XML Kurs for eArkivar

Timeplan

Dette er en 2 dagers kurs i XML og XSD. Kurset er totalt på 11 timer med en repetitsjonstime. Hver time varer ca. 50 minutter og det er lagt opp til 1 times lunsj pause fra 1200 til 1300.

Dag 1 : 11/10/2012 fra 1000-1600

Time	Tema	Beskrivelse
1	Det digitale	Vi begynner med hva som ligger bak ordet ***digital***. Derfra ser vi på utvikling av tegnsett med spesielt fokus på ASCII og UTF
2	Migrasjon og XML	I Norge har vi valgt *migrasjon* som bevaringsstrategi (i motsetning til feks emulering) og vi bruker XML som et verktøy til å realisere denne bevaringsstrategien. I denne timen ser vi på forskjellige måter vi kan lagre data på (fastform, komma separert) og hvorfor bruken av XML er en god måte å løse dette på. Vi ser på hvordan en XML dokument formelt er oppbygget og introduserer element begrepet.
3	Et XML dokument	Nå begynner vi med mer praktisk arbeid og utvikler XML filer i xmlcopyeditor
4		Vi fortsetter med praktisk arbeid og utvider til å inkludere attributter
5	Navnerom og prosessering	Vi går i en retning av mer avansert bruk av XML og ser på prosessering instruksjoner og bruken av navnerom.

Dag 2 (12/10/2012) 0900-1600

Time	Tema	Beskrivelse
1	Gjennomgang	Den første timen er satt av til å gå igjennom alt vi gikk gjennom på dag 1.
2	XML regler og validering	Vi ser på hva det betyr når et XML dokument er korrekt strukturert og kan valideres og viktigheten med dette, spesielt i en arkivsammenheng
3	En XSD fil og hvordan den brukes	Nå utvikler vi en XSD fil og bruker den til å validere en XML fil
4		Vi fortsetter og utvikler flere eksempler av XML / XSD
5	XSD Restriksjoner	Vi utvikler XSD filen med restriksjoner som sier noe om hvilken verdier et element kan inneholde
6	Noark 5 XSD	I den siste timen setter vi sammen alt vi har lært om XML og XSD og utvikler en XML som heter arkivstruktur som vi skal validere opp mot den formelle Noark 5 XSD filen. Det er ingen foilsett for denne timen.

Hvilken kunnskap skal du ha etter kurset:

Tema	Beskrivelse
Generelt	Du skal forstå • hvorfor XML utmerker seg som et uttrekksformat når vi bruker migrasjon som bevaringsstrategi og hvorfor det er bedre enn fastform og komma separert verdier ○ Spesielt viktig med uavhengighet til system/leverandør • skillen mellom data, struktur og presentasjon • at selv om XSLT kan brukes for å prosessere XML og CSS kan brukes til å formatere presentasjonen så brukes de (vanligvis/enda) ikke i depot sammenheng
Tegnsett	Du skal forstå at det er noe som heter 7 og 8 bits ASCII og at vi nå (skal ha) løst bruker tegnsett problematikk med UTF. Bruken av UTF-8 er bakover kompatibel med ASCII.
Tallsystem	På et overordnet nivå forstå det binære tallsystemet og hvordan og hvorfor vi bruker HEX (16 tallsystemet)
XML	Du skal vite følgende om en XML fil: • formelle oppbygging (prolog, dokumentelement osv) • XML elementer og navngiving • XML syntaks regler • XML attributter
Navnerom	Du skal forstå hvilken problemstilling navnerom løser og hvordan det gjør det
XSD	Du skal forstå • hva XSD brukes til og hvorfor det er viktig i en arkiv sammenheng • hva XSD restriksjoner er og hvordan de brukes
XML/XSD	Forskjellen mellom korrekt strukturert (som handler om *syntaks*) og validering (som handler om semantikk)
XSD og Noark 5	Du skal kunne forstå arkivstuktur.xsd og diskutere viderbruk av denne filen

Hvilken ferdigheter skal du ha etter kurset:

Her stiller vi spørsmålet 'Hvordan skal du kunne anvende den kunnskapen du har tilegnet deg?' Gjennom det praktiske arbeidet skal kunnskapen utvikles til noen konkrete ferdigheter.

Tema	Beskrivelse
XML	Du skal være i stand til å • lese og forstå en XML fil • utvikle egne XML filer (ved behov) ○ feks *arkivuttrekk.xml* eller *info.xml* • redigere en XML fil (ved behov) • fikse syntaks feil
XSD	Du skal kunne • utvikle en enkel XSD fil og bruke et program som xmlcopyeditor til å validere en XML fil • utvikle XSD restriksjoner

Husk dette er en introduksjonskurs og tidsrammene begrenser utvikling av ferdigheter gjennom praktisk arbeid. Du blir på ingen måte en mester en XML/XSD mester men vi bygger et fundament som du selv kan bygge videre på.

XML Kurs for eArkivar

Time 1 : Det digitale

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 11/10/2012

Hva det betyr å være digitalt?

Når vi sier at noe ***er digitalt*** så mener vi at vi ***uttrykker en observasjon*** av en virkelig (real world) fenomen *med en (*digital*)* ***bestemt verdi***

Digitalisering av den 'virkelige' verden

- For å få noe fra den virkelige verden inn i en datamaskin brukes det en eller annen form for input enhet

En tastatur genererer et binært tallkode for hver tast som blir trykket

Et kamera 'fanger' lys og konverterer det til binære data ved hjelp av Analog til Digital Konverterer (ADK)

En mikrofon 'fanger' lyd og konverterer det til binære data ved hjelp av Analog til Digital Konverterer (ADK)

Tallsystemer

- ***Desimaltallsystemet*** beskriver et system som har *ti* mulige sifre

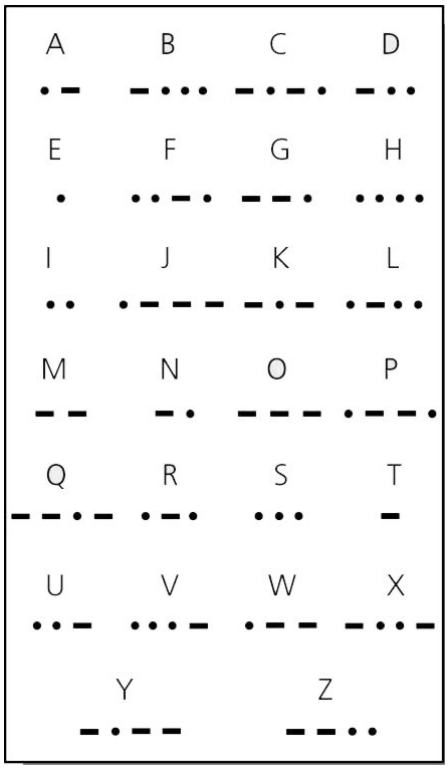
  0, *1*, *2*, *3*, *4*, *5*, *6*, *7*, *8*, *9*

  - Hvert nummer uttrykt i desimalsystemet er en kombinasjon av disse ti sifre
- ***Binærtallsystemet*** beskriver et system som bare har *to* mulige sifre

  0 og *1*

Det binære systemet

- Det binære systemet fungerer i hovedsak på samme måte, bare at den bruker kun ***to*** sifre
  - Hvert nummer uttrykt i det binære systemet er en kombinasjon av disse to sifre
- Binære systemet har vært essensielt i utvikling av IT
  - Elektroniske kretser kan bare ha to mulige tilstander, av eller på
    - Tenk på en lysbryter der bryteren gir deg to valg,
    - lyset er enten ***av*** eller ***på***

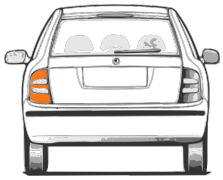


Morsekode

SOS

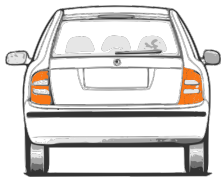
. . . _ _ _ . . .

Jeg kjører rett fram



Jeg kjører til venstre

Jeg kjører til høyre



Skal bare parkere her i to minutter

Hvordan brukes det binære systemet?

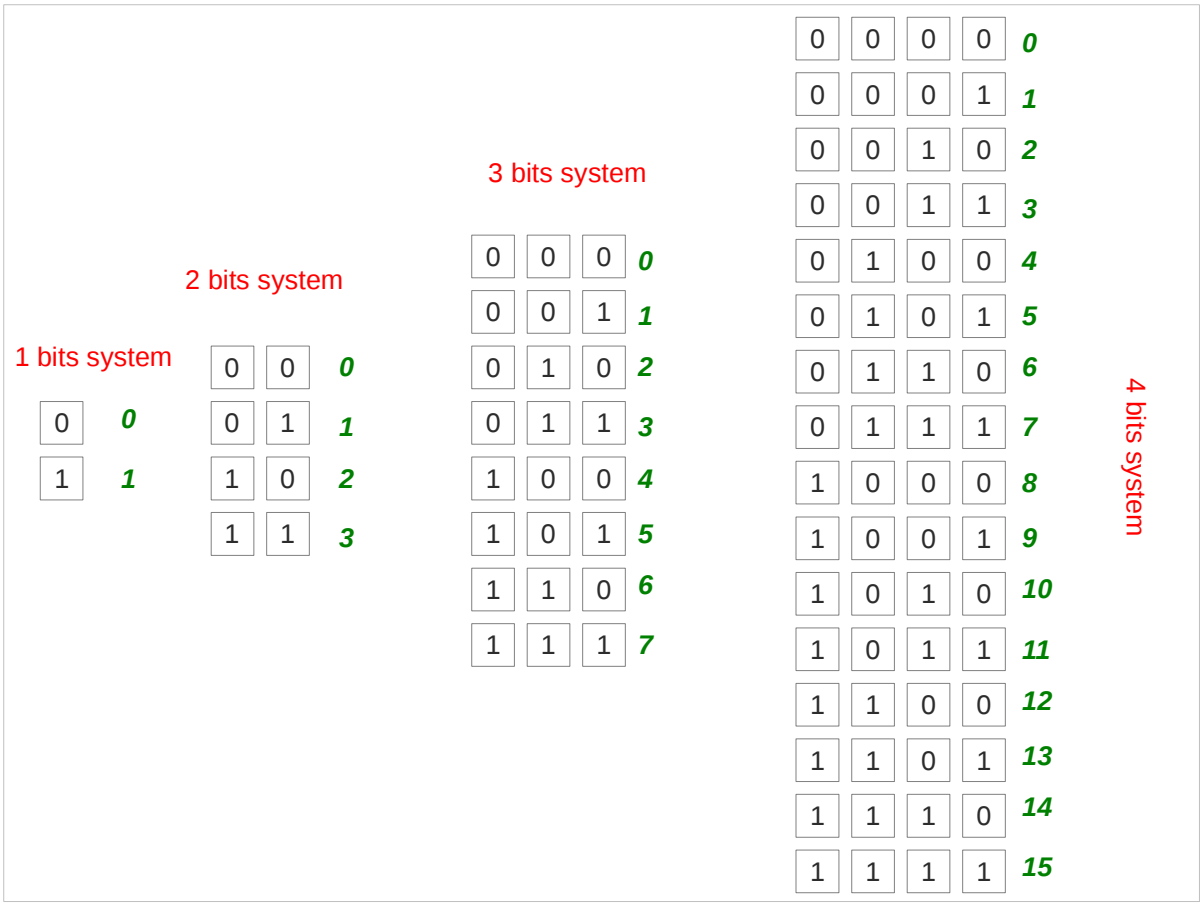
- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$

128	64	32	16	8	4	2	1

Hvordan brukes det binære systemet?

2	1	
0	0	*(2*0) + (1*0) = 0*
0	1	*(2*0) + (1*1) = 1*
1	0	*(2*1) + (1*0) = 2*
1	1	*(2*1) + (1*1) = 3*

4	2	1	
0	0	0	*(4*0) + (2*0) + (1*0) = 0*
0	0	1	*(4*0) + (2*0) + (1*1) = 1*
0	1	0	*(4*0) + (2*1) + (1*0) = 2*
0	1	1	*(4*0) + (2*1) + (1*1) = 3*
1	0	0	*(4*1) + (2*0) + (1*0) = 4*
1	0	1	*(4*1) + (2*0) + (1*1) = 5*
1	1	0	*(4*1) + (2*1) + (1*0) = 6*
1	1	1	*(4*1) + (2*1) + (1*1) = 7*



A,B,C som binære og desimal tal

$$A = 01000001_{bin} = 65_{des}$$

$$B = 01000010_{bin} = 66_{des}$$

$$C = 01000011_{bin} = 67_{des}$$

Det er 10 typer mennesker i verden.

