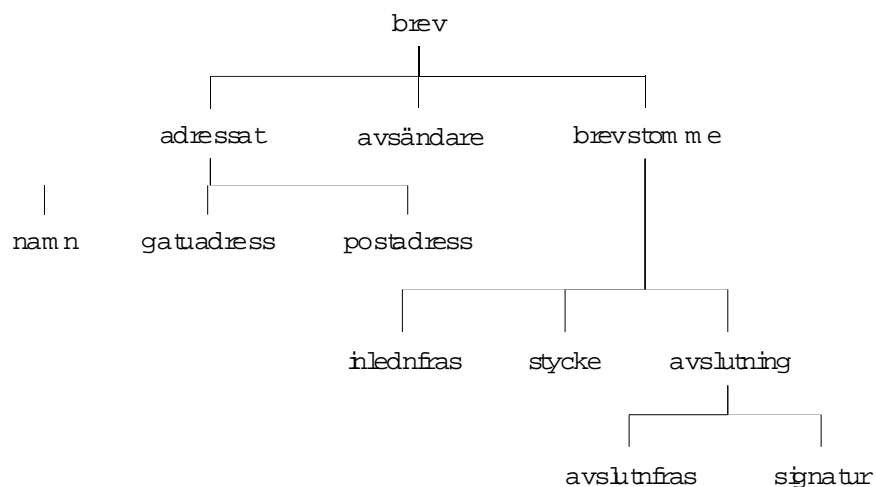
En DTD innehåller regler för hur informationen är strukturerad:

- Definierar vilka element som förekommer i informationen.
- Bestämmer hur dessa element får förekomma och ingå i varandra.
- Definierar vilka attribut som finns till varje element.
- Kan ses som ett träd, se nedan.

En DTD kan liknas vid hur ett upp- och nervänt träd som grenar ut sig åt olika håll.

Man bestämmer helt själv hur uppsättningen regler skall se ut. Som namnet dokumenttypsdefinition säger skriver man en DTD för varje typ av dokument. Exempel på olika typer av dokument är t ex. fakturor, beslut, protokoll, SOU:er och patientjournaler. DTDn är mallen för en dokumenttyp.

I SGML är man tvungen att ha en DTD. I XML har man i många fall möjligheten att själv välja om man vill använda en DTD eller



inte. Till exempel kommer det i de flesta fall inte att behövas någon DTD för att kunna läsa ett XML-dokument i en webbläsare. När man skriver sitt dokument i en XML-editor är dock det naturligaste och effektivaste att ha en DTD till grund. XML-editorn läser då först in DTDn. Man får sedan mycket hjälp av editorn att märka upp dokumentet, det vill säga på varje nivå presenteras vilka märkord som är möjliga att använda. Om man tänker sig att skriva XML-dokument utan någon DTD måste man för XML-editorn explicit ange namnet på varje märkord.

En programvara kallad parser kontrollerar att den struktur man skrivit i XML-dokumentet överensstämmer med DTDn. I de XML-editorer vi kommer att få se kommer XML-parsern att vara inbyggd, vilket innebär att man under tiden man skriver får stöd för sitt skrivande, parsern är interaktiv.

DTDn och parsern säkerställer att alla dokument följer den givna strukturen, detta har stor betydelse vid maskinell behandling av dokumenten. Till dokument med samma DTD kan sedan knytas en eller flera layoutmallar. DTDn kan också ge stöd åt författaren att avgöra vilken information som skall skrivas in på vilket ställe i dokumentet.

När man läser om XML stöter man ofta på uttrycken ”**valid**” och ”**well-formed**”. Om ett XML-dokument sägs vara ”valid” innebär det att dokumentet är helt korrekt avseende syntaxen samt att det följer en viss angiven DTD. Med ”well-formed” menas att dokumentet är helt korrekt syntaktiskt men inte följer någon angiven DTD.

Behöver man då skapa en egen DTD när man skall skriva ett XML-dokument? Svaret på frågan beror vilken typ av dokument som man skall skapa. Är det ett dokument som är unikt för den egna verksamheten kommer man att behöva göra en informationsanalys och skapa sin egen DTD. Är det ett mer generellt dokument, till exempel en enkel rapport eller brev, är det troligaste är att det kommer att finnas ett antal publika DTD:er som man kan använda. Det är också troligt att företag och organisationer går samman och tar fram gemensamma DTD:er som sedan kan anpassas efter behov. Detta har skett för SGML inom flera olika branscher. Så fort man har egna önskemål om hur informationen skall märkas upp kommer man att behöva definiera sin egen DTD eller förändra någon befintlig. I nästa kapitel (3.3.1) beskrivs syntaxen för hur en DTD skall skrivas.

### 3.4.1 DTD-syntax

Innan man definierar sin struktur i en DTD måste man göra ett arbete i form av en dokument/informationsanalys. Man måste ha klart för sig vilka märkord och vilken struktur man behöver för den aktuella dokumenttypen. Hur man går tillväga vid en dokumentanalys är ett område i sig och tas inte upp i denna rapport.

När dokumentanalysen är klar definierar man strukturen i en DTD. Ju mer omfattande och komplex den aktuella informationen är desto större blir behovet av en korrekt informationsanalys och DTD. För att beskriva syntaxreglerna återgår vi dock till vårt tidigare enkla brevxempel. Regeluppsättningen för vår brev-DTD kommer att se ut enligt följande:

```
<!ELEMENT brev (adressat, avsändare, brevstomme) >
<!ELEMENT adressat (namn, gatuadress, postadress) >
<!ELEMENT avsändare (namn, gatuadress, postadress) >
<!ELEMENT brevstomme (inlednfras, stycke+, avslutning) >
<!ELEMENT avslutning (avslutnfras,signatur) >0
<!ELEMENT namn (#PCDATA) >
<!ELEMENT gatuadress (#PCDATA) >
<!ELEMENT postadress (#PCDATA) >
<!ELEMENT inlednfras (#PCDATA) >
<!ELEMENT stycke (#PCDATA) >
<!ELEMENT avslutnfras (#PCDATA) >
<!ELEMENT signatur (#PCDATA) >
```

I DTDn ovan kan man utläsa att dokumenttypen ”brev” måste innehålla elementen adressat, avsändare och brevstomme. Dessa element får bara förekomma en gång och de måste förekomma i nämnd ordning. Adressat och avsändare innehåller båda i sin tur elementen namn, gatuadress och postadress, vilka också måste komma i den ordningen och endast en gång. Namn, gatuadress och postadress är i sin tur deklarerade att innehålla PCDATA, dvs. text. För brevstomme ser vi att stycke är försett med ett plustecken. Det innebär att stycke får förekomma en eller flera gånger.

Följande syntaxregler gäller för att definiera element och dess inbördes ordning:

- Parenteser ( ) omger en sekvens av element eller en lista av alternativ.