De som forstår det binære systemet
og de som ikke gjør det

Tallsystemer

- Vi er vant til desimaltallsystemet
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Og nå har vi sett på binærtallsystemet
  - 0, 1
- Et annet viktig tallsystem i datamaskiner er sekstentallsystemet
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Dette sekstentallsystemet kalles for det heksadesimale tallsystemet eller HEX
  - Vi representerer ***en*** 8 bits binær nummer med ***to*** HEX tall

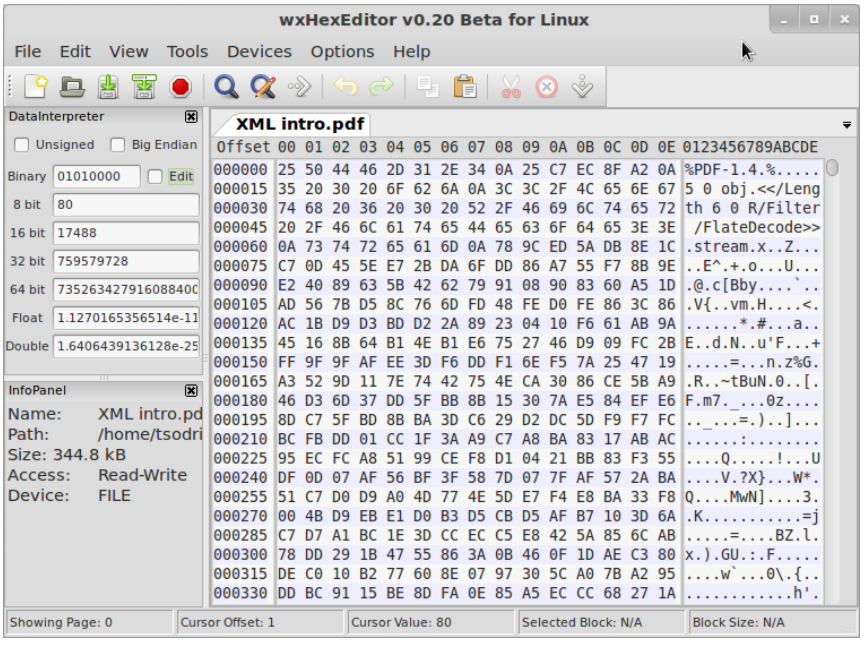
HEX

- Det er tungvint for mennesker å jobbe med binærsystemet 11001111_{bin}
- Det er også tungvint hvis vi hele tiden må konvertere fram og tilbake mellom binær og desimaltall systemer 11001111_{bin} eller 207_{des}
- HEX er en enkel måte å forkorte en 8 bits binærtall til en 2 tegn HEX tall
- Vi representerer en 8 bits binær nummer med to HEX tall
  - Del 8 bits i to

0	0	0	0	0_{des} eller 0_{hex}
0	0	0	1	1_{des} eller 1_{hex}
0	0	1	0	2_{des} eller 2_{hex}
0	0	1	1	3_{des} eller 3_{hex}
0	1	0	0	4_{des} eller 4_{hex}
0	1	0	1	5_{des} eller 5_{hex}
0	1	1	0	6_{des} eller 6_{hex}
0	1	1	1	7_{des} eller 7_{hex}
1	0	0	0	8_{des} eller 8_{hex}
1	0	0	1	9_{des} eller 9_{hex}
1	0	1	0	10_{des} eller A_{hex}
1	0	1	1	11_{des} eller B_{hex}
1	1	0	0	12_{des} eller C_{hex}
1	1	0	1	13_{des} eller D_{hex}
1	1	1	0	14_{des} eller E_{hex}
1	1	1	1	15_{des} eller F_{hex}

Eksempler på HEX

00000101_{bin}	==	005_{des}	==	05_{hex}
00001111_{bin}	==	128_{des}	==	$0F_{hex}$
10001000_{bin}	==	136_{des}	==	88_{hex}
11001111_{bin}	==	128_{des}	==	CF_{hex}
11111111_{bin}	==	256_{des}	==	FF_{hex}



HEX fil eksempel

Tegnsett

- Et tegnsett er det settet av tegn som vi kan både skrive og se på en datamaskin
  - Består av bokstavene, tallene, og kontroll tegn
- Et tegnsett består av en liste assosiasjoner der ***bokstaver og tall blir assosiert med et binært tall*** som en datamaskin kan tolke
  - Vi trenger tegnsett som alle (elektroniske enheter) kan være enig for å kunne utveksle data
    - Bokstaven 'a' er representert med 01100001 (desimaltall 97)
    - Bokstaven 'b' er representert med 01100010 (desimaltall 98)

ASCII

- Utviklet av ANSI (American National Standards Institute)
  - Definert i ANSI dokumentet X3.4-1977
- 7-bits kode, 2^7 = 128 ulike koder (0-127)
- Åttende bit er ubrukt
- To generelle typer koder:
  - 95 av kodene kan vi se på en skjerm
  - 33 av kodene er kontrollkoder
    - har en kontrollfunksjoner på skjerm/printer eller kommunikasjonskanal
- Representerer
  - Latinske alfabetet, arabiske tall, standard tegnsettingstegn
  - Pluss et lite sett med aksenter og andre europeiske spesialtegn

ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

•Hvor er:

- Kontrollkoder? Dec (0-32 + 127)
- Alfanumeriske koder? Dec (65-90, 97-122)
- Numeriske koder? Dec (48-57)
- Tegnsetting koder?

ASCII Kontrollkoder

33 kontrollkoder

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space										
1	1	001	SOH (start of heading)															
2	2	002	STX (start of text)															
3	3	003	ETX (end of text)															
4	4	004	EOT (end of transmission)															
5	5	005	ENQ (enquiry)															
6	6	006	ACK (acknowledge)															
7	7	007	BEL (bell)															
8	8	010	BS (backspace)															
9	9	011	TAB (horizontal tab)															
10	A	012	LF (NL line feed, new line)															
11	B	013	VT (vertical tab)															
12	C	014	FF (NP form feed, new page)															
13	D	015	CR (carriage return)															
14	E	016	SO (shift out)															
15	F	017	SI (shift in)															
16	10	020	DLE (data link escape)															
17	11	021	DC1 (device control 1)															
18	12	022	DC2 (device control 2)															
19	13	023	DC3 (device control 3)															
20	14	024	DC4 (device control 4)															
21	15	025	NAK (negative acknowledge)															
22	16	026	SYN (synchronous idle)															
23	17	027	ETB (end of trans. block)															
24	18	030	CAN (cancel)															
25	19	031	EM (end of medium)															
26	1A	032	SUB (substitute)															
27	1B	033	ESC (escape)															
28	1C	034	FS (file separator)															
29	1D	035	GS (group separator)															
30	1E	036	RS (record separator)															
31	1F	037	US (unit separator)															
														127	7F	177	&#127;	DEL

Source: www.asciitable.com

ASCII Kontrollkoder

- CR (13, ^M) carriage return
- HT (9,^I) horisontal tab
- LF (10, ^J) linefeed
- DEL (127, ALT-127) delete
- FF (12, ^L) form feed (new page)
- BS (8, ^H) backspace
- ESC (27, ^[) escape

ASCII Bokstaver

52 bokstaver, 10 tall

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
									65	41	101	A	A	97	61	141	a	a
									66	42	102	B	B	98	62	142	b	b
									67	43	103	C	C	99	63	143	c	c
									68	44	104	D	D	100	64	144	d	d
									69	45	105	E	E	101	65	145	e	e
									70	46	106	F	F	102	66	146	f	f
									71	47	107	G	G	103	67	147	g	g
									72	48	110	H	H	104	68	150	h	h
									73	49	111	I	I	105	69	151	i	i
									74	4A	112	J	J	106	6A	152	j	j
									75	4B	113	K	K	107	6B	153	k	k
									76	4C	114	L	L	108	6C	154	l	l
									77	4D	115	M	M	109	6D	155	m	m
									78	4E	116	N	N	110	6E	156	n	n
									79	4F	117	O	O	111	6F	157	o	o
				48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
				49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
				50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
				51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
				52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
				53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
				54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
				55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
				56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
				57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
									90	5A	132	Z	Z	122	7A	172	z	z

Source: www.asciitable.com

ASCII Tegnsetting

28 tegnsettingkoder

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
									64	40	100	@	@	96	60	140	`	`
				33	21	041	!	!										
				34	22	042	"	"										
				35	23	043	#	#										
				36	24	044	$	$										
				37	25	045	%	%										
				38	26	046	&	&										
				39	27	047	'	'										
				40	28	050	(	(										
				41	29	051	)	)										
				42	2A	052	*	*										
				43	2B	053	+	+										
				44	2C	054	,	,										
				45	2D	055	-	-										
				46	2E	056	.	.										
				47	2F	057	/	/										
				58	3A	072	:	:										
				59	3B	073	;	;	91	5B	133	[	[					
				60	3C	074	<	<	92	5C	134	\	\					
				61	3D	075	=	=	93	5D	135	]	]					
				62	3E	076	>	>	94	5E	136	^	^					
				63	3F	077	?	?	95	5F	137	_	_					

Source: www.asciitable.com

ASCII

- På grunn av 7-bits begrensningen ble ASCII utvidet til 8-bits, også kjent som Latin-I
  - Dette ga plass til 256 tegn (2^8 = 256) og de fleste europeiske tegn ble inkludert bla æ, ø og å
  - Standardisert i ISO-8859-1
- Men det finnes mange andre tegnsett for forskjellige språk feks, gresk, russisk, japansk, koreansk, kinesisk, samisk
  - For enkelhets skyld så er de første 127 tegn alltid den gamle ASCII standarden
- Men dette har skapt en del problemer spesielt i programerings verden
  - Bla i et populær web applikasjon språk, php

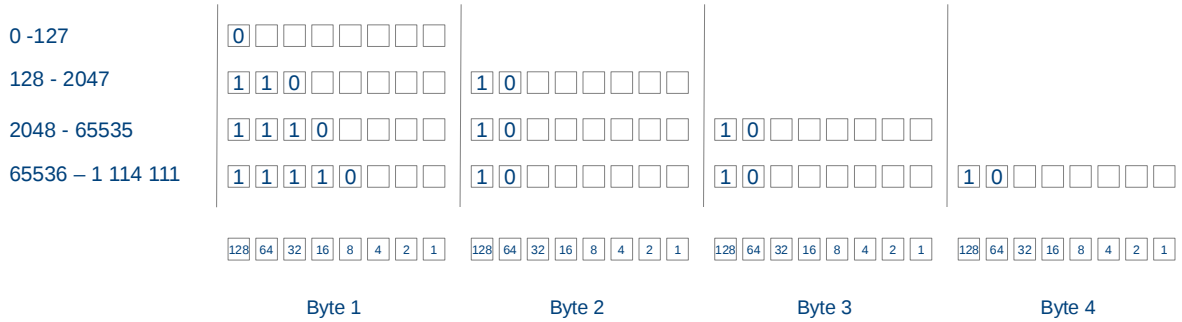
Fra ASCII til Unicode

- Latin-1 er en ISO-standard, men 8-bits som gir plass til 256 tegn er ikke tilstrekkelig for å representere alle mulige tegn i bruk i verden
  - Det har blitt utviklet et tegnsett som heter *Unicode*
- Unicode (Unicode Transformation Format, UTF) har standardisert flere tegnsett for lagring og utveksling av tekstlig data
- Unicode gir et unikt nummer for hvert tegn
  - Uavhengig av plattform
  - Uavhengig av program
- Overgangen fra ASCII til Unicode er ufullstendig

Se også http://www.tamasoft.co.jp/en/general-info/unicode.html

UTF-8

- UTF-8 er et tegnsett som bruker mellom 1 og 4 bytes for å angi et tall til å representere et tegn
- De originale ASCII verdiene lagres i *byte 1* og begynner alltid med 0 (bit 8), dette gjør at UTF-8 er kompatibel med ASCII og mye eksisterende programvare trenger ingen endring for å jobbe med UTF-8



UTF-8

- UTF-8 har variabel tegnlengde
  - Kan føre til nedsatt hastighet i systemer som behandler UTF-8 tekst
    - Du må hele tiden regne ut og avgrense de forskjellige tegn
  - Har også en del å si for filstørrelsen
    - 10,000 tegn * 1 byte (10,000 bytes) bare engelske tegn
    - 10,000 tegn * 2 byte (20,000 bytes) med norske tegn
    - 10,000 tegn * 3 byte (30,000 bytes) med internasjonale tegn
    - 10,000 tegn * 4 byte (40,000 bytes) med internasjonale tegn

Gratis Unicode bok: http://www.unicode.org/versions/Unicode6.0.0/

ISO og Unicode

- ISO og Unicode prøvde å utvikle egne standarder samtidig men ISO ga opp sitt løp og gikk med Unicode
  - Dette er det som heter UCS
- Andre UTF versjoner
  - UTF-16
    - Har variabel tegnlengde (2-4)
    - Bruker minimum 2 bytes, dvs den er ikke bakover kompatibel med ASCII
      - Med bare engelske tegn så vil UTF-8 filer være mindre
    - UTF-16 er nesten det samme som UCS-2
  - UTF-32
    - Fast tegnlengde, fortere å prosessere
    - Bruker minimum 4 bytes
    - Ganske store filer
    - UTF-32 er nesten det samme som UCS-4

UCS= Universal Character Set

ISO-8859-?

- ISO-8859-1 eller *Latin 1 Del 1*
  - Dekker de fleste vest-europeiske språk:
    - dansk*, nederlandsk*, engelsk, færøysk, finsk*, fransk*, tysk, islandsk, irsk, italiensk, norsk, portugisisk, retoromansk, skotsk gælisk, spansk og svensk
- ISO-8859-4 eller *Latin 1 Del 4*
  - Dekker Estisk, latvisk, litauisk, grønlandsk, og samisk
- Avlevering til depot:
  - med bare norske tegn kan bruke Latin 1 Del 1
  - med samiske tegn må bruke Latin 1 Del 4

*mangler noen bokstaver

Latin 1 : Del 1 og Del 4

Table 2 – Code table of Latin alphabet No. 1

b4 b3 b2 b1 \ b8 b7 b6 b5	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0 0 0 0	00		SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð	0
0 0 0 1	01		!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ	1
0 0 1 0	02		"	2	B	R	b	r			¢	²	Â	Ò	â	ò	2
0 0 1 1	03		#	3	C	S	c	s			£	³	Ã	Ó	ã	ó	3
0 1 0 0	04		$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô	4
0 1 0 1	05		%	5	E	U	e	u			¥	µ	Å	Õ	å	õ	5
0 1 1 0	06		&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö	6
0 1 1 1	07		'	7	G	W	g	w			§	·	Ç	×	ç	÷	7
1 0 0 0	08		(	8	H	X	h	x			¨	¸	È	Ø	è	ø	8
1 0 0 1	09		)	9	I	Y	i	y			©	¹	É	Ù	é	ù	9
1 0 1 0	10		*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú	A
1 0 1 1	11		+	;	K	[	k	{			«	»	Ë	Û	ë	û	B
1 1 0 0	12		,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü	C
1 1 0 1	13		-	=	M	]	m	}			SHY	½	Í	Ý	í	ý	D
1 1 1 0	14		.	>	N	^	n	~			®	¾	Î	Þ	î	þ	E
1 1 1 1	15		/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex

Table 2 – Code table of Latin alphabet No. 4

b4 b3 b2 b1 \ b8 b7 b6 b5	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0 0 0 0	00		SP	0	@	P	`	p			NBSP	°	Ā	Ð	ā	đ	0
0 0 0 1	01		!	1	A	Q	a	q			Ą	ą	Á	Ņ	á	ņ	1
0 0 1 0	02		"	2	B	R	b	r			ĸ	˛	Â	Ō	â	ō	2
0 0 1 1	03		#	3	C	S	c	s			Ŗ	ŗ	Ã	Ķ	ã	ķ	3
0 1 0 0	04		$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô	4
0 1 0 1	05		%	5	E	U	e	u			Ĩ	ĩ	Å	Õ	å	õ	5
0 1 1 0	06		&	6	F	V	f	v			Ļ	ļ	Æ	Ö	æ	ö	6
0 1 1 1	07		'	7	G	W	g	w			§	ˇ	Į	×	į	÷	7
1 0 0 0	08		(	8	H	X	h	x			¨	¸	Č	Ø	č	ø	8
1 0 0 1	09		)	9	I	Y	i	y			Š	š	É	Ų	é	ų	9
1 0 1 0	10		*	:	J	Z	j	z			Ē	ē	Ę	Ú	ę	ú	A
1 0 1 1	11		+	;	K	[	k	{			Ģ	ģ	Ë	Û	ë	û	B
1 1 0 0	12		,	<	L	\	l	|		Ŧ	ŧ	Ė	Ü	ė	ü	C
1 1 0 1	13		-	=	M	]	m	}			SHY	Ŋ	Í	Ũ	í	ũ	D
1 1 1 0	14		.	>	N	^	n	~			Ž	ž	Î	Ū	î	ū	E
1 1 1 1	15		/	?	O	_	o				¯	ŋ	Ī	ß	ī	˙	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex

Unicode kan føre til homograf problemer

paypal

(Russian Cyrilliac characters in a unicode font)
actual text is “raural”

paypal

(Standard Latin characters in a unicode font)

Bilde fra http://mashable.com/2010/01/01/idn-phishing/

Det finnes mange mekanismer bla i nettleseren som hindrer at dette er et problem. Eksemplet er ment til å illustrere homograf problemet.

Oppsummering

- Tegnsett kan være et problem for depot men behøver ikke være det
- Norge forholder seg stort sett til Latin 1, del 1
  - Riksarkivet mottar avlevering der
    - Latin 1, del 1 brukes for Norsk
    - Latin 1, del 4 brukes for Samisk
- I et større Europeisk sammenheng (Europeana) så blir det viktigere fra et interoperabilitets standpunkt
- Globalt er det veldig viktig
- Kort fortalt
  - ASCII (7bit), ASCII (8bit), Unicode

Hva har dette med XML å gjøre?

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Dette her er vanskelig å forklare uten den diskusjonen vi har vært gjennom

XML Kurs for eArkivar

Time 2 : Migrasjon og XML

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 11/10/2012

Uttrekksbehandling

Fastform eksempel

```
1-8:ID
       9-17:Fornavn
               18-28:Etternavn
                         29-41:Adresse
                                     42-44:Adresse (tall)
                                        45-48:Postnummer
                                            49-58:Sted

10002__Pål_____Solberg___Storgata_______40182Oslo
10002__Thomas__Hansen____Bakken_________81406Ski
10003__Eli_____Rørvik____Saturnringen__471808Askim
10004__Børre___Andersen__Bekkefaret_____50348Oslo
```

Fastform eksempel med feil

```
1-8:ID
       9-17:Fornavn
               18-28:Etternavn
                         29-41:Adresse
                                     42-44:Adresse (tall)
                                        45-48:Postnummer
                                            49-58:Sted

10002__Pål_____Solberg___Storgata_______40182Oslo
10002__Thomas__Hansen____Bakken_________8b1406Ski
10003__Eli_____Rørvik____Saturnringen__471808Askim
10004__Børre___Andersen__Bekkefaret_____50348Oslo
```

Adresse tall endret til 8b

Kommaseparert

- Forekomster av felter skilles med komma
  - csv, comma seperated values

Hvert linje representer en menneske

– ID, fornavn, etternavn, adresse, adresse (tall), postnummer, poststed
– start
10002, Pål, Solberg, Storgata, 4, 0182, Oslo
10002, Thomas, Hansen, Bakken, 8b, 1406, Ski
10003, Eli, Rørvik, Saturnringen, 47, 1808, Askim
10004, Børre, Andersen, Bekkefaret, 5, 0348, Oslo
– stopp

Kommaseparert med feil

- Feiler og manglende informasjon i filen kan være vanskelig å oppdage
  - Spesielt når store filer er prosessert i bulk
  - Feks hva skjer hvis det mangler et felt?

Hvert linje representer en menneske

– ID, fornavn, etternavn, adresse, adresse (tall), postnummer, poststed
– start
10002, Solberg, Pål, Storgata, 4, 0182, Oslo
10002, Thomas, Bakken, 8b, 1406, Ski
10003, Eli, Rørvik, Saturnringen, 47, 1808, Askim
10004, Børre, Andersen, 5, 0348, Oslo
– stopp

Hvis bare det fantes en måte å tagge dataen på slik at man viste hva hvert enkelt felt betyr

XML

Sånn som dette?

<id> 10002 </id>
<etternavn> Solberg </etternavn>
<adresse> Storgata </adresse>
<fornavn> Pål </fornavn>
<adressetall> 4 </adressetall>
<postnummer> 0182 </postnummer>
<sted> Oslo </sted>

XML som uttrekksformat

- XML er et markeringsspråk og kan brukes både som
  - et uttrekksformat ved eksport og import til depotet
  - format for langtidslagring
- Et markeringsspråk (eng. Markup language) kombinerer tekst og ekstra informasjon (metadata) om teksten
  - Den ekstra informasjonen, for eksempel informasjonen om tekstens struktur og oppsett, blir uttrykt ved bruk av markering, som er innlemmet i hovedteksten

Hvorfor XML ved uttrekksbehandling

- Selvbeskrivende
- Interoperabilitet
- Lett å utvide
- Ikke proprietær

Makeringsspråk

- Begrepet “Markup Language” stammer fra prosessen med å brekke et manuskript, der symbolsk instruksjoner er lagt til et manuskript som er tolket ved utskrift/publisering
  - GenCode from 1960s
  - Scribe 1980
  - GML
  - SGML
  - HTML
  - XML
  - xhtml

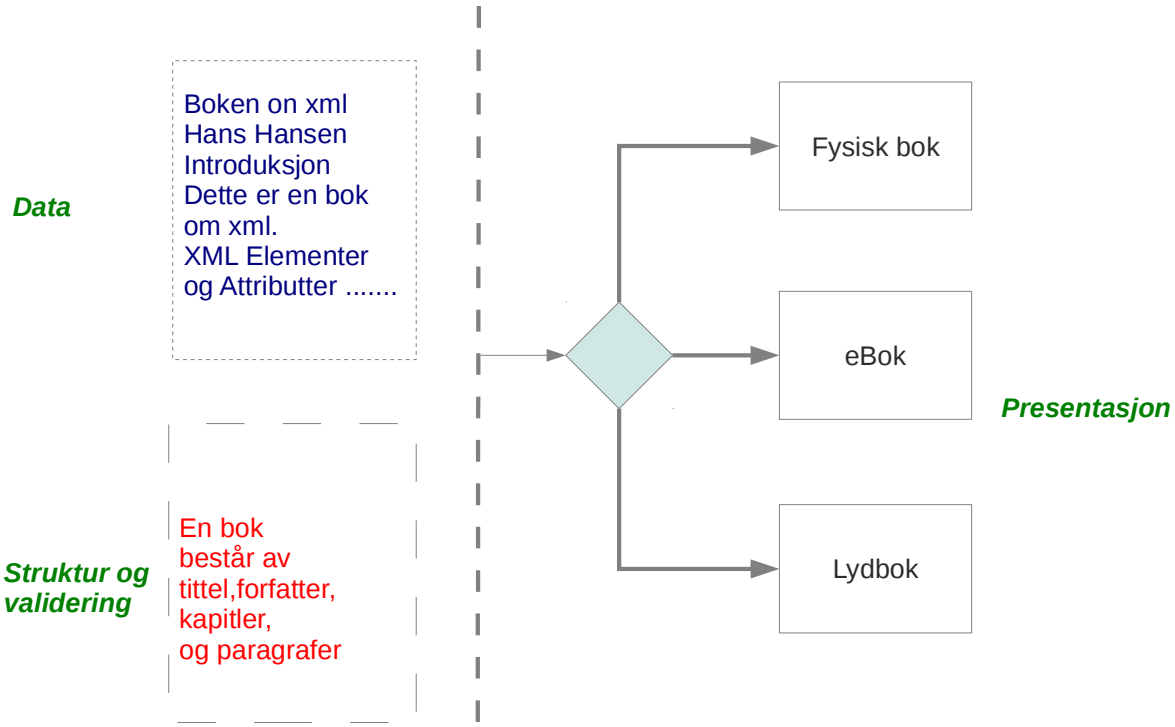
Markeringsspråket XML

- eXtensible Markup Language
- Forenkler noen aspekter av SGML
- Mye mer fleksibelt og tilpasningsdyktig enn HTML
- Utgitt av w3 konsortiet
  - http://www.w3.org/

En 'suite' av teknologier

- XML
  - Markerings språk
- XSD/DTD
  - Definerer struktur og kan brukes til å validere
- XSLT
  - Brukes til å presentere/endre data
- Flere men ikke dekket i kurset her

Data / Struktur / Presentasjon



Vi er kun interessert i disse delene

XML

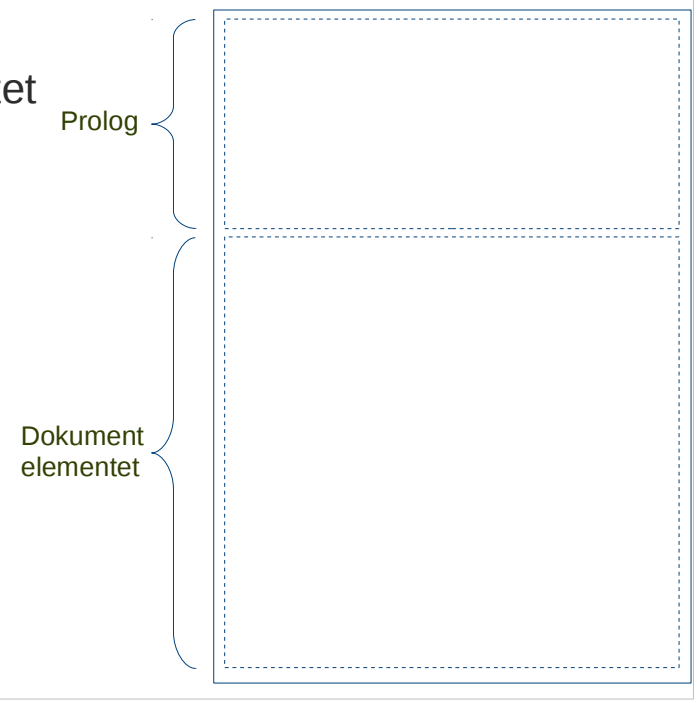
- Et XML dokument er et dokument som inneholder data som er markert på en bestemt måte
- På mange måter er XML et metaspråk for å lage forskjellige tekstmarkeringspråk som brukes til å beskrive noe tekst
  - Det er et verktøy
- Brukes for Noark 4/5 avlevering og det brukes i elektronisk arkivpakker (DIAS)
  - Det er en viktig standard i både dannings og bevarings fasene
    - Interoperabilitet, uttrekk, langtidslagring

XML Dokument

- Fornuftig valg av elementnavn gjør lesing og forståelse av XML filen intuitivt
- Et XML dokument består av en prolog og et rotelement (og alt xml'n)
  - Rotelementet kalles også for dokument elementet
  - Det er bare det første elementet i XML dokumentet
- Alt etter rotelementets avslutnings element ansees som søppel

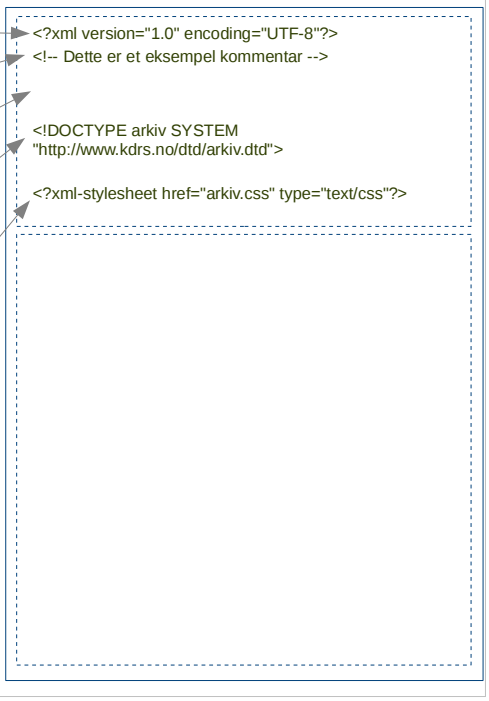
Hva et XML dokument består av

- Et XML dokument består av en
  - Prolog
  - Dokumentelementet



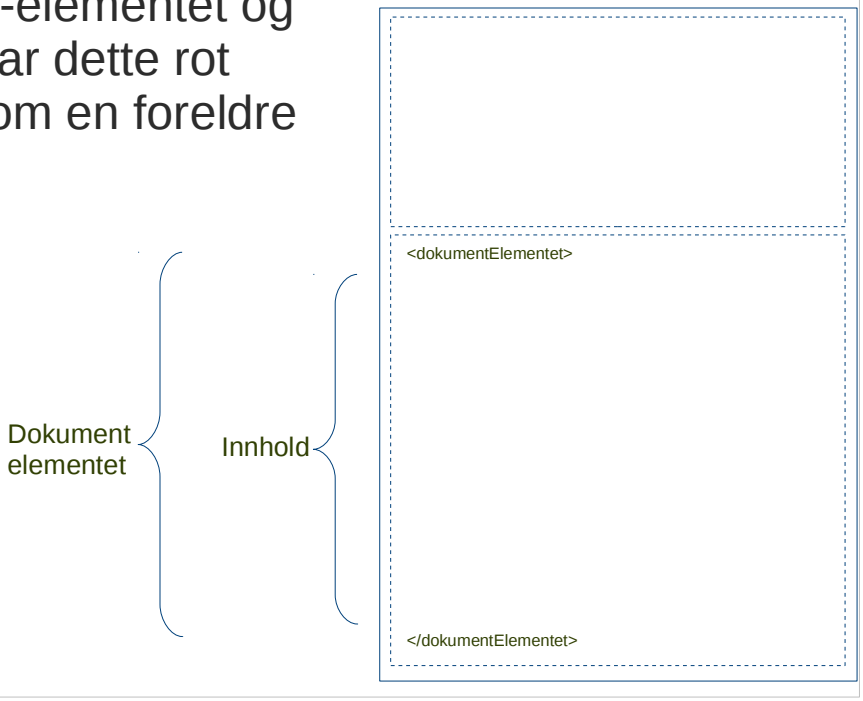
Hva et XML dokument består av

- Prologen består av:
  - XML deklarasjon
  - Kommentarer
  - Blanke linjer
  - Struktur valideringsinformasjon
  - Prosessorinstruksjoner
    - feks stilinformasjon