- Kommatecken , används i sekvenser för att skilja elementen åt.
- Tecknet | används för att skilja element åt i en lista av alternativ.
- Frågetecknen ? följer efter ett element eller grupp av element och indikerar förekomsten noll eller en gång.
- Tecknet \* följer efter ett element eller grupp av element och indikerar förekomsten noll eller flera gånger.
- Tecknet + följer efter ett element eller grupp av element och indikerar förekomsten en eller flera gånger.
- Om ett element eller grupp av element inte följs av ?, \*, eller + innebär det att det måste förekomma exakt en gång.

Genom att kombinera ovanstående syntaxregler kan man till exempel definiera ett memo. på följande sätt:

```
<!ELEMENT memo (from, to+, date?, (p | note)+)>
```

Detta innebär att vårt memo innehåller ett och endast ett from-element, vilket innebär att bara en person kan stå som författare till memot. Därefter följer ett eller flera to-element, memot kan vara riktat till en eller flera personer. Date-elementet är försett med frågetecknen. Det behöver alltså inte förekomma och om det förekommer får det bara finnas en gång. Avslutningsvis finner vi en gruppering av p- och note-element. Konstruktionen innebär att åtminstone ett p- eller note-element måste förekomma. Både p- och note-elementet får förekomma hur många gånger som helst.

Observera att memo-exemplet ovan inte är färdigdeklarerat. Varje element måste delas upp i ytterligare element tills man kommit ner till teckennivå, dvs. PCDATA. Utöver PCDATA finns det också andra datatyper. Exempelvis kan man välja att låta ett element vara tomt, vilket man gör genom att deklarerat det som EMPTY. Det gör man ofta för t ex. bilder där element får vara tomt men förses med ett attribut som anger vilken bild man vill infoga.

### 3.4.2 Samma information – olika struktur

Vid analys av en dokumenttyp eller en informationstyp kan man beskriva strukturen på en mängd olika sätt. Man kan se det som att informationen har flera olika inneboende strukturer, och man måste försöka finna en, den som bäst uppfyller de krav man har. Nedan följer tre exempel för att exemplifiera att man för samma dokument

kan hitta flera olika strukturer. Vilken man väljer måste vara avhängigt hur informationen kommer att användas. Ingen struktur är i sig bättre än någon annan.

### ***Exempel 1***

Man kan analysera ett dokument och säga att det består av kapitel, avsnitt, underavsnitt, textstycken, uppställningar, bilder och tabeller. Detta ger oss en "logisk layout-orienterad" struktur. Ett annat namn är "dokumentorienterad"

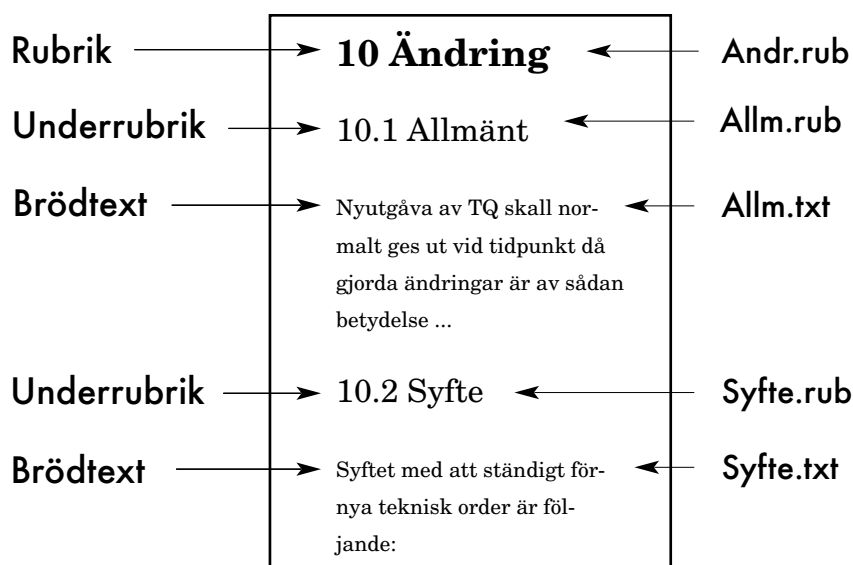
### ***Exempel 2***

Man kan analysera samma dokument och säga att det består av beskrivning, förberedelse, ingredienser, redskap och servering. Vidare kan en förberedelse innehålla ett eller flera steg, etc. Detta ger oss en "innehållsorienterad" struktur. Man försöker beskriva vilken typ av innehåll dokumentet har, dvs. vad det är för typ av information.

### ***Exempel 3***

Man kan analysera dokumentet på ett tredje sätt och säga att det består av olika avsnitt, t ex. inledning, syfte, bakgrund, förberedelse, etc. och att dessa i sin tur består av textstycken, uppställningar, bilder och tabeller. Detta ger oss en blandning av "logisk layout-orienterad" (eller "dokumentorienterad") och "innehållsorienterad" struktur.

Två olika sätt att se på struktur hos samma information är exemplifierat i bilden på sidan 23.

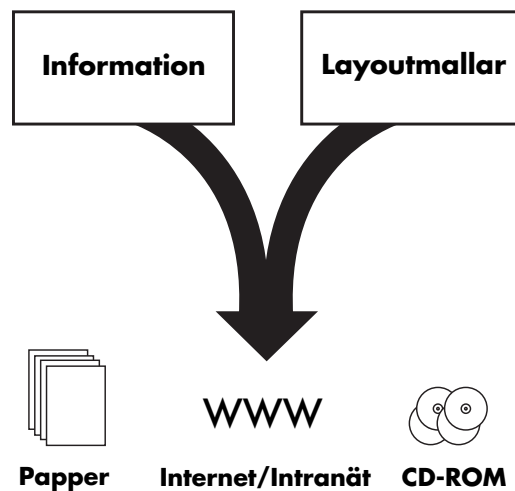


*Två tänkbara sätt att strukturera information.*

### 3.5 Presentation

Som nämndes tidigare finns det i själva XML-filen ingen information om hur innehållet skall presenteras. Information om layouten ligger i stället lagrad i en egen fil, i en så kallad layoutmall (stylesheet).