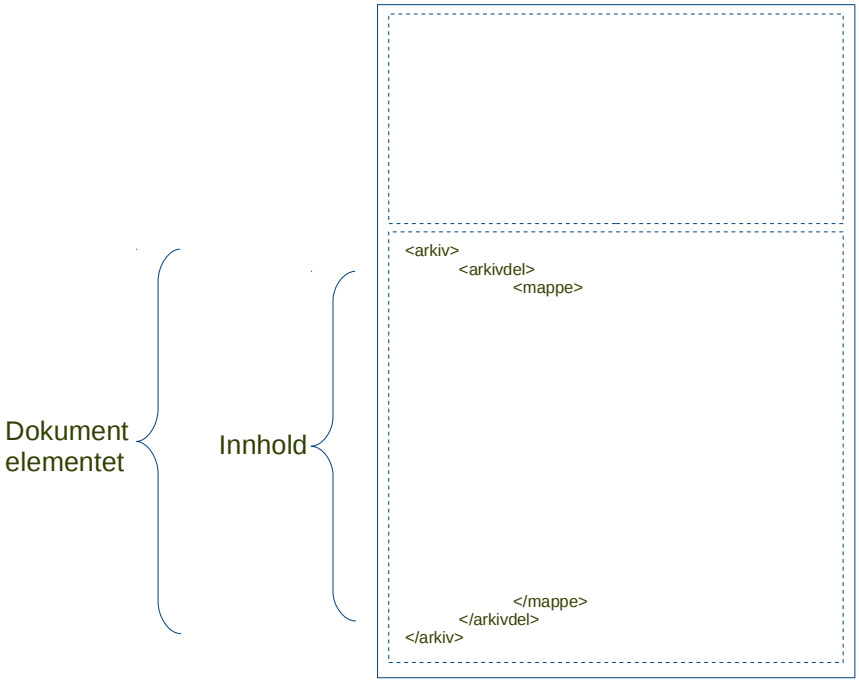


Hva et XML dokument består av

- Dokument elementet er også kalt rot-elementet og alt innhold har dette rot elementet som en foreldre



En kjapp eksempel



Praktisk

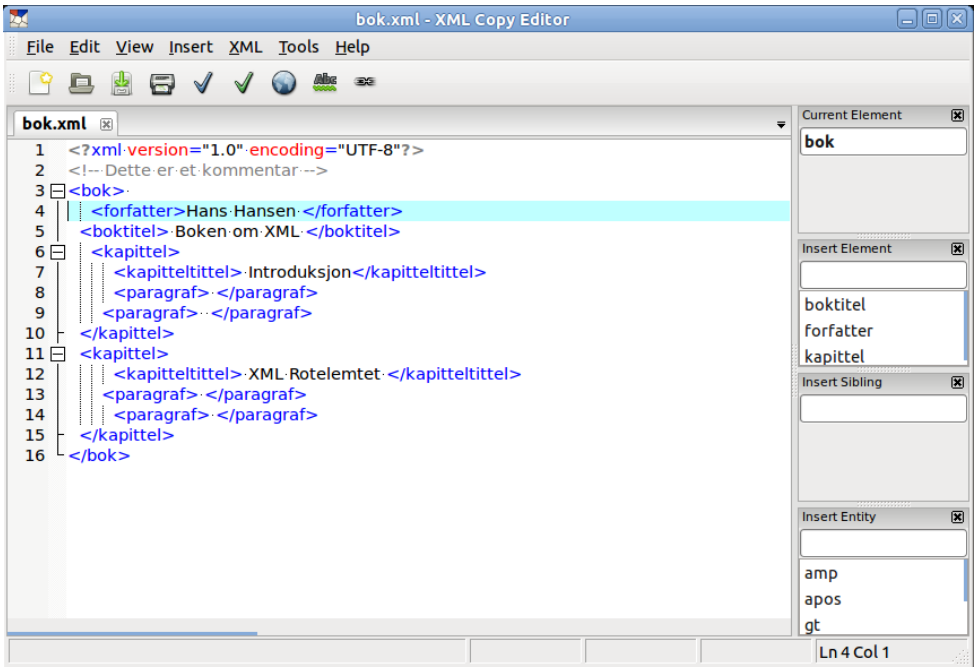
- Lage vår første XML dokument med prolog men uten dokumentelementet

```
<?xml version="1.0" encoding="UTF-8"?>
 <!-- Dette er et kommentar -->
```

xmlcopyeditor

- Vi bruker et program som heter *xmlcopyeditor* for å utforske og jobbe med XML
  - Kan lastes ned fra
    - http://xml-copy-editor.sourceforge.net/
  - Greit for en intro kurs
- Hvis du jobber med XML i depot vil du sikkert bruke proprietær XML redigerings programvare
  - Det har med skalerbarhet og brukervennlighet
- Det er prinsippene vi lærer her

xmlcopyeditor

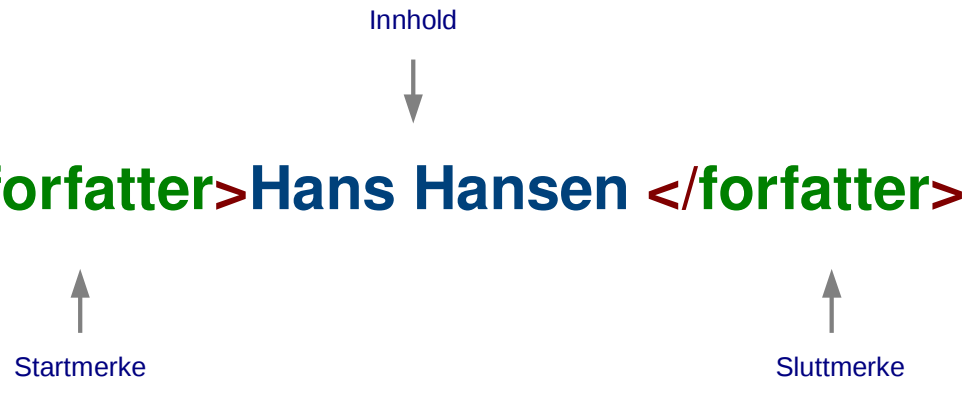


Tagger/Merker

- Først må du forstå en tagg/merke
- Tagg avgrenset av **<** og **>**
  - Eksempler på tagger er
    - <fra>, <til>, <navn>

Elementer

- Elementer er selve fundamentet i merking av data i XML filer og består av
  - Startmerke
  - Innhold
  - Sluttmerke



Elementer

- Det er vanlig at navnet til start og slutt merker beskriver selve innholdet
  - derfor sier vi at XML er selvbeskrivende
- Forslag på elementnavn som kan brukes til å beskrive et menneske?
  - <person></person>
  - <navn></navn>
  - <alder></alder>

Elementer

- Når vi omfavner data med start og slutt merker så markerer vi data
  - og dette er en av grunnene hvorfor XML er riktig format når det gjelder uttrekksbehandling

XML Kurs for eArkivar

Time 3 : Praktisk øvelse

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 11/10/2012

Denne timen

- Vi skal utvikle 4 forskjellige XML filer
  - navn.xml
  - menneske.xml
  - menneskeListe.xml
  - bok.xml
- Og vi bruker disse filene igjen senere i dag og i morgen

navn.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Dette er et kommentar -->

<navn>
    <fornavn>Hans</fornavn>
    <etternavn>Hansen</etternavn>
</navn>
```

Identifiser

- Hvis vi ser på navn.xml
  - Hvor er prologen?
  - Hva er rotelementet?
  - Hva skjer hvis vi legger til noe etter </navn>?

Mellomrom i data

- Mellomrom kan være utfordrende når det oppstår i innhold

<etternavn> Hansen </etternavn>

menneske.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneske>
    <fornavn>Hans</fornavn>
    <mellomnavn>John</mellomnavn>
    <etternavn>Hansen</etternavn>
    <personnummer>01108298649</personnummer>
    <kjønn>mann</kjønn>
</menneske>
```

Men jeg vil ha en liste av mennesker

- Hva skjer hvis vi prøver å lage en til <menneske></menneske> i samme fil?
  - Søppel etter rotelementet!
- <menneskeListe></menneskeListe>
- Dette er en klassisk utfordring en nybegynner støtter på
- Du må faktisk planlegge XML din
  - Det er noe vi egentlig gjør med XSD

menneskeListe.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneskeListe>
   <menneske>
      <fornavn>Hans</fornavn>
      <mellomnavn>John</mellomnavn>
      <etternavn>Hansen</etternavn>
      <personnummer>01108298649</personnummer>
      <kjønn>mann</kjønn>
    </menneske>
</menneskeListe>
```

bok.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bok>
        <forfatter>Hans Hansen</forfatter>
        <boktittel>Boken om XML</boktittel>
            <kapittel>
                <kapitteltittel>Introduksjon</kapitteltittel>
                <paragraf></paragraf>
                <paragraf></paragraf>
        </kapittel>
        <kapittel>
                <kapitteltittel>XML Rotelementet</kapitteltittel>
                <paragraf></paragraf>
                <paragraf></paragraf>
        </kapittel>
</bok>
```

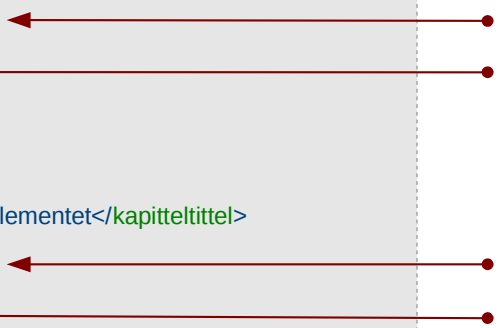
Tomme elementer

- Elementer som ikke har noe data er tomme
  - <paragraf></paragraf>
- og kan skrives slik
  - <paragraf/>

- Tomme elementer skal aldri være med i et Noark 5 uttrekk
  - Krav 5.12.5 Metadataelementer som ikke har verdi, skal utelates fra arkivuttrekket. I uttrekket skal det med andre ord ikke forekomme tomme elementer med kun start- og slutt-tagg

Tomme elementer

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Dette er et kommentar -->
<bok>
        <forfatter>Hans Hansen</forfatter>
        <boktittel>Boken om XML</boktittel>
            <kapittel>
                <kapitteltittel>Introduksjon</kapitteltittel>
                <paragraf></paragraf>
                <paragraf/>
        </kapittel>
        <kapittel>
                <kapitteltittel>XML Rotelementet</kapitteltittel>
                <paragraf></paragraf>
                <paragraf/>
        </kapittel>
</bok>
```



XML Kurs for eArkivar

Time 4 : Praktisk øvelse og attributter

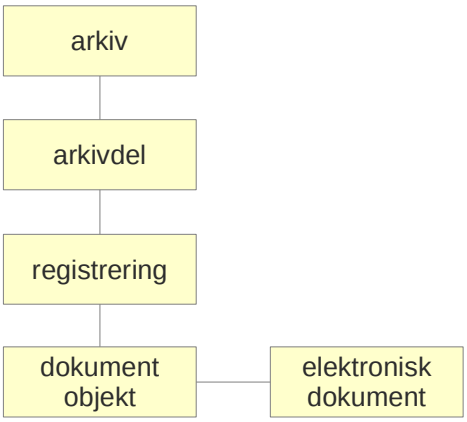
KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 11/10/2012

Denne timen

- Nå tar vi og relaterer dette til arkiv og skal utvikle 2 filer
  - arkiv_1.xml
  - arkiv_2.xml
- Samtidig er det viktig å jobbe med forskjellige eksempler
- Nå kommer vi til å utvide og utvide
- Bruker Noark 5 metadatakatalogen
  - http://arkivverket.no/arkivverket/content/download/7898/80003/version/1/file/Vedlegg+2+Metadatakatalog++objektsortert.pdf

1. nivå: **arkiv (*)**
2. nivå: **arkivdel**
3. nivå: **registrering**
4. nivå: **dokumentobjekt**




```
<arkiv>
    <arkivdel>
        <registrering>
            <dokumentobjekt>
            </dokumentobjekt>
        </registrering>
    </arkivdel>
</arkiv>
```

Tilfellet der kun ett dokument er tilknyttet en registrering

Metadata for <dokumentobjekt>

Nr.	**Navn**	**Forek.**	**Datatype**
M005	versjonsnummer	1	Heltall
M700	variantformat	1	Tekststreng
M701	format	1	Tekststreng
M702	formatDetaljer	0-1	Tekststreng
M600	opprettetDato	1	Dato og klokkeslett
M601	opprettetAv	1	Tekststreng
M218	referanseDokumentfil	1	Tekststreng (filkatalogstruktur + filnavn)
M705	sjekksum	1	Tekststreng
M706	sjekksumAlgoritme	1	Tekststreng
M707	filstoerrelse	1	Tekststreng

Metadata for <registrering>

Navn	Forek	Avl.	Datatype
systemID	1	A	Tekststreng
opprettetDato	1	A	Dato og klokkeslett
opprettetAv	1	A	Tekststreng
arkivertDato	1	A	Dato og klokkeslett
arkivertAv	1	A	Tekststreng
referanseArkivdel	0-M	A	arkivdel.systemID

Metadata for <arkivdel>

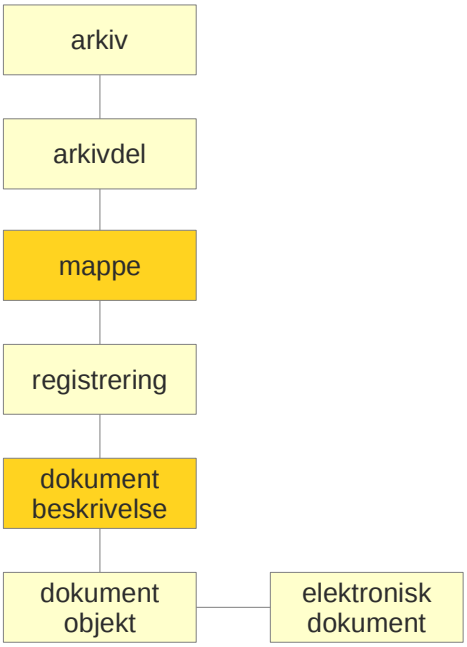
Nr.	Navn	Forek	Datatype
M001	systemID	1	Tekststreng
M020	tittel	1	Tekststreng
M021	beskrivelse	0-1	Tekststreng
M051	arkivdelstatus	1	Tekststreng
M300	dokumentmedium	0-1	Tekststreng
M301	oppbevaringssted	0-M	Tekststreng
M600	opprettetDato	1	Dato og klokkeslett
M601	opprettetAv	1	Tekststreng
M602	avsluttetDato	1	Dato og klokkeslett
M603	avsluttetAv	1	Tekststreng
M107	arkivperiodeStartDato	0-1	Dato
M108	arkivperiodeSluttDato	0-1	Dato
M202	referanseForloeper	0-1	arkivdel.systemID
M203	referanseArvtaker	0-1	arkivdel.systemID

Metadata for <arkiv>

Nr.	Navn	Forek.	Datatype
M001	systemID	1	Tekststreng
M020	tittel	1	Tekststreng
M021	beskrivelse	0-1	Tekststreng
M050	arkivstatus	0-1	Tekststreng
M300	dokumentmedium	0-1	Tekststreng
M301	oppbevaringssted	0-M	Tekststreng
M600	opprettetDato	1	Dato og klokkeslett
M601	opprettetAv	1	Tekststreng
M602	avsluttetDato	1	Dato og klokkeslett
M603	avsluttetAv	1	Tekststreng

Time out!

- Filen blir fort stor
- Vanskelig å holde styr på nøsting og rekkefølge
  - Påminelse at F11 i xmlcopyeditor er Pretty Print
- Nå lagrer vi filen som arkiv_2.xml og fortsetter med neste foil



Ingen klassifikasjonsystem

1. nivå: **arkiv (*)**
2. nivå: **arkivdel**
3. nivå: **mappe (*)**
4. nivå: **registrering**
5. nivå: **dokumentbeskrivelse**
6. nivå: **dokumentobjekt**

```
<arkiv>
    <arkivdel>
        <mappe>
            <registrering>
                <dokumentbeskrivelse>
                    <dokumentobjekt>
                    </dokumentobjekt>
                </dokumentbeskrivelse>
            </registrering>
        </mappe>
    </arkivdel>
</arkiv>
```

Metadata for <mappe>

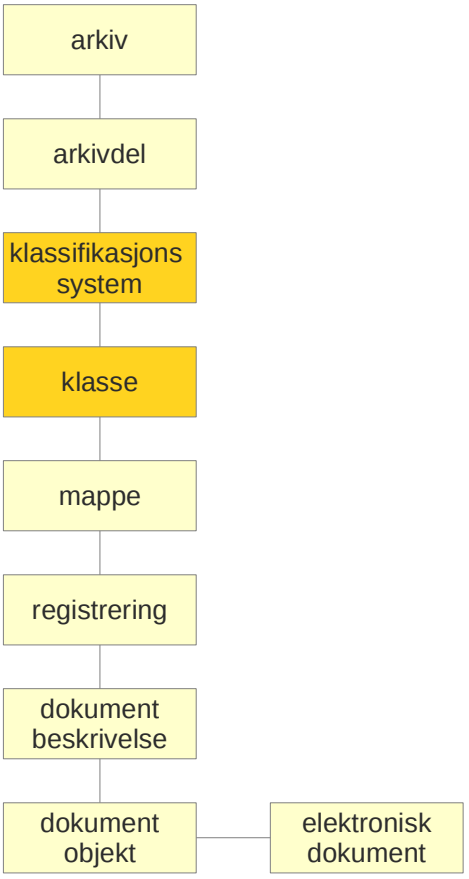
Nr.	Navn	Forek	Datatype
M001	systemID	1	Tekststreng
M003	mappeID	1	Tekststreng
M020	tittel	1	Tekststreng
M025	offentligTittel	0-1	Tekststreng
M021	beskrivelse	0-1	Tekststreng
M022	noekkelord	0-M	Tekststreng
M300	dokumentmedium	0-1	Tekststreng
M301	oppbevaringssted	0-M	Tekststreng
M600	opprettetDato	1	Dato og klokkeslett
M601	opprettetAv	1	Tekststreng
M602	avsluttetDato	1	Dato og klokkeslett
M603	avsluttetAv	1	Tekststreng
M208	referanseArkivdel	0-M	arkivdel.systemID
M711	virksomhetsspesifikke Metadata	0-1	Vilkårlig struktur

Metadata for <dokumentbeskrivelse>

Nr.	Navn	Forek	Datatype
M001	systemID	1	Tekststreng
M083	dokumenttype	1	Tekststreng
M054	dokumentstatus	1	Tekststreng
M020	tittel	1	Tekststreng
M021	beskrivelse	0-1	Tekststreng
M024	forfatter	0-M	Tekststreng
M600	opprettetDato	1	Dato og klokkeslett
M601	opprettetAv	1	Tekststreng
M300	dokumentmedium	0-1	Tekststreng
M301	oppbevaringssted	0-1	Tekststreng
M208	referanseArkivdel	0-M	arkivdel.systemID
M217	tilknyttetRegistreringSom	1	Tekststreng
M007	dokumentnummer	1	Heltall
M620	tilknyttetDato	1	Dato og klokkeslett
M621	tilknyttetAv	1	Tekststreng

Time out!

- Nå skal du føle at du mestrer dette
- Du ser en helhet og forstår hvordan det er å videreutvikle filen
- Neste foil skal vi ikke jobbe med men hvis dere selv ønsker så kan dere utvikle arkiv_3.xml i første time i morgen



1. nivå: **arkiv (*)**
2. nivå: **arkivdel**
3. nivå: **klassifikasjonssystem (*)**
4. nivå: **klasse**
5. nivå: **mappe (*)**
6. nivå: **registrering**
7. nivå: **dokumentbeskrivelse**
8. nivå: **dokumentobjekt**

```
<arkiv>
    <arkivdel>
        <klassifikasjonssystem>
            <klasse>
                <mappe>
                    <registrering>
                        <dokumentbeskrivelse>
                            <dokumentobjekt>
                            </dokumentobjekt>
                        </dokumentbeskrivelse>
                    </registrering>
                </mappe>
            </klasse>
        </klassifikasjonssystem>
    </arkivdel>
</arkiv>
```

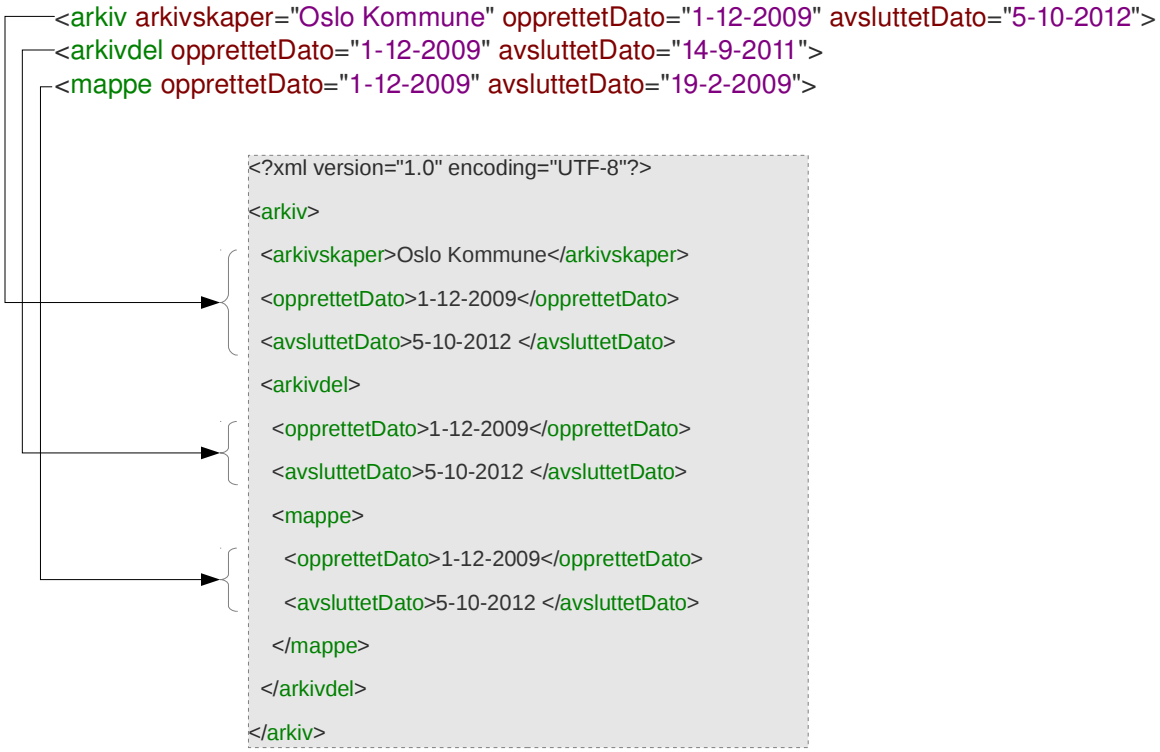
XML Attributter

- Metadata om data i et element
  - Gir ytterlige informasjon
- Består av et navn og verdi par som er en del av en startmerke av en element

<elementnavn attributt="verdi"> data </elementnavn>

```
<arkiv arkivskaper="Oslo Kommune" opprettetDato="1-12-2009" avsluttetDato="5-10-2012">
<arkivdel opprettetDato="1-12-2009" avsluttetDato="14-9-2011">
 <mappe opprettetDato="1-12-2009" avsluttetDato="19-2-2009">
```

XML eksempel med attributter



XML eksempel med attributter

```
<?xml version="1.0" encoding="UTF-8"?>


<arkiv arkivskaper="Oslo Kommune" opprettetDato="1-12-2009" avsluttetDato="5-10-2012">
 <systemID>E9F3C110-F8C4-11E1-A21F-0800200C9A66</systemID>
 <tittel>Kollektivtransport</tittel>
 <arkivdel opprettetDato="1-12-2009" avsluttetDato="14-9-2011">
  <systemID>BB46B6E9-D81E-4D3F-8450-4C991F44BD98</systemID>
  <tittel>Innkjøp</tittel>
  <mappe opprettetDato="1-12-2009" avsluttetDato="19-2-2009">
   <systemID>481C46C5-EF2A-4AEA-BA51-93233DF331FC</systemID>
   <tittel>Innkjøp av busser fra MAN</tittel>
  </mappe>
 </arkivdel>
</arkiv>
```

Samme eksempel uten attributter