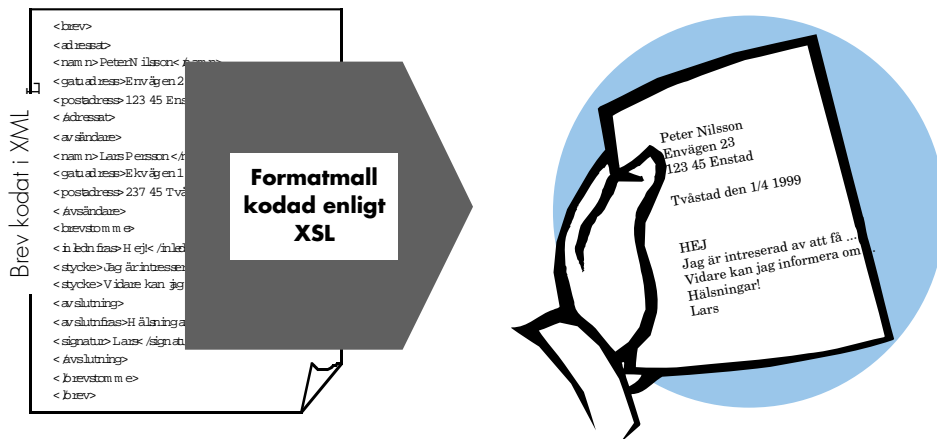
En stor fördel med att särskilja informationen och presentationen av den är att samma information kan presenteras på olika sätt utan att man behöver ändra i XML-dokumentet. Man länkar i stället olika layoutmallar till informationen vid olika presentationstillfällena. Man kan alltså ha en layoutmall för bildskärmspresentation och en för pappersutskrift. Man kan dessutom tänka sig att ha mer än en layoutmall för bildskärmspresentation. Användaren kan då välja mellan olika layouter och välja den som hon anser presenterar informationen på bästa sättet. Ett annat exempel på när det kan vara en fördel att skilja på layout och information är vid så kallad parallell publicering dvs. när man vill publicera samma information på flera olika medier samtidigt som exemplifieras i bilden på sidan 24



*Genom att separera layouten från själva informationen kan man lätt åstadkomma olika layout för olika media.*

Även för layoutmallar pågår arbetet att specificera en syntax för hur uppbyggnaden skall se ut, vilket innebär att även de blir plattformsoch verktygsberoende. W3Cs rekommendation för layoutmallar kallas XSL, eXtensible Style Language. Den kombinerar erfarenheter och egenskaper från ISO-standarderna DSSSL (Document Style Semantics Specification Language) och CSS (Cascading Style Sheets).

Layoutreglerna i XSL skrivs enligt XML-syntax. Således är en layoutmall kodad enligt XSL ett XML-dokument. Med hjälp av reglerna i XSL kan man skapa mycket avancerade layoutmallar. I layoutmallen bestämmer man hur varje informationselement skall presenteras, vilket teckensnitt som skall användas, textens storlek och färg. Man kan också välja att inte visa viss information eller visa den i en annan ordning än den förekommer i själva källdokumentet. Vidare kan man lägga till information eller grafiska objekt, exempelvis kan man lägga till dagens datum eller företagets logotyp. Detta illustreras i bilden på sidan 25.



*Layoutmallen kan ses som ett filter som talar om hur innehållet i märkorden skall presenteras.*

Man kan också använda XSL för att transformera informationen till något annat format. Exempelvis kan man med hjälp av XSL transformera ett XML-dokument till HTML och på så sätt presentera det i en HTML-läsare.

## 3.6 Länkar

### 3.6.1 Utvecklingen av XLL

Att via länkar kunna söka sig fram mellan olika informationsresurser är mycket centralt för användningen och nyttan av webben. Arbetet med att ta fram en ny länkstandard pågår för närvarande inom ramarna för W3C. Den nya länkstandarderna är tänkt att användas tillsammans med XML och har fått namnet eXtensible Linking Language (XLL). Det är egentligen en paraplyterm för två separata standarder som heter XLink och XPointer. Lite förenklat kan man säga att XLink specificerar hur man kan länka sig till en resurs medan XPointer specificerar hur man länkar sig in i en resurs.

De länkar som används i HTML idag s. k. "href-länkar", är ganska primitiva och dessutom svåra att underhålla. XLL erbjuder flera nya

intressanta länkmekanismer. Givetvis kommer den vanliga typen av länk där man klickar på ett markerat objekt och ”hoppas till” målet att stödjas även i XLL, men man får också tillgång till ny funktionalitet. Några exempel på länkar som är möjliga att åstadkomma med XLink och XPointer är:

### 3.6.2 Fördefinierade attribut för XLL-länkar

En länk i HTML kan endast utgå från två specifika märkord, nämligen <a> och <link> (i version 4.0 av HTML-DTDn, och kan till exempel se ut som följande:

```
<A HREF= "måldokument.htm">Den här länken tar dig till måldokumentet</A>
```

Href-attributet är här den viktigaste delen i länken. Det innehåller länkens mål uttryckt som en URL (Uniform Resource Locator).

De enkla href-länkarna som finns i HTML idag kommer att stödjas även i XLL men de har en lite annorlunda syntax. Eftersom man själv definierar sina märkord i XML finns det inte något bestämt element som länkar utgår ifrån som i HTML. I XML definierar man att ett visst element kan innehålla en länk genom att definiera det speciella attributet ”xlink”. Detta attribut definierar inte bara att elementet det sitter på kan innehålla en länk utan också vilken typ av länk det är fråga om. Se exempel:

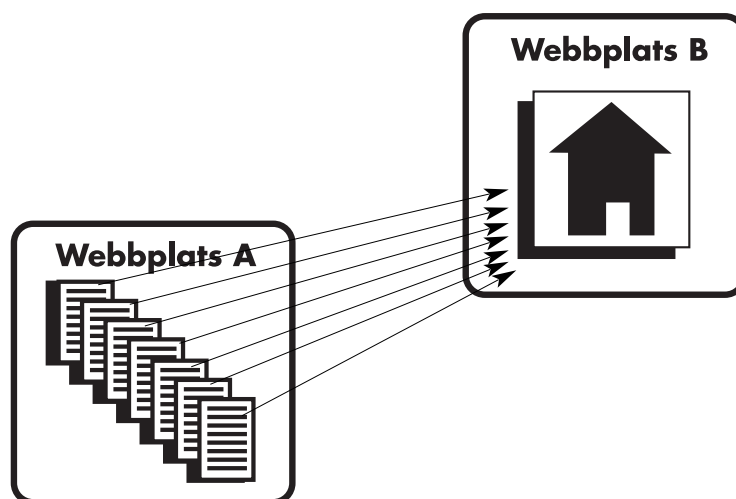
```
<förening  
xlink:form="simple"  
href= "http://www.sgmlförening.se/dokument.xml">  
SGML Användarförening  
</förening>
```

I XLL har man också möjlighet att specificera vad som skall hända när man klickar på en länk. Attributet ”show” indikerar om målet som länken pekar på ska ”bäddas in” (embed) i det nuvarande innehållet, ersätta det (replace) eller visas i ett nytt fönster (new). Attributet ”actuate” styr huruvida användaren bestämmer när länken ska följas eller om det sker automatiskt så fort ett dokument öppnats. Ett exempel på användningen av det senare kan vara när man har en startsida som endast består av länkar till andra dokument,

kanske från olika webbplatser. När användaren öppnar startsidan infogas de refererade dokumenten automatiskt.

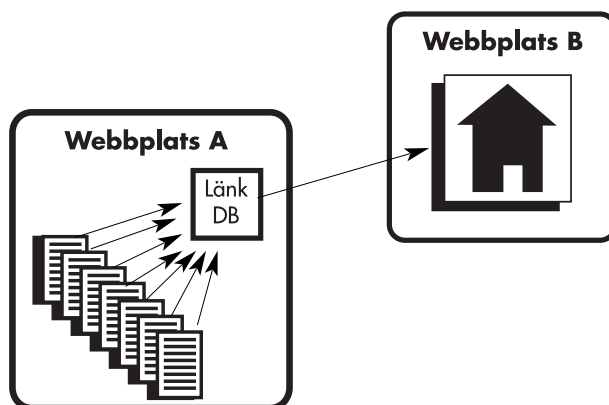
### 3.6.3 Enkla och utökade länkar

Den typ av länkar som finns i HTML (href-länkar) är ett exempel på vad man i XML kallar enkla länkar. En enkel länk har sin källa i ett XML-dokument och pekar på ett enda mål. Utökade länkar är som namnet antyder mer avancerade. En utökad länk kan till exempel peka flera mål samtidigt. Man har också möjlighet att lägga länkarna utanför de dokument de länkar till/från i separata filer eller länkdata-baser.



Ett exempel på vad man kan ha för nytta av en länkdata-bas är följande: Antag att man från sin webbplats vill länka sig till samma mål från tio olika ställen, se bild nedan.  
*Från en och samma webbplats finns det ofta många länkar från olika källor som går till samma mål.*

Om webbplats B av någon anledning tvingas ändra URL (Uniform Resource Locator) så bryts alla länkarna som går från dokumenten på Webbplats A. För att återskapa länkarna måste man gå in i varje dokument på webbplats A som länkar till webbplats B och uppdatera



länkarna. Ett problem i sammanhanget är naturligtvis att hitta alla länkarna som går till webbplats B. En lösning på problemet skulle kunna vara att hantera länkarna i en länkdatabas, se bild nedan. *Genom att samla alla länkar i en länkdatabas kan man förenkla underhållet av dem.*

Att organisera länkarna på detta sätt är möjligt i XLL eftersom länkar kan separeras från själva källdokumentet.

### 3.6.4 Länkar inuti dokument

XPointer gör det möjligt att länka till speciella platser inuti dokument. Att med hjälp av XPointer skapa en länk till t ex. en viss rubrik kan ske antingen genom att man pekar ut rubriken med hjälp av ett ID eller att man "vandrar" ner till rubriken genom dokumentstrukturen. Ett exempel på användning av det sist nämnda kan vara när man vill bygga upp en dynamisk innehållsförteckning. Man kan då exempelvis länka sig till den första rubriken i varje dokument. Om man använder denna konstruktion gör det inget om rubrikens ID ändras.

XLink och XPointer kan mycket väl kombineras. Ett exempel på hur en länk till roten av ett dokument skulle kunna se ut är följande:

```
<xlink:form="simple" href="http://www.admin.kth.se/SGML/dokument.xml#root()">
```

Med hjälp av XPointer kan man sedan fortsätta och klättra nedåt i dokumentetstrukturen.

### 3.7 Verktyg

För att vi skall kunna skapa, administrera och presentera information med hjälp av XML, XSL och XLL måste vi ha verktyg som stödjer standarderna. Ordet stödja är lite diffust och det finns heller inga entydliga definitioner av vad en programvara skall klara för att kunna kallas ett XML-verktyg. Det finns lite olika kategorier av XML-verktyg, gränserna är dock inte glasklara, men i första hand behöver vi följande verktyg för att använda XML:

- Verktyg för editering. För att skapa XML-kodad information behöver man en XML-editor. Tekniskt sett är det fullt möjligt att skriva ett XML-dokument i en vanlig texteditor men i praktiken blir det mycket arbetsamt att själv infoga alla märkord. Editorn ger författaren stöd bl. a. genom att tolka strukturen i DTDn och ge förslag på element som för närvarande är möjliga att infoga.
- Verktyg för lagring och administration. Har man stora informationsmängder kommer man att behöva en programvara som hjälper till att administrera informationen. Beroende på vilka krav man har kan man tänka sig olika typer av administrationsverktyg. Från enkla som endast håller ordning på informationen på dokumentnivå till avancerade databaser där man t ex. bryter ner dokument i mindre komponenter och kan versionshantera ner på elementnivå.
- Verktyg för presentation. För att presentera XML-kodad information behöver vi en webbläsare som kan tolka själva XML-dokumentet och layoutmallen. För att kunna skapa mer avancerade presentationer måste XML-läsaren också kunna tolka strukturen i DTDn.

Förutom ovanstående verktyg för att använda XML behöver vi också verktyg för att utveckla XML-system, denna verktygskategori kan vidare delas in i följande kategorier:

- Verktyg för design av DTD:er.

- Parsers, d.v.s. program som har till uppgift att kontrollera att XML-dokument följer struktur och syntax.
- Verktyg för design av layoutmallar.
- Verktyg för att konvertera XML-dokument mellan olika strukturer.

För mer information om XML-verktyg se Statskontorets rapport (1999:16) XML-verktyg.

## 4 XML OCH TILLÄMPNINGAR

### 4.1 Tillämpningsområden

Intresset för XML är mycket stort. På webben och i tidningar kan man läsa att XML kommer att medföra en revolution på Internet, att det är en plattform för handel över nätet, ett koncept för dokumenthantering, ett format för databaskommunikation, storskalig webbpublicering, intranät, extranät och dataöverföring i allmänhet. Det används på olika sätt i flera viktiga kommersiella programvaror, t ex. Office 2000, SAP R3 och Oracle 8. Vissa menar att det kommer att ersätta HTML på Internet. Strateger på ledande IT-företag som Microsoft, Netscape, Oracle och Sun är alla lyriska över de nya möjligheterna som erbjuds.

Eftersom XML är mycket flexibelt kan man tänka sig många olika tillämpningar inom ett mycket stort spann. XML kan naturligtvis användas för presentation av information på samma sätt som HTML används idag men också till mycket mer. XML skapar förutsättningar för att utveckla tillämpningar som bearbetar strukturerad information på många olika sätt. Vilka tillämpningar kan man då tänka sig för XML?