```
<?xml version="1.0" encoding="UTF-8"?>
<arkiv>
 <opprettetDato>1-12-2009</opprettetDato>
 <avsluttetDato>5-10-2012 </avsluttetDato>
 <systemID>E9F3C110-F8C4-11E1-A21F-0800200C9A66</systemID>
 <tittel>Kollektivtransport</tittel>
 <arkivdel>
  <opprettetDato>1-12-2009</opprettetDato>
  <avsluttetDato>5-10-2012 </avsluttetDato>
  <systemID>BB46B6E9-D81E-4D3F-8450-4C991F44BD98</systemID>
  <tittel>Innkjøp</tittel>
  <mappe>
   <opprettetDato>1-12-2009</opprettetDato>
   <avsluttetDato>5-10-2012 </avsluttetDato>
   <systemID>481C46C5-EF2A-4AEA-BA51-93233DF331FC</systemID>
   <tittel>Innkjøp av busser fra MAN</tittel>
  </mappe>
 </arkivdel>
</arkiv>
```

Når bruker vi attributter?

- Det er valgfri men bare hvis antall forekomster av innhold er lik 0 eller 1
  - Hvis attributtet kan forekomme mange ganger så må det være element
  - Oppbevaringssted kan forekomme mange ganger for fysisk arkiv
    - <oppbevaringsted>P48-R428-1</oppbevaringsted>
    - <oppbevaringsted>P48-R428-1</oppbevaringsted>
  - Oppbevaringssted kan ikke være en attributt

Attributt begrensinger

- Attributter er mer restriktive enn elementer
  - Kan ikke nøste elementer i en attributt
  - Kan bare brukes en gang
  - Kan vanskeliggjøre framtidige endringer
- Noark 5 uttrekk har ingen attributter
- De forskjellige DIAS standardene har varierende bruk av attributter
- Vi prøver å unngå bruken av attributter

Se også http://recycledknowledge.blogspot.no/2008/03/elements-or-attributes.html

XML Kurs for eArkivar

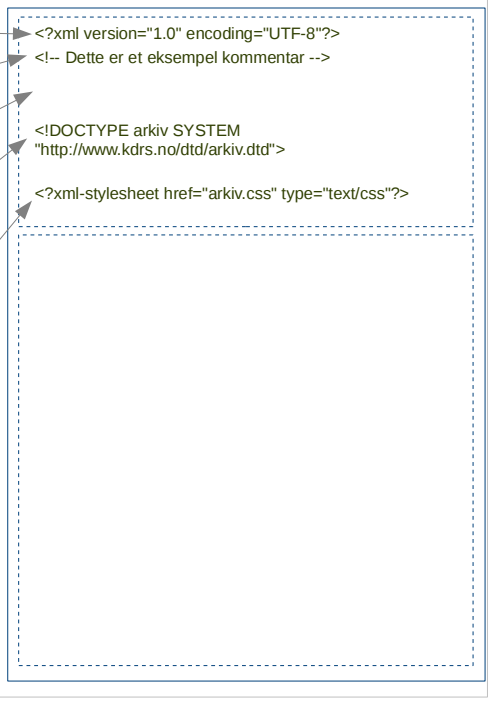
Time 5 : Navnerom og prosessering

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 11/10/2012

Hva et XML dokument består av

- Prologen består av:
  - XML deklarasjon
  - Kommentarer
  - Blanke linjer
  - Struktur valideringsinformasjon
  - Prosessorinstruksjoner
    - feks stilinformasjon

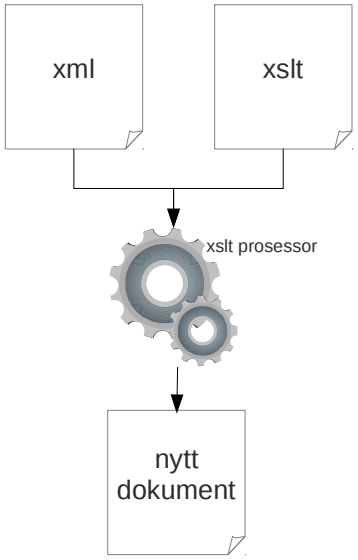


Prosesseringsinstruksjoner

- Prosesseringsinstruksjoner er ikke en del av et XML dokument
  - angir instruksjoner til en (ekstern) applikasjon som skal prosessere XML dokumentet
    - Feks konvertere det til et annet format
  - Begynner med
    - <? og avsluttes med ?>
  - Eksempel:
    - <?xml-stylesheet href="arkiv.css" type="text/css"?>
    - XML filen blir formatert på bakrgunn av css format instruksjoner angitt i filen arkiv.css

Prosesseringsinstruksjoner

- xslt spesifiserer hvordan xml skal prosesseres for å lage et nytt dokument
- Trenger en xslt prosessor (firefox)



Prosessering av arkiv.xml

- arkiv.xml har 5 mapper med tittel og offentligtittel
  - arkiv_offentlig.xslt printer ut bare offentligtittel
  - arkiv_intern.xslt printer offentligtittel og tittel
- Demo

Prosessering av arkiv.xml

- Vi kan også legge til stilinformasjon og angi hvordan innholdet skal vises
  - arkiv.css

Prosesseringsinstruksjoner

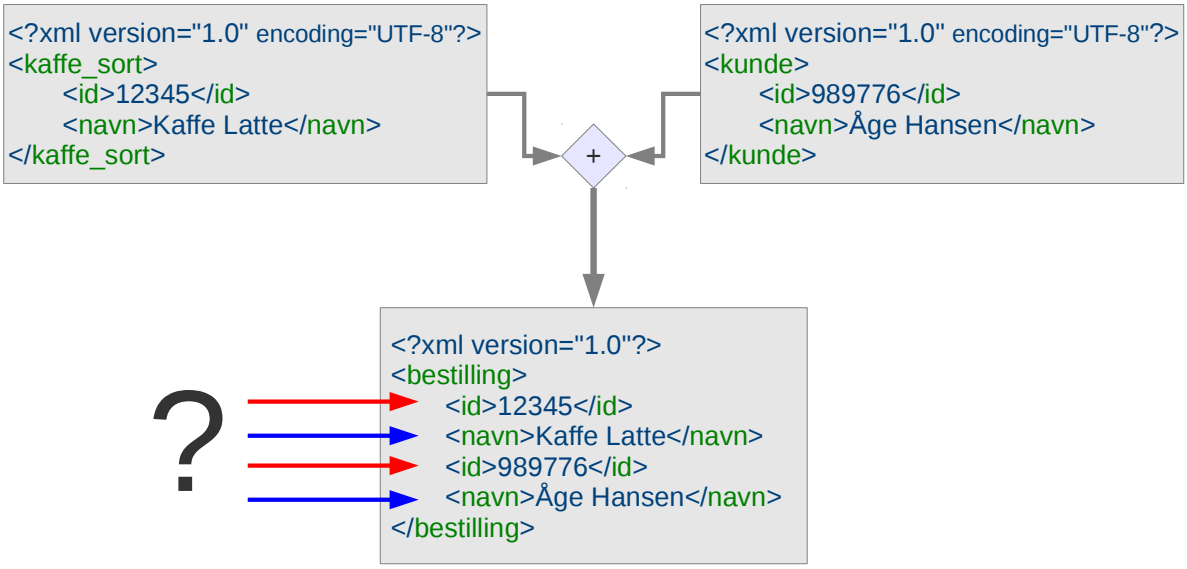
- Dette er noe som kommer til å bli viktigere for fagfeltet
- Når vi lagrer XML er det for å prosessere eller bevare?
- Inntrykk at fagfeltet er veldig avhengig av RDBMS for bruk av data

XML navnerom (namespace)

- Navnerom brukes i XML for å hindre forvirring hvis en organisasjon bruker forskjellige XML filer der element navn blir gjenbrukt
  - Feks dato
- Oppretter en 'domene' slik at overlappende element navn ikke skaper konflikt

Når bruker vi navnerom?

- XML navnerom er en måte å unngå konflikt når to forskjellige XML filer har elementer med samme navn



Når bruker vi navnerom?

- Eksempelet viser to forskjellige *'domener'*
- For å unngå konflikt så får alle elementer en prefiks som identifiserer domenet elementer tilhører

Navnerom eksempel

```
<?xml version="1.0" encoding="UTF-8"?>

<bestilling
    xmlns:ks="http://www.kdrs.no/xmlns/kaffe_sort"
    xmlns:kunde="http://www.kdrs.no/xmlns/kunde">

    <ks:id>12345</ks:id>
    <ks:navn>Kaffe Latte</ks:navn>

    <kunde:id>989776</kunde:id>
    <kunde:navn>Åge Hansen</kunde:navn>

</bestilling>
```

XML navnerom

- Syntaks: <prefiks:element xmlns:prefiks="uri">
  - Eksempel

    <ark:arkiv xmlns:ark="http://www.kdrs.no/arkiv/">
  - Betyr at elementet arkiv er definert med navnerommet http://www.kdrs.no/arkiv/
    - Adressen http://www.kdrs.no/arkiv/ behøver ikke å finnes
    - Det brukes for å skape en unik verdi på
    - Kan også gi assosiasjoner til organisasjonen
  - I Noark 5 uttrekk bruker Riksarkivet
    - http://www.arkivverket.no/standarder/noark5/metadatakatalog
    - Som navnerom

XML navnerom

- Hvis du vet at du bare skal bruke innholdet i en lokal kontekst så trenger du ikke navnerom
- Brukes når du blander fra forskjellige 'domener'
  - Ikke bare to domener som i eksempelet
- Blander betyr ikke nødvendigvis at vi skal lage en XML som i eksempelet
- Det holder at vi må skille internt i organisasjonen
  - Feks internt i depot ønsker vi å skille mellom Noark 3 / Noark 4 og Noark 5 uttrekk

Navnerom

- Vi har egentlig gjort følgende når vi bruker navnerom

```
<?xml version="1.0" encoding="UTF-8"?>

<bestilling>

 <{http://www.kdrs.no/xmlns/kaffe_sort}id> 12345 </{http://www.kdrs.no/xmlns/kaffe_sort}id>
 <{http://www.kdrs.no/xmlns/kaffe_sort}navn> Kaffe Latte </{http://www.kdrs.no/xmlns/kaffe_sort}navn>

 <{http://www.kdrs.no/xmlns/kunde}id>989776</{http://www.kdrs.no/xmlns/kunde}id>
 <{http://www.kdrs.no/xmlns/kunde}navn>Åge Hansen</{http://www.kdrs.no/xmlns/kunde}navn>

</bestilling>
```

..... men dette er ikke lett å lese

Dette er for illustrasjon, dette er ikke en riktig xml fil

Ulemper/Kritikk av XML

- Unødvendig!
- For mye markeringdata
  - Filstørrelsen større en nødvendig
    - Kan være et problem der det er begrenset bredbånd / Internet koster mye (?)

```
<person>
    <fornavn> Hans </fornavn>
    <etternavn>Hansen</etternavn>
    <alder>45</alder>
</person>
```

95 tegn

```
(person
    fornavn(Hans)
    etternavn(Hansen)
    alder(45)
    )
```

61 tegn

Fordeler

- Markup er ren tekst og menneske lesbar
- Uavhengig av system, programvare og maskinvare
- Ikke proprietær
- (Relativt) enkel å implementere løsninger
  - Importere fra / eksporter til database
  - Data er adskilt fra presentasjon

XML som arkivformat

- XML brukt i bevaringssammenheng
  - Avlevering til depot for Noark 5
  - DIAS spesifisere en nasjonal arkivpakkeformat som bygger veldig mye på XML
- XML brukt i danningssammenheng
  - Eksport/import til Noark kjerne
  - Utveksling av saksdokumentasjon (BEST)

XML Kurs for eArkivar

Time 2 : XML regler og validering

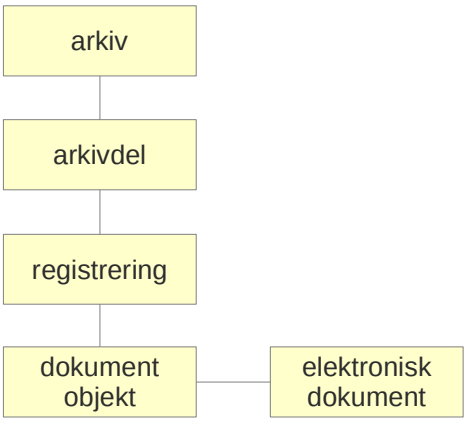
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 12/10/2012

Hvorfor se på XSD

- Vi vet at Noark 5 uttrekk skal avleveres i XML format
  - Men er det noe forskjell mellom en riktig eller gal XML fil?
  - Depot må stille krav til en avlevering
    - XML filen må følge noen bestemte syntaks regler
    - Strukturen må følge en Noark-5 struktur
      - Depot kan stille krav til XML filen, at den skal følge en bestem struktur som kan verifiseres
      - feks en *arkivdel* må være en del av en *arkiv*
      - feks en *sak* må være en del av en *arkivDel*
  - Verifisering av struktur er det vi skal se på nå

1. nivå: **arkiv (*)**
2. nivå: **arkivdel**
3. nivå: **registrering**
4. nivå: **dokumentobjekt**