De applikationer som framförallt kommer att driva XMLs fortsatta utveckling är sannolikt de som inte kan åstadkommas på grund av begränsningarna i HTML. Exempel på sådana applikationer är:

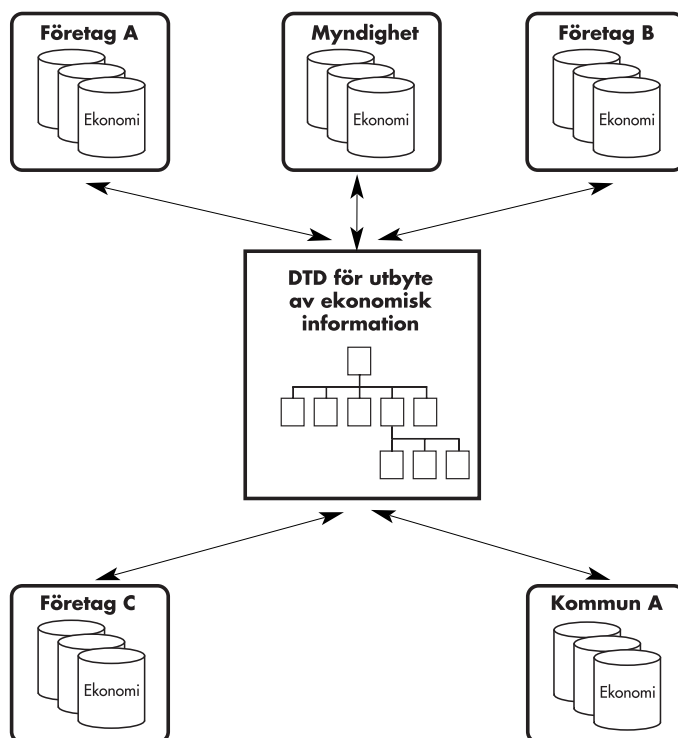
- När man vill hantera information på ett strukturerat och plattformsoberoende sätt under hela skapandeprocessen. Ett exempel på detta kan vara att ett företag hanterar produktinformation i XML internt och gör den tillgänglig för kunderna i ett extranät. Vem som får ta del av vilken information styrs av metadata (i form av t ex. XML-attribut) som knyts till informationselementen.
- Applikationer där man vill att samma informationsmängd skall kunna presenteras på olika sätt för användaren. Ett exempel på detta är att hantera patientjournaler i XML. Genom att förse de olika informationselementen med information om vem som skall ha åtkomst till dem kan man ge olika kategorier av användare tillgång till olika delar av informationen. Patienten och läkaren kanske har tillgång till hela journalen medan forskaren har tillgång till avpersonifierade diagnoser och ordinationer.

- Applikationer i form av ”intelligenta webb-agenter” som söker upp och skräddarsyr information till användaren. Idag bygger i princip all sökning på webben på fritextsökning, vilket innebär att man kan få väldigt många träffar om väldigt många olika saker som man inte är intresserad av. Genom att strukturera information och förse den med information om informationen (så kallad metadata) kan man göra betydligt effektivare sökningar än genom enkel fritext.
- Vid storskalig webbpublicering när man vill strukturera informationen och förse den med metadata som gör den lättare att hitta.
- När man vill publicera samma information på flera olika medier. Om man till exempel vill publicera samma information på CD-ROM, Internet, intranet, papper och som SMS-meddelande på en mobiltelefon kan man med fördel hantera den i XML och använda olika layoutmallar för de olika medierna.
- Applikationer för elektronisk handel. XML kan fungera som en syntax för att överföra strukturerade meddelanden (jfr. EDI). På så sätt kan man dra nytta av kommersiella programvaror som stödjer XML.
- När man vill säkerställa att dokument följer en given struktur. Om man till exempel vill förvissa sig om att alla beslut, protokoll, journaler, säkerhetsföreskrifter, kvalitetsdokument och fakturor ser ut på föreskrivet sätt kan man skapa DTD:er för dem och koda dem i XML.
- När man vill bygga ”intelligenta informationslösningar”. Exempel på det kan vara avancerade interaktiva blanketter eller elektroniska manualer eller när man vill använda ”pushteknik” för att skicka information till användarna.

## 4.2 XML som universellt dataformat

Hittills har vi främst behandlat XML med utgångspunkt i att det är ett språk för att strukturera dokument och information som skall publiceras på webben eller andra media. Eftersom XML är standardiserat och flexibelt är det också mycket väl lämpat som ett format för dataöverföring i allmänhet. Man kan t. ex. använda XML för adresslistor, fakturor, personalregister, prislister, ordlistor etc. Man kan också använda XML som ett överföringsformat när man vill

flytta information mellan olika typer av ekonomisystem. När man vill överföra data mellan olika databaser använder man idag ofta vanliga textfiler som mellanlagringsformat. Nackdelen med det är att det är svårt att säkerställa att textfilen följer en viss struktur.



*XML kan utgöra ett generellt standardiserat format för överföring av data mellan databaser med sinsemellan olika struktur.*

#### 4.2.1 XML och EDI

Electronic Data Interchange (EDI) innebär att man definierar en samling standardiserade meddelanden så att företag och organisationer kan utbyta information via elektroniska meddelanden. EDI förknippas med elektronisk handel som innebär att två eller flera organisationer kommer överens om att utbyta affärsrelaterad information på ett standardiserat sätt. Affärstransaktioner som t ex. beställning, orderbekräftelse, betalning o.s.v. kan överföras direkt

mellan olika organisationers ekonomisystem utan någon manuell hantering.

Ett problem med EDI har varit att definiera standarder för hur informationen skall struktureras. En sådan standard är EDIFACT som specificerar ett antal bestämda fält som måste förekomma i en viss följd. Idag måste ofta de företag som vill etablera en EDI-förbindelse kontakta varandra i förväg samt göra komplexa installationer av speciell programvara för att konvertera de elektroniska meddelandena och översätta dem till ett format som kan tolkas av de interna systemen. Detta är naturligtvis ett både kostsamt och statiskt arbetssätt.

”The XML/EDI group” menar att XML är mycket väl lämpat för att använda som format för EDI. XML är ett publikt format som sannolikt kommer att stödjas av de flesta webbläsare under lång tid. Traditionellt sett har företag och organisationer fått bygga upp sina EDI-system från grunden. Det är ingen hållbar strategi idag. Man måste kunna utnyttja standardprodukter för att på så sätt kunna dra nytta av andras utveckling och dela på kostnaderna.

#### 4.2.2 XML och Java

XML skapar förutsättningar för bearbeta information på nya intressanta sätt. Ett intressant område är t ex. utvecklingen av Java som är speciellt framtaget för att användas i samband med Internet och intranet. Java är ett programmeringsspråk som är utvecklat av Sun Microsystem. Java skall enligt intentionen vara plattformsoberoende, det är nämligen skrivet för en s.k. ”virtuell maskin” som kan emuleras av alla javainstallationer. XML skapar nya förutsättningar för att utveckla Java-program som på olika sätt bearbetar den strukturerade informationen. En HTML/XML-sida kan innehålla Javakod som exekveras i webbläsaren.

Antag t ex. att man vill skriva ett program som interaktivt kan visa respektive dölja olika uppgifter om personer på en adresslista. Antag vidare att man i ett visst sammanhang endast vill presentera efternamn och telefonnummer till de personer som finns med på listan. Detta är svårt om all information om en person lagras i en enda post, t ex.:

<P>Kalle Svensson Storgatan 32 08-102 030 40</P>

Programmet kan då inte med säkerhet avgöra vad som är förnamn, efternamn, gatunamn o.s.v. Det vore naturligtvis en stor fördel om dessa uppgifter i stället hade lagras åtskilda med väldefinierade märkord enligt nedanstående exempel:

```
<ADRESS>
<FORNAMN>Kalle</FORNAMN>
<EFTERNAMN>Svensson</EFTERNAMN>
<GATA>Storgatan</GATA>
<NR>32</NR>
<RIKTNUMMER>08</RIKTNUMMER>
<TFNNUMMER>102 030 40</TFNNUMMER>
</ADRESS>
```

XML tillåter användaren att skapa sina egna märkord vilket i sin tur innebär att det blir möjligt att strukturera informationen på ett sätt som är mer lämpat för automatisk behandling. Jon Bosak på Sun Microsystems menar att "XML ger Java något att göra".

Java används idag bl.a. för att göra HTML-sidor mer levande och interaktiva, t. ex. genom dialogrutor, texter som blinkar, bilder som snurrar o.s.v. Eftersom Java kan distribueras över webben tillsammans med en HTML/XML-sida så finns det stora möjligheter att skapa intressanta interaktiva webblösningar. Java kan t. ex. användas för att skapa expanderbara innehållsförteckningar, interaktiva beställningsverktyg, sökfunktioner, funktioner för användaranpassad presentation etc. Java kan mycket väl användas för applikationer utanför webben.

### 4.3 Metainformation

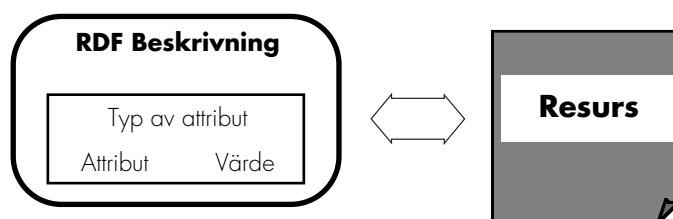
Information om själva informationen, det vill säga metadata, är mycket viktigt för att kunna hitta den information man söker. Exempel på metadata till en artikel är till exempel titel, författare, skapandedatum, ansvarig utgivare, version och nyckelord. Om man i ett arkiv förser alla arkiverade artiklar med sådan uppgifter blir det betydligt enklare att hitta just den artikel du söker.

Sökning är ett stort problem på webben idag. De flesta som använt Internet har erfarenhet av hur svårt det kan vara att hitta just den information man letar efter. Om jag till exempel vill ha tag i en artikel som jag vet är skriven av Maria Johansson kan jag göra en sök-

ning på "Maria Johansson" i en sökmotor på webben. Problemet är då att jag får en mängd träffar där namnet Maria Johansson nämns. Det finns idag inget sätt för mig att tala om att jag bara vill ha träffar på dokument som är skrivna av Maria Johansson.

För att råda bot på det här problemet har W3C tagit fram Resource Description Framework (RDF). Det är en syntax, som bygger på XML, för att förse "resurser" med beskrivande metadata. En resurs avser i det här fallet information som är elektronisk tillgänglig, till exempel webbplatser, dokument, hemsidor, elektroniska arkiv etc.

Enkelt uttryckt kan man säga att RDF bygger på tre delar, de resurser som beskrivs, attributen som beskriver och värden på dessa attribut. Med hjälp av RDF kan man beskriva nästan vad som helst så länge de har en unik webbadress (URI - Uniform Resource Identifier).



*I RDF beskriver man resurser med hjälp av attributvärden.*

Så här kan det se ut om vi beskriver den här skriften i RDF:

```
<Description about = " http://www.statskontoret.se/publ/stk-pub1.htm#986" >  
<Title>Vad är XML?</Title>  
<Creator> <Seq>  
<li>Jan Lundh</li>  
<li>Daniel Björkman</li>  
<li>Marie Österberg</li></Seq>  
</Creator>  
<Date> 1999-03-15</Date>
```

<Subject>**XML, SGML, metadata, standarder**</Subject>  
</Description>

Målet med RDF är att den skall utgöra ett standardiserat sätt att beskriva resurser på webben. RDF specificerar inte vilka attribut som skall användas. Tanken är att man själv skall välja attribut som är meningsfulla för den information man vill beskriva. Det kan dock vara en fördel att, om möjligt, använda någon erkänd publik attributsamling, som t. ex. Dublin Core (för mer information om Dublin Core se, <http://purl.oclc.org/dc/>), eftersom det skapar likformighet vilket i sin tur gör det lättare att hitta informationen.

#### 4.4 Exempel

Ett tänkbart *scenario* över hur XML skulle kunna användas är följande:

*XML Soft* är en påhittad organisation i IT-branschen som arbetar med att skapa effektiva tillämpningar för XML. Systemerna och programmerarna på XML Soft har gedigen erfarenhet av XML och har arbetat med denna standard sedan 1996. All försäljning av programvaror och den största delen av kundkontakterna sker idag via webben. Likaså sker alla beställningar, eventuella betalningar, leverans av program och dokumentation via webben.

När en kund väljer att göra en beställning av en programvara lot-sas han eller hon igenom hela beställningsprocessen av en försäljnings/distributionsagent som XML Soft har utvecklat i program-språket Java. Försäljnings/distributionsagenten ställer ett antal frågor till kunden efter en fördefinierad XML-mall (DTD). Allt eftersom kunden svarar på frågor om operativsystem, webb-läsare, adressinformation m.m. lägger försäljnings/distributionsagenten in informationen i en XML-baserad kunddatabas. När själva beställningen har avslutats och kunden har verifierat att den är korrekt får kunden en elektronisk bekräftelse, samtidigt skickas också en elektronisk faktura till den bank som kunden har specificerat. Efter-som fakturan skickas enligt den standardiserade XML DTDn, invoice2000, kan den expedieras av banken helt automatiskt.

Efter avslutad beställning får kunden åtkomst till utbildnings-agenten som också är ett javabaserat program som hanterar all användardokumentation till det program som användaren just köpt.

Utbildningsagenten behöver inte laddas ner till användarens lokala dator utan kan köras direkt över webben. Utbildningsagenten hämtar information från kunddatabasen och ber dessutom användaren att komplettera informationen med upplysningar om utbildningsnivå, intresse och ett antal andra uppgifter som sedan styr hur utbildningsagenten presenterar information för användaren. Naturligtvis kan användaren själv välja om han eller hon vill söka efter information eller få den presenterad enligt ett viss upplägg. Presentationen stöds av ljud och video när användaren så önskar och när detta är lämpligt.