```
<arkiv>
    <arkivdel>
        <registrering>
            <dokumentobjekt>
            </dokumentobjekt>
        </registrering>
    </arkivdel>
</arkiv>
```

Tilfellet der kun ett dokument er tilknyttet en registrering

Metadata for <arkivdel>

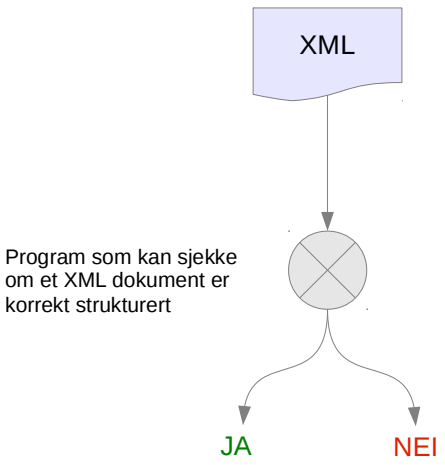
Nr.	Navn	Forek	Datatype
M001	systemID	1	Tekststreng
M020	tittel	1	Tekststreng
M021	beskrivelse	0-1	Tekststreng
M051	arkivdelstatus	1	Tekststreng
M300	dokumentmedium	0-1	Tekststreng
M301	oppbevaringssted	0-M	Tekststreng
M600	opprettetDato	1	Dato og klokkeslett
M601	opprettetAv	1	Tekststreng
M602	avsluttetDato	1	Dato og klokkeslett
M603	avsluttetAv	1	Tekststreng
M107	arkivperiodeStartDato	0-1	Dato
M108	arkivperiodeSluttDato	0-1	Dato
M202	referanseForloeper	0-1	arkivdel.systemID
M203	referanseArvtaker	0-1	arkivdel.systemID

Hvordan håndterer vi dette i XML?

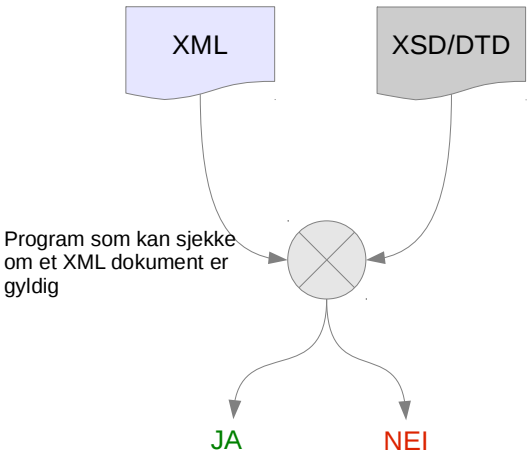
Hva kan kalles en XML dokument?

- For at et dokument skal kunne kalles en XML dokument så må den være både
  - *korrekt strukturert* eller *eng*:well-formed
  - *gyldig* eller *eng*:valid
- Hvorfor er dette viktig?
  - Arkivskaper skal lage uttrekk som er både korrekt strukturert og gyldig
    - Hvis ikke, så skal ikke depot ta imot disse
  - Om noe er korrekt strukturert kan lett sjekkes
  - Hva som er gyldig er noe som depot selv bestemmer

Korrekt strukturert?



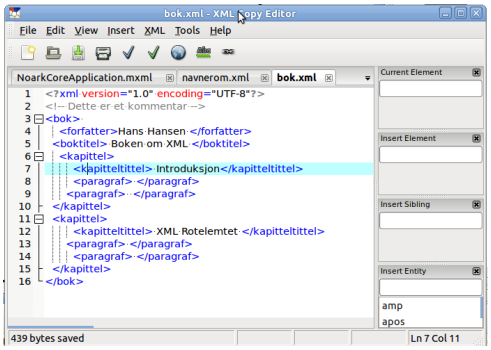
Gyldig?



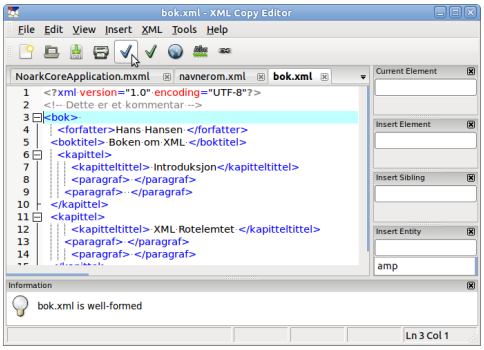
Korrekt strukturert og gyldig

- Et *korrekt strukturert* XML dokument er et som er i overensstemmelse med alle XML syntaks regler
  - Et dokument som ikke er korrekt strukturert *anses ikke å være et XML* dokument
- Et *gyldig* XML dokument er en som er både *korrekt strukturert* og som oppfyller noen bestemte semantiske regler

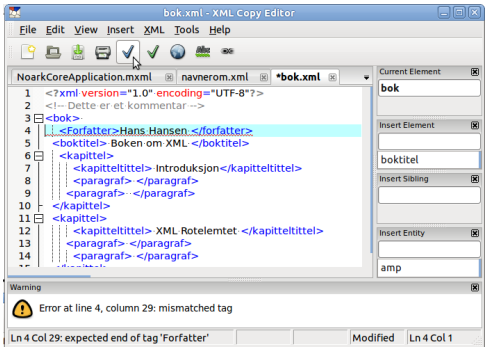
Korrekt Strukturert?



1. xmlcopyeditor kan brukes til å teste om en XML fil er korrekt strukturert.



2.Trykk på den blå knappen for å teste XML filen. Hvis filen er korrekt strukturert så får du melding om det.



3.Jeg legger til en opplagt feil (linje 4 f->F) og igjen trykker på den blå knappen for å teste XML filen. Filen er ikke lenger korrekt strukturert og vi får melding om det.

Korrekt strukturert regler

- XML dokumentet kan kun ha ett rot element
- Ikke-tomme elementer må bruke startmerke og en sluttmerke med samme navn
  - <navn> </navn>
- Tomme elementer kan angis med en selvlukkende merke
  - <navn/>

Korrekt strukturert regler

- Attributtverdier må bruke anførselstegn
  - Fritt fram til å bruke enten ' eller "
  - Du kan ikke mikse bruken av ' eller "
    - Feks "verdi' eller 'verdi" er ikke lov
    - Må være enten "verdi" eller 'verdi'
  - Hvis du ønsker å bruke anførselstegn som en del av attributtverdien kan du bruke den andre type anførselstegn enn den du bruker til å identifisere attributtet
    - "En 'fantastisk' tur", 'En "fantastisk" tur'
- Det er ***INGEN*** krav om å bruke camelCase men det er ofte brukt i XML
  - storageLocation

Korrekt strukturert regler

- Elementer kan inneholde andre elementer men de har ikke lov til overlappe hverandre
  - <bil><farge>...</farge></bil> *er lov*
  - <bil><farge>...</bil></farge> *er ikke lov*
- Det er lov å bruke små og store bokstaver i samme elementnavn
  - <Bil> og </Bil> er et velformet par
  - <Bil> og </bil> er ikke et velformet par
- Det hadde vært rart om du hadde <arkiv> og <Arkiv> i samme XML fil (selv om det er lov)
  - Hvorfor? Selvbeskrivende

Korrekt strukturert regler

- Noen tegn kan ikke brukes som innhold i et element
  - < erstattes med <
  - > erstattes med >
  - " erstattes med "
  - ' erstattes med '
  - & erstattes med &

<tall> 5 > 4 </tall>

blir til

<tall> 5 > 4 </tall>

Korrekt strukturert regler

- Elementnavn har kun lov til å starte med en ***bokstav*** eller ***understrek***
  - <1arkivNoekkel></1arkivNoekkel> *er ikke lov*
  - <*arkivNoekkel1*></*arkivNoekkel1*> *er lov*
  - <_1arkivNoekkel></_1arkivNoekkel> *er lov*
- Resten av elementet kan bruke tall, bindestrek, punktum, bokstaver, eller flere understrek
  - <_2011.3.2-KKODE><_2011.3.2-KKODE> *er lov*
  - <2011.3.2-KKODE><2011.3.2-KKODE> *er ikke lov*

Korrekt strukturert regler

- Lengden på elementnavnet er ubegrenset
- Det er kun tillat med bokstaver, tall, bindestrek, punktum og understrek i element navnet
  - Andre symboler er ikke tillatt i elementnavn
  - <navn> *er lov*
  - <na vn> og <na?vn> og <na&vn> *er ikke lov*
- Kolon kan bare brukes i elementnavn når navnerom er i bruk
  - <navn:fornavn> *er ikke lov* **(hvis ikke navn er spesifisert som navnerom)**

Praktisk

- Her var det mye som skal forståes
- Nå bruker vi litt tid og går gjennom de forskjellige reglene
- Lag et XML dokument der du tester reglene med elementnavn

XML og gyldighet

- Det er to måter å se om en XML-fil er gyldig
  - Document Type Definition (DTD)
  - XML-skjema
- Riksarkivet:
  - NOARK-4 bruker DTD
  - NOARK-5 bruker XSD
- Bruken av DTD
  - mangler noe fleksibilitet og mens det er viktig å forstå dem, er XSD viktigere (for arkiv)
  - Tidsperspektiv, når vil depot ikke lenger ta imot Noark4 avlevering?

XML Skjema (XSD)

- XML-skjema (XSD)
  - Basert på XML og dermed kraftigere
  - Støtter navnerom og datatyper

XML Skjema

- XML Skjema kan brukes til å
  - beskrive tillatt dokumentinnhold
  - kontrollere riktigheten av data
  - jobbe med data fra en database
  - definere data fasetter (restriksjoner på data)
  - definere datamønstre (dataformater)
  - konvertere data mellom ulike datatyper

XML Skjema

- XML Skjema definerer
  - *elementer* i et XML dokument
  - *attributter* i et XML dokument
  - *underordnede* elementer
  - *rekkefølgen* på underordnede elementer
  - *antall* underordnede elementer
  - om et element kan være tom, eller **må** inneholde noe
  - *datatyper* for elementer og attributter
  - *standard* og *faste* verdier for elementer og attributter

XML Kurs for eArkivar

Time 3 : Oppbygging og validering

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 12/10/2012

Ofte brukt XSD Datatyper

- string == Streng
  - Feks «Sak om rehabiliterings søknad»
- boolean == boolsk
  - Feks Ja eller Nei
- integer == Heltall
  - 345858
- decimal == Desimaler
  - 3.1472748494
- date == dato
  - ÅÅÅÅ.MM.DD format feks 2011.01.24
- time = tid
  - TT:MM:SS feks 12:15:00
- dateTime == datoTid
  - ÅÅÅÅ.MM.DDTTT:MM:SS format feks 2011.01.24T12:15:00

Numeriske Datatyper

byte	8-bit signert heltal	-128 til 127
decimal	desimal	Ikke begrenset
int	32-bit signert heltal	Fra − 2,147,483,648 til 2,147,483,647
integer	heltal	Ikke begrenset
long	64-bit signert heltal	Fra − 9,223,372,036,854,775,808 til 9,223,372,036,854,775,807
negativeInteger	heltal som kun består negative verdier	Ikke begrenset (..,-2,-1)
nonNegativeInteger	heltal som kun består av ikke negative verdier	Ikke begrenset (0,1,2,..)
nonPositiveInteger	heltal som kun består av ikke-positive verdier	Ikke begrenset (..,-2,-1,0)
positiveInteger	heltal som kun består av positive verdier	Ikke begrenset (1,2,..)
short	16-bit signert heltal	Fra − 32,768 to 32,767
unsignedByte	8-bit usignert heltal	Fra 0 til 255
unsignedInt	32-bit usignert heltal	Fra 0 til 4,294,967,295
unsignedShort	16-bit usignert heltal	Fra 0 til 65,535
unsignedLong	64-bit usignert heltal	Fra 0 til 18,446,744,073,709,551,615

Definer elementer

- Vi begynner med å definere elementer
  - simpleType
  - complexType

simpleType (enkeltype)

- simpleType
  - Format:
    - <xs:element name="" type=""/>

Definer navnet på elementet

Definer datatypen til elementet

- Kan ikke inneholde andre elementer, bare data
  - <xs:element name="etternavn" type="xs:string"/>
  - <xs:element name="alder" type="xs:integer"/>
  - <xs:element name="fødselsdato" type="xs:date"/>

complexType (komplekstype)

- complexType
  - Brukes til å definere restriksjoner
  - Elementer som bare inneholder andre elementer
  - Tomme elementer
  - Elementer som bare inneholder tekst
  - Elementer som inneholder både andre elementer og tekst

Vi ser ikke på disse

```
<xs:element name="navn"/>
   <xs:complexType>
     <xs:sequence>
         <xs:element name="fornavn" type="xs:string"/>
         <xs:element name="etternavn" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
```

complexType

- complexType består vanligvis av andre elementer
- complexType brukes for å lage et tre av elementer

```
<navn>
  <fornavn> </fornavn>
  <etternavn> </etternavn>
</navn>
```

xml

```
1  <xs:element name="navn">
2    <xs:complexType>
3      <xs:sequence>
4a       <xs:element name="fornavn" type="xs:string"/>
4b       <xs:element name="etternavn" type="xs:string"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
```

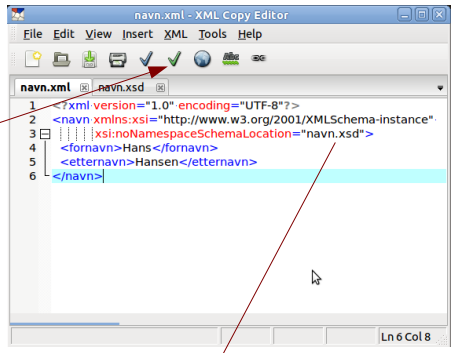
xsd

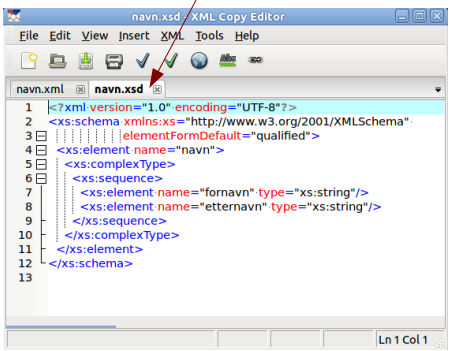
1. Definer et element som heter <navn>
2. Definer at dette element består av noen under elementer
3. Den består av et sett underelementer

4a/4b. Den består av to stk simpleType elementer, <fornavn> og <etternavn>

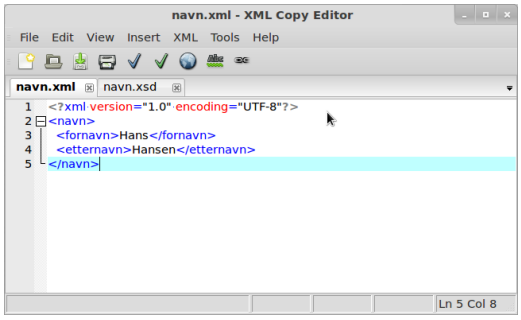
Hvordan validere

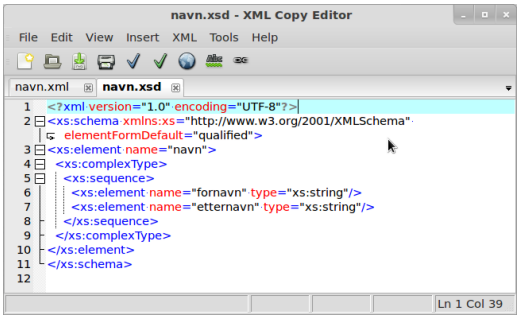
Vi kan bruke validerings verktøyet (den grønne knappen) til xmlcopyeditor men vi er nødt til å lage en XSD file og assosiere den med XML filen

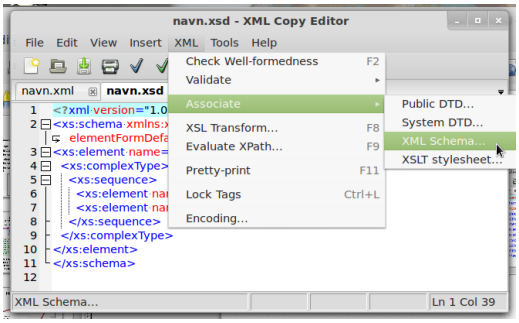


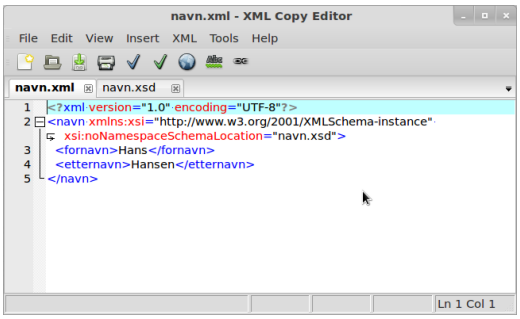


Assosiere en xsd fil med en xml fil









Praktisk

```
<?xml version="1.0" encoding="UTF-8"?>

<navn xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="navn.xsd">

  <fornavn> Hans </fornavn>
  <etternavn> Hansen </etternavn>
</navn>
```

navn.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

      <xs:element name="navn">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="fornavn" type="xs:string"/>
                        <xs:element name="etternavn" type="xs:string"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

navn.xsd

simpleType/complexType

- simpleType kan sees på som en byggekloss, der en byggekloss er et enkel element
- complexType er det som lar deg sette disse byggeklossene sammen til å lage fornuftige ting
  - I arkivsammenheng
    - <arkiv>
    - <arkivdel>
    - .....
    - </arkivdel>
    - </arkiv>

menneske.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneske>
    <fornavn>Hans</fornavn>
    <etternavn>Hansen</etternavn>
    <personnummer>01108298649</personnummer>
</menneske>
```

menneske.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

</xs:schema>
```

menneske.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="menneske">

    </xs:element>
</xs:schema>
```

menneske.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="menneske">
        <xs:complexType>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

menneske.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="menneske">
        <xs:complexType>
            <xs:sequence>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

menneske.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="menneske">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="fornavn" type="xs:string"/>
                <xs:element name="etternavn" type="xs:string"/>
                <xs:element name="personnummer" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

menneske.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneske  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="menneske.xsd">