Givet att säkerheten hanteras på lämpligt sätt är scenariot ovan på intet sätt orälistiskt och fördelarna för kunden är uppenbara:

- Kunden får hjälp att välja rätt program till rätt systemmiljö genom beställningsagentens försorg.
- Kunden behöver inte själv ombesörja eventuell betalning eftersom detta sker helt automatiskt som en standardiserad tjänst mellan XML Soft och banken.
- Kunden kan direkt ta del av dokumentationen för programmet. Dokumentationen presenteras efter kundens förutsättningar och önskemål genom utbildningsagentens försorg.

Dessa fördelar är naturligtvis inte endast XMLs förtjänst. Utan Java eller något annat programspråk som fungerar tillsammans med webbläsaren skulle scenariot vara otänkbart. XML är dock en mycket viktig komponent i sammanhanget. Det hade varit mycket svårt att skapa dessa applikationer i HTML. För att kunna skapa beställnings- och utbildningsagenterna måste man veta hur data är strukturerad. Det räcker inte att veta hur det är tänkt att data skall vara strukturerad, man måste kunna försäkra sig om att den verkligen är strukturerad på det avsedda sättet. Det är inte heller speciellt troligt att vi får se en standardiserad elektronisk faktura i HTML men det kommer vi med säkerhet få göra i XML.

Förutom de uppenbara fördelarna för kunden så åtnjuter också XML Soft en mängd fördelar genom att hantera sin information i XML.

All information som XML Soft hanterar lagras och administreras i en stor XML-databas. Informationen finns lagrad på endast ett ställe och kan återanvändas eftersom informationen är väl strukturerad i moduler. När man skall ta fram en manual till ett nytt pro-

gram använder man en XML-editor och manual-DTDn som ger användaren stöd genom att tala om vilka märkord (element) som är tillåtna. XML-editorn varnar användaren om han eller hon bryter mot strukturen.

Att utveckla beställnings- och utbildningsagenten var inte speciellt svårt eftersom XML Soft arbetar med strukturerad information, specificerad i olika DTD:er. När man väl har gjort en korrekt informationsanalys och strukturerat sin information på ett effektivt sätt är det inte speciellt svårt att presentera informationen på olika sätt.

Eftersom utbildningsagenten jobbar mot informationen direkt via webben så har kunden alltid tillgång till den senaste informationen. Om XML Soft upptäcker något fel i informationen så är det bara att ändra på ett ställe och därefter har användaren direkt tillgång till den uppdaterade informationen.

Om XML Soft vill byta XML-databas eller operativsystem så är det inget problem eftersom XML är plattformsoberoende och standardiserat.

Detta scenario tar endast upp några exempel på vilka fördelar XML skulle kunna ge i en organisations informationshantering. Naturligtvis är det dock så att XML i sig självt inte skapar några direkta fördelar. XML ger dock en stabil grund, väldigt goda förutsättningar för effektivt informationsutbyte, möjlighet till att definiera väl strukturerad och korrekt information, möjlighet att skapa intelligenta informationslösningar m.m.

## 5 BEDÖMNING AV FRAMTIDSUTSIKTER

XML som blev W3C rekommendation i februari 1998 har fått ett mycket varmt välkomnande och väckt stort intresse inom en rad områden. XML kommer med största sannolikhet få en viktig roll både för webbens framtida utveckling och för hur man hanterar information internt inom organisationer. Det finns alltså skäl att omgående börja bygga upp kunskap kring tekniken och hur den kan användas.

De flesta vill numera inte bara använda webben för att publicera information utan även för verksamhetskritiska tillämpningar, utbyte av information, elektronisk handel och mycket mer. Dessa typer av tillämpningar kräver ett format som kan hantera strukturerad information. Efterfrågan på nya avancerade tillämpningar på webben kommer att bidra till en fortsatt snabb utveckling av XML och verktyg för XML. Den stora uppståndelsen och det faktum att W3C:s ca 250 medlemsorganisationer alla är positiva till XML talar också för att det stöd som behövs för en snabb utveckling finns tillgängligt.

När man tar del av det som skrivs om XML slås man av de höga förväntningarna och de genomgående positiva inställningen till XML och till dess framtida användning. Finns det då inga problem?

Att utveckla XML-verktyg som stödjer själva XML kommer förmodligen inte att stöta på så stora hinder. Att utveckla verktyg som stödjer XLL som ju bygger bland annat på HyTime och TEI kan komma utgöra ett större problem. Trots att HyTime har funnits i fem år är det mycket få SGML-verktyg som stödjer HyTime. Inte heller lär XSL i sin helhet bli lätt att implementera. Inledningsvis finns dock möjligheter att använda Cascading Style Sheet för layout.

De företag och organisationer som idag arbetar med att utveckla verktyg för SGML kommer naturligtvis att ha relativt lätt att anpassa sina tillämpningar så att de också fungerar med XML och ytterst drastiska prissänkningar är att vänta.

Många företag och organisationer i Sverige och utomlands använder redan idag SGML på ett mycket effektivt sätt. Stora summor pengar har investerats inom området och erfarenhet finns sedan snart ett decennium. Exempel på svenska företag och organisationer som arbetar med SGML idag är Astra, Ericsson, Försvaret, Hägglunds Vehicle, Saab, Volvo, Kockums, Bofors och SAS.

Det bedrivs idag en rad projekt, både inom det privata näringslivet och inom den offentliga sektorn, där man analyserar tillväga-

gångssätt, bygger prototyper och implementerar systemlösningar som på olika sätt drar nytta av de fördelar som XML ger. Ett exempel på detta är Regeringskansliet som från regeringen fått i uppdrag att lämna förslag på ett nytt offentligt rättsinformationssystem. Arbetsgruppen skriver ”Mycket tyder på att SGML eller XML eller båda är framtidens märkningsspråk för Internet. Användningen av något av dessa märkningsspråk är såvitt kan bedömas i dag en förutsättning för att kravbeskrivningens långsiktiga mål skall kunna uppnås.” (Ett offentligt rättsinformationssystem, Ds 1998:10)

## 6 REFERENSER

### 6.1 WWW-referenser

***Arbortext***

[http://www.arbortext.com/Think\\_Tank/XML\\_Resources/xml\\_resources.html](http://www.arbortext.com/Think_Tank/XML_Resources/xml_resources.html)

***Architag***

<http://architag.com/xmlu/>

***Cafe con Leche***

<http://metalab.unc.edu/xml/>

***Dublin Core***

<http://purl.oclc.org/dc/>

***Microsoft XML***

<http://msdn.microsoft.com/xml>

***Schmanet***

<http://www.schema.net/web/>

***SGML användarföreningen i Sverige.***

<http://www.admin.kth.se/SGML/>

***Webtools***

<http://www.webtools.com/>

***World Wide Web Consortium (W3C)***

<http://www.w3.org/>

***XML.com***

<http://www.xml.com/xml/pub>

***XML/EDI group***

<http://www.geocities.com/WallStreet/Floor/5815/index.html#Architecture>

***Xmlsoftware.com***

<http://www.xmlsoftware.com/>

### 6.2 Litteratur

Connolly Dan (Guest Editor), *XML: Principles, Tools, and Techniques*, World Wide Web Journal: Volume 2, Issue 4, 1997.

Harold Elliott Rusty, *Structuring Complex Content for the Web, XML*, IDG Books Worldwide, 1998.

Jelliffe Rick, *The XML&SGML cookbook, Recipes for structured information*, Prentice Hall, 1998.

Light Richard, *Presenting XML*, SamsNet, 1997.

### 6.3 Övriga publikationer

Regeringskansliet, *Ett offentligt rättssystem*, Ds 1998:10.

## 7 ORDFÖRKLARINGAR

- DTD** Dokument Typs Definition, uppsättning regler i SGML/XML som talar om vilka märk-ord som får förekomma och i vilken ordning de får förekomma.
- Element** Informationsmängd som definieras av ett start- och ett slutmärkord.
- HTML** HyperText Markup Language, det format som i huvudsak används på webben idag.
- Java** Objektorienterat programmeringsspråk som är utvecklat av Sun Microsystems.
- Märkord** Ord inom "<>" som används för att beskriva en viss informationsmängds innehåll eller presentation.
- Parser** Program för att bearbeta och validera SGML/XML-information.
- Plugin** Tillägsprogram till webb-läsare som gör det möjligt att läsa/bearbeta andra format än HTML, t ex. PDF.
- SGML** Standard Generalized Markup Language, internationell ISO-standard som gett upphov till HTML och XML.
- Stylesheet** En uppsättning regler som enligt en viss syntax bestämmer hur olika element (informationer inom ett märkord) skall presenteras.
- W3C** World Wide Web Consortium, ca 250 medlemsorganisationer, arbetar med att ta fram gemensamma protokoll för webben.

## 8     STATSKONTORETS PUBLIKATIONER FÖR 1998/1999

1998

- 98:1     Statskontorets publikationer utgivna under åren 1995-1997. GRATIS
- 98:2     Offentlig förvaltning och demokrati i informationssamhället. 150:-
- 98:3     Ramavtal om Programvaror och Tjänster. GRATIS
- 98:4     Upphandling - Abonnentväxlar. 100:-
- 98:5     Myndigheterna och Informationssystemen inför år 2000 - delrapport. GRATIS
- 98:6     Vad är XML? Reviderad upplaga (1999). 125:-
- 98:7     Utgått
- 98:8     Statens tillsyn över verksamhet i landsting och kommuner. GRATIS
- 98:9     Skiftet till år 2000-läget i myndigheter och samhällsfunktioner: lägesrapport 3. 100:-
- 98:9A    Coping with the year 2000. GRATIS
- 98:10    Analysmodell för Anpassning av IT-system till Euro. GRATIS
- 98:11A   Förvaltningspolitik för tillväxt. 175:-
- 98:11B   Konkurrenskraft - Genomgång av internationella rapporter. 50 kr. Paketpris A-B 200 kr.
- 98:12    Privatfinansiering genom partnerskap. 150:-
- 98:13    Kostnader till följd av personuppgiftslagen. GRATIS
- 98:14    EU-bedrägeridelegationen. GRATIS
- 98:15    Staten i omvandling 1998. 175:-


- 98:16 Säker e-post med S/MIME. Del IX. 60:-
- 98:17 Skiftet till År 2000 läget i myndigheter. GRATIS
- 98:18 Sammanhållen strategi för samhällets IT-säkerhet.  
GRATIS
- 98:19 Länsstyrelsen - gårdagens eller morgondagens förvaltning?  
GRATIS
- 98:20 Kunskapslyftet som modell och metod - adhocрати eller  
byråkrati? GRATIS
- 98:21 IT-kompetens för offentlig sektor. 75:-
- 98:22 Samordning mot droger - kartläggning och skiss till ny  
myndighetsstruktur på alkohol- och narkotika-  
området. 75 :-
- 98:23 Produktivitetsutveckling i statsförvaltningen 1990-97.  
GRATIS
- 98:24 Kommittéernas prövning av det offentliga åtagandet - hur  
sköter de sig? GRATIS
- 98:25 Smittskyddsarbetets organisation. GRATIS
- 98:26 Skiftet till år 2000 — Läget i myndigheter. GRATIS
- 98:27 EG-regler och svensk regelkvalitet. GRATIS
- 98:28 Gamla län blir nya regioner? Lägesrapport. GRATIS
- 98:29 Läget i 10 samhällsfunktioner — Skiftet till år 2000.  
GRATIS
- 98:30 Svenska EU-programkontoret för utbildning och kompe-  
tensutveckling — En utvärdering. GRATIS
- 98:31 Regeringens styrning och kontroll av den statliga lokalför-  
sörjningen. GRATIS

1999

- 99:1 Statskontorets publikationer utgivna under åren 1996-1998. GRATIS
- 99:2 Skiftet till år 2000 — läget i myndigheter. Lägesrapport 7. GRATIS
- 99:3 E-post i förvaltningen — En rättslig översikt. 75:-
- 99:4 Samordning och styrning av smittskyddsverksamheten. GRATIS
- 99:5 Informationskampanjen Starta eget tillsammans. GRATIS
- 99:6 Utvecklingsgarantin för ungdomar — det första året. GRATIS
- 99:7 Regelförenkling i EU och Sverige. GRATIS
- 99:8 Användning av otraditionella medel i Skåne åren 1998 och 1999. GRATIS
- 99:9 Myndigheternas förutsättningar för deltagande i ett nytt offentligt rättsinformationssystem. GRATIS
- 99:12 Tillstånd för näringsverksamhet. GRATIS
- 99:13 Intensifierat nordiskt samarbete — analys av det svenska styrsystemet. GRATIS
- 99:14 E-post i skolan. 150:-



---



**S**tatskontoret vill rikta uppmärksamheten på XML. Standarden som är utvecklad av W3C erbjuder ett nytt och mer avancerat sätt att definiera information på en webbsida eller i ett dokument. XML erbjuder helt nya möjligheter att strukturera information och bedöms få mycket stor betydelse.

I rapporten ges en kortfattad beskrivning av vad XML är och vilka relationer standarden har till SGML och HTML.

Rapporten riktar sig i första hand till personer inom offentlig sektor som arbetar med att göra information tillgänglig via Internet eller som på annat sätt arbetar med att utbyta information med andra organisationer eller enskilda.

---