    <fornavn>Hans</fornavn>
    <etternavn>Hansen</etternavn>
    <personnummer>01108298649</personnummer>

</menneske>
```

Med assosiert xsd fil menneske.xsd

menneske.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneske xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="menneske.xsd">

    <fornavn>Hans</fornavn>
    <mellomnavn>John</mellomnavn>
    <etternavn>Hansen</etternavn>
    <personnummer>01108298649</personnummer>
    <kjønn>mann</kjønn>

</menneske>
```

Nå bygger vi ut eksemplet med <mellomnavn> og <kjønn>

menneske.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="menneske">
        <xs:complexType>
                <xs:sequence>
                <xs:element name="fornavn" type="xs:string"/>
                <xs:element name="mellomnavn" type="xs:string"/>
                <xs:element name="etternavn" type="xs:string"/>
                <xs:element name="personnummer" type=xs:string"/>
                <xs:element name="kjønn" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

menneskeListe.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneskeListe>
    <menneske>
        <fornavn>Hans</fornavn>
        <etternavn>Hansen</etternavn>
        <personnummer>01108298649</personnummer>
    </menneske>
</menneskeListe>
```

menneskeListe.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="menneske">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="fornavn" type="xs:string"/>
                <xs:element name="mellomnavn" type="xs:string"/>
                <xs:element name="etternavn" type="xs:string"/>
                <xs:element name="personnummer" type="xs:string"/>
                <xs:element name="kjønn" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="menneskeListe">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="menneske" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

menneskeListe.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<menneskeListe xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="menneskeListe.xsd">
    <menneske>
        <fornavn>Hans</fornavn>
        <mellomnavn>John</mellomnavn>
        <etternavn>Hansen</etternavn>
        <personnummer>01108298649</personnummer>
    </menneske>
</menneskeListe>
```

XML Kurs for eArkivar

Time 4 : Mer praktisk øvelse

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 12/10/2012

Denne timen

- Her fortsetter vi med bok.xml og arkiv.xml
- Ser på hvordan vi definerer attributter

bok.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bok>
        <forfatter>Hans Hansen</forfatter>
        <boktittel>Boken om XML</boktittel>
            <kapittel>
                <kapitteltittel>Introduksjon</kapitteltittel>
                <paragraf></paragraf>
                <paragraf></paragraf>
        </kapittel>
        <kapittel>
                <kapitteltittel>XML Rotelementet</kapitteltittel>
                <paragraf></paragraf>
                <paragraf></paragraf>
        </kapittel>
</bok>
```

bok.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">


    <xs:element name="bok">
        <xs:complexType>
            <xs:sequence>
            </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

bok.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">


    <xs:element name="bok">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="forfatter" type="xs:string"/>
                <xs:element name="boktittel" type="xs:string"/>
                <xs:element ref="kapittel" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
     </xs:element>
</xs:schema>
```

bok.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="kapittel">
        <xs:complexType>
            <xs:sequence>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="bok">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="forfatter" type="xs:string"/>
                <xs:element name="boktittel" type="xs:string"/>
                <xs:element ref="kapittel" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
     </xs:element>
</xs:schema>
```

bok.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

    <xs:element name="kapittel">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="kapitteltittel" type="xs:string"/>
                <xs:element name="paragraf" type="xs:string" maxOccurs="unbounded/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="bok">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="forfatter" type="xs:string"/>
                <xs:element name="boktittel" type="xs:string"/>
                <xs:element ref="kapittel" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
     </xs:element>
</xs:schema>
```

arkiv.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<arkiv>
 <opprettetDato>1-12-2009</opprettetDato>
 <avsluttetDato>5-10-2012 </avsluttetDato>
 <systemID>E9F3C110-F8C4-11E1-A21F-0800200C9A66</systemID>
 <tittel>Kollektivtransport</tittel>
 <arkivdel>
  <opprettetDato>1-12-2009</opprettetDato>
  <avsluttetDato>5-10-2012 </avsluttetDato>
  <systemID>BB46B6E9-D81E-4D3F-8450-4C991F44BD98</systemID>
  <tittel>Innkjøp</tittel>
  <mappe>
   <opprettetDato>1-12-2009</opprettetDato>
   <avsluttetDato>5-10-2012 </avsluttetDato>
   <systemID>481C46C5-EF2A-4AEA-BA51-93233DF331FC</systemID>
   <tittel>Innkjøp av busser fra MAN</tittel>
  </mappe>
 </arkivdel>
</arkiv>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="qualified">

 <xs:element name="arkiv">
  <xs:complexType>
   <xs:sequence>
    <xs:element type="xs:string" name="opprettetDato"/>
    <xs:element type="xs:string" name="avsluttetDato"/>
    <xs:element type="xs:string" name="systemID"/>
    <xs:element type="xs:string" name="tittel"/>
    <xs:element name="arkivdel">
     <xs:complexType>
      <xs:sequence>
       <xs:element type="xs:string" name="opprettetDato"/>
       <xs:element type="xs:string" name="avsluttetDato"/>
       <xs:element type="xs:string" name="systemID"/>
       <xs:element type="xs:string" name="tittel"/>
       <xs:element name="mappe">
        <xs:complexType>
         <xs:sequence>
          <xs:element type="xs:string" name="opprettetDato"/>
          <xs:element type="xs:string" name="avsluttetDato"/>
          <xs:element type="xs:string" name="systemID"/>
          <xs:element type="xs:string" name="tittel"/>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

arkiv.xsd

Noe rart med xsd?

- Det var ikke jeg som lagde arkiv.xml
  - http://www.freeformatter.com/
- Det finnes verktøy som kan lage en xsd fra xml
- Nyttig spesielt hvis du har fått en xml fil fra et fagsystem for å få oversikt over innhold
  - Du kan plutselig oppdage to forskjellige felter som skal være det samme
    - <opprettetDato>
    - <oppretteDato> eller <OpprettetDato>

xsd: attributter

Attributter kan defineres på følgende måte

```
<xs:element name="kunde">
    <xs:complexType>
        <xs:attribute name="kundeID"
                      type="xs:integer"
                      use="required"/>
    </xs:complexType>
</xs:element>
```

```
<kunde kundeID='123'> Thomas </kunde>
<kunde> Johanne </kunde>
<kunde kundeID='124'> Liv </kunde>
<kunde kundeID='X98923'> Pål </kunde>
```

XML Kurs for eArkivar

Time 5 : Innhold og restriksjoner

KDRS
Kommunearkivinstitusjonenes Digitale RessursSenter

Trondheim 12/10/2012

Fra struktur til innhold ....

- Så langt har vi egentlig bare sett
  - xml syntaks regler må følges (korrekt strukturert)
  - Definert oppbygging/struktur med complexType
- Vi har ikke sett på innholdet i elementer
- Det er det vi ser på i denne time

<xs:sequence>

- <xs:sequence> brukes til å definere en sekvens av elementer
  - Alle elementer må forekomme i sekvensen som er oppgitt
    - Hvis ikke minOccurs = 0

menneskeListe.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

    <xs:element name="menneske">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="fornavn" type="xs:string"/>
                <xs:element name="mellomnavn" type="xs:string"/>
                <xs:element name="etternavn" type="xs:string"/>
                <xs:element name="personnummer" type="xs:string"/>
                <xs:element name="kjønn" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="menneskeListe">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="menneske" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

maxOccurs/minOccurs

- Kan brukes til å angi hvor mange ganger et element kan/må forekomme og bruken er valgfritt
- Gjør at
  - et minimum med data er fanget
  - hindre duplisert innhold (feks personnummer)
- maxOccurs
  - Angir det *maksimale* antall ganger elementet kan forekomme
  - Verdien er >= 0, eller "unbounded"
- minOccurs
  - Angir *minimum* antall ganger elementet kan forekomme
  - Verdien er >= 0

<xs:all>

- <xs:all> brukes til å definere elementer der
  - rekkefølgen er ikke viktig
  - elementer kan forekomme 0 eller 1 gang

Alternativ menneskeListe.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">

     <xs:element name="menneske">
          <xs:complexType>
               <xs:all>
                    <xs:element name="fornavn" type="xs:string"/>
                    <xs:element name="mellomnavn" type="xs:string"/>
                    <xs:element name="etternavn" type="xs:string"/>
                    <xs:element name="personnummer" type="xs:string"/>
                    <xs:element name="kjønn" type="xs:string"/>
               </xs:all>
          </xs:complexType>
     </xs:element>
     <xs:element name="menneskeListe">
          <xs:complexType>
               <xs:sequence>
                    <xs:element ref="menneske" maxOccurs="unbounded"/>
               </xs:sequence>
        </xs:complexType>
       </xs:element>
</xs:schema>
```

<xs:choice>

- <xs:choice> kan brukes for å definere en enten eller forhold

```
.....
<!-- Dette er en del av filen arkivstruktur.xsd -->
<!-- Dette er fra delen som definerer <arkiv> elementet -->
<!-- Vil alltid inneholde minst en forekomst av enten arkiv eller arkivdel -->
 <xs:element name="arkiv" type="arkiv"/>
  <xs:complexType name="arkiv">
    <xs:sequence>
      <xs:choice>
        <xs:element name="arkiv" type="arkiv" maxOccurs="unbounded"/>
        <xs:element name="arkivdel" type="arkivdel" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
.....
```

Tilbake til datatyper

- string == Streng
  - Feks «Sak om rehabiliterings søknad»
- boolean == boolsk
  - Feks Ja eller Nei
- integer == Heltall
  - 345858
- decimal == Desimaler
  - 3.1472748494
- date == dato
  - ÅÅÅÅ.MM.DD format feks 2011.01.24
- time = tid
  - TT:MM:SS feks 12:15:00
- dateTime == datoTid
  - ÅÅÅÅ.MM.DDTTT:MM:SS format feks 2011.01.24T12:15:00

XSD restriksjoner

- Vi kan bruke XSD restriksjoner til å si noe om tillatt innhold
- For streng:
  - Definere minimums lengde, maksimums lengde, eksakt lengde
  - Definere lovlig innhold
  - Begrense mønsteret på innholdet
- For tall
  - Definere minimum og maksimums verdier
  - Definere lovlig innhold

xsd streng restriksjoner

enumeration

enumeration restriksjonen kan brukes til begrense gyldige verdier på innholdet til et element

```
<xs:element name="arkivdelstatus">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Aktiv periode"/>
      <xs:enumeration value="Overlappingsperiode"/>
      <xs:enumeration value="Avsluttet periode"/>
      <xs:enumeration value="Uaktuelle mapper"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<arkivdelstatus>Aktiv periode</arkivdelstatus>
<arkivdelstatus>Uaktuelle mapper</arkivdelstatus>
<arkivdelstatus> Uaktuelle mapper </arkivdelstatus>
<arkivdelstatus>aktiv periode</arkivdelstatus>
```

xsd streng restriksjoner

pattern

pattern restriksjonen kan brukes til angi nøyaktig hvilken tegn en streng kan inneholde

```
<xs:element name="kjønn">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="mann|kvinne"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<kjønn> mann </kjønn>
<kjønn> han </kjønn>
<kjønn> kvinne </kjønn>
<kjønn> hun </kjønn>
```

Men vi kunne brukt enumeration restriksjonen for å løse dette

xsd streng restriksjoner

pattern

pattern restriksjonen kan brukes til angi nøyaktig hvilken tegn en streng kan inneholde

```
<xs:element name="studentNummer">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:pattern value="[s][0-9][0-9][0-9][0-9][0-9][0-9]"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

```
<studentNummer>s123456</studentNummer>
<studentNummer>S123456</studentNummer>
<studentNummer>s000000</studentNummer>
<studentNummer>s12345</studentNummer>
```

xsd streng restriksjoner

minLength og maxLength

minLength og *maxLength* restriksjonen kan brukes til angi et minimum/maksimum lengde på en streng

```
<xs:element name="brukernavn">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:minLength value="5"/>
   <xs:maxLength value="20"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

```
<brukernavn> tsodring </brukernavn>
<brukernavn> tom </brukernavn>
<brukernavn> thomasdenmektige </brukernavn>
<brukernavn> thomasdenstorevikingen </brukernavn>
```

xsd streng restriksjoner

length

length restriksjonen kan brukes til angi en eksakt lengde på en streng

```
<xs:element name="personnummer">
  <xs:simpleType>
    <xs:restriction base="xs:string">
          <xs:length value="11"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<personnummer>01107848874</personnummer>
<personnummer>0104784992</personnummer>
<personnummer>01047849922</personnummer>
<personnummer>010478499223</personnummer>
```

xsd tall restriksjoner

totalDigits

totalDigits restriksjonen kan brukes til angi nøyaktig hvor mange siffer innholdet kan bestå av

```
<xs:element name="personnummer">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
          <xs:totalDigits value="11"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<personnummer>01107848874</personnummer>
<personnummer>0104784992</personnummer>
<personnummer>01047849922</personnummer>
<personnummer>010478499223</personnummer>
```

Hva med -01107848874? er personnummer streng/tall? Kanskje bruke positiveInteger som datatype?

xsd tall restriksjoner

minInclusive og maxInclusive

minInclusive og *maxInclusive* restriksjonen kan brukes til angi et spekter av gyldige verdier (inkluderer verdien som er oppgitt)

```
<xs:element name="år">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="1900"/>
   <xs:maxInclusive value="2000"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

```
<år>1900</år>
<år>1899</år>
<år>2000</år>
<år>2001</år>
```

xsd tall restriksjoner

minExclusive og maxExclusive

minExclusive og *maxExclusive* restriksjonen kan også brukes til angi et spekter av gyldige verdier (ekskluderer verdien som er oppgitt)

```
<xs:element name="år">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minExclusive value="1900"/>
   <xs:maxExclusive value="2000"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

```
<år>1901</år>
<år>1900</år>
<år>1999</år>
<år>2000</år>
```

xsd dato restriksjoner

- Datoer kan bruke følgende restriksjoner
  - Enumeration
    - Velge fra en liste datoer
  - minInclusive og maxInclusive
    - Definere en periode av verdier
  - Pattern
    - Definere tillat innhold

Avslutning

- Vi har sett på de viktige restriksjonene for tall, dato og streng
  - Men det er flere datatyper
- Dette er på en intro nivå for å bli kjent
- For å bli flink med dette handler det om ***øvelse